

linux firewalls

PAralax Elements

linux firewalls

**konzeption und implementierung für
kleine netzwerke und PCs**

robert l. ziegler
übersetzt von
marc-andré selig

 **new technology**

Bitte beachten Sie:

Der originalen Printversion liegt
eine CD-ROM bei.

In der vorliegenden elektronischen
Version ist die Lieferung einer
CD-ROM nicht enthalten.

Alle Hinweise und alle Verweise
auf die CD-ROM sind ungültig.

Markt+Technik Verlag

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich.

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen
eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter
Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben
und deren Folgen weder eine juristische Verantwortung noch
irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und
Herausgeber dankbar.

Autorisierte Übersetzung der amerikanischen Originalausgabe:
Linux Firewalls © by New Riders Publishing

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der
Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten
ist nicht zulässig.

Fast alle Hardware- und Softwarebezeichnungen, die in diesem Buch erwähnt werden,
sind gleichzeitig auch eingetragene Warenzeichen oder sollten als solche betrachtet
werden.

Umwelthinweis:
Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt.
Die Einschrumpffolie – zum Schutz vor Verschmutzung – ist aus
umweltverträglichem und recyclingfähigem PE-Material.

10 9 8 7 6 5 4 3 2 1

03 02 01 00

ISBN 3-8272-5849-9

© 2000 by Markt+Technik Verlag,
ein Imprint der Pearson Education Deutschland GmbH,
Martin-Kollar-Straße 10–12, D-81829 München/Germany
Alle Rechte vorbehalten
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Deutsche Übersetzung, Fachlektorat und inhaltliche Ergänzung: Marc-André Selig, München
Herstellung: Ulrike Hempel, uhempel@pearson.de
Sprachliches Lektorat: Karl-Heinz Stemberg, Münster
Satz: reemers publishing services gmbh, Krefeld
Druck und Verarbeitung: Bercker, Kevelaer
Printed in Germany

One of my two lifelong best friends died while the book was being written. To Gloria Frawley, who across the lifetimes has been friend, lover, brother, sister, parent, child, husband, wife. Until we walk together again, thank you for the purest, most trusting, nonjudgmental love I've known, and the best playmate I could ever hope for.

Inhaltsverzeichnis

	Einführung	17
	E.1 Übersicht	18
	E.2 Was dieses Buch nicht beschreibt	19
	E.3 Einbruchsversuche: Dimension eines Problems	20
	E.4 Was hat der Eindringling davon?	20
	E.5 Was steht für Sie auf dem Spiel?	21
	E.6 Firewalls und Black Hats in einer idealen Welt	22
Teil 1	Vorüberlegungen	23
Kapitel 1	Grundlegende Konzepte für Paketfilter-Firewalls	25
	1.1 Das OSI-Referenzmodell	27
	1.2 Ports: Tore zu den Programmen auf Ihrem Computer	30
	1.3 Pakete: Botschaften im IP-Netz	32
	1.3.1 IP-Nachrichtentypen: ICMP	33
	1.3.2 IP-Nachrichtentypen: UDP	34
	1.3.3 IP-Nachrichtentypen: TCP	35
	1.4 Zusammenfassung	39
Teil 2	Paketfilter und grundlegende Sicherheitsmaßnahmen	41
Kapitel 2	Konzepte für Paketfilter	43
	2.1 Eine Paketfilter-Firewall	45
	2.2 Auswahl der Voreinstellung für den Umgang mit Paketen	47
	2.3 REJECT und DENY: Pakete ablehnen oder verwerfen?	50
	2.4 Filtern ankommender Pakete	51
	2.4.1 Filtern nach Absender-IP	51
	2.4.2 Filtern nach Empfänger-IP	54
	2.4.3 Filtern nach Absender-Port	54

2.4.4	Filtern nach Empfänger-Port	55
2.4.5	Filtern nach TCP-Status-Flags	55
2.4.6	Abtastversuche und Scans	55
2.4.7	Denial-of-Service-Angriffe	59
2.4.8	Weitere Überlegungen zum Filtern von Paketen	63
2.5	Filtern abgehender Pakete	64
2.5.1	Filtern nach Absender-IP	65
2.5.2	Filtern nach Empfänger-IP	65
2.5.3	Filtern nach Absender-Port	66
2.5.4	Filtern nach Empfänger-Port	66
2.5.5	Filtern nach TCP-Status-Flags	66
2.6	Private und öffentliche Dienste im Netz	67
2.6.1	Unsichere lokale Dienste schützen	68
2.6.2	Auswahl der Dienste	68
2.7	Zusammenfassung	86
Kapitel 3	Gestaltung und Installation einer Firewall	87
3.1	ipchains: Firewall-Verwaltung unter Linux	88
3.1.1	Im Firewall-Skript eingesetzte ipchains-Optionen	90
3.1.2	IP-Adressen von Absender und Empfänger	91
3.2	Initialisierung der Firewall	93
3.2.1	Symbolische Konstanten für die Beispiele	94
3.2.2	Löschen alter Firewall-Regeln	94
3.2.3	Festlegung der voreingestellten Policy	95
3.2.4	Einschalten des Loopback-Interfaces	95
3.2.5	Gefälschte Absender und andere fehlerhafte Adressen	96
3.3	ICMP-Nachrichten filtern	103
3.3.1	Fehler- und Kontrollnachrichten	103
3.3.2	Kontrollnachrichten für ping: echo-request (Typ 8) und echo-reply (Typ 0)	106
3.4	Dienste auf festen, unprivilegierten Ports schützen	108
3.4.1	Häufige TCP-Dienste auf unprivilegierten Ports	109
3.4.2	Lokale UDP-Dienste auf unprivilegierten Ports	111

3.5	Zwingend benötigte Dienste erlauben	114
3.5.1	DNS (UDP- und TCP-Port 53)	114
3.5.2	Der auth-Service (TCP-Port 113)	120
3.6	Häufige TCP-Dienste	121
3.6.1	E-Mail (TCP: SMTP – Port 25; POP – Port 110; IMAP – Port 143)	122
3.6.2	Usenet News (NNTP: TCP-Port 119)	129
3.6.3	telnet (TCP-Port 23)	131
3.6.4	ssh (TCP-Port 22)	133
3.6.5	ftp (TCP-Ports 20 und 21)	135
3.6.6	World Wide Web	139
3.6.7	finger (TCP-Port 79)	142
3.6.8	whois (TCP-Port 43)	143
3.6.9	gopher (TCP-Port 70)	144
3.6.10	WAIS (TCP-Port 210)	145
3.7	Häufige UDP-Dienste	145
3.7.1	traceroute (UDP-Port 33434)	145
3.7.2	Zugriff auf den DHCP-Server Ihres Internet-Providers (UDP-Ports 67 und 68)	147
3.7.3	Network-Time-Server abfragen (UDP-Port 123)	150
3.8	Abgewiesene Pakete protokollieren	151
3.9	Zugriff für problematische Sites pauschal sperren	153
3.10	LAN-Zugang	154
3.10.1	LAN-Zugang zum internen Netzwerkinterface der Firewall	154
3.10.2	LAN-Zugriffe auf das Internet: IP-Forwarding und -Masquerading	155
3.11	Die Firewall installieren	156
3.11.1	Installation einer Firewall mit einer statischen IP-Adresse	156
3.11.2	Installation einer Firewall für PPP	157
3.11.3	Installation einer Firewall für DHCP	158
3.12	Zusammenfassung	160

Kapitel 4	LANs, mehrfache Firewalls und Perimeter-Netze	161
4.1	Sicherheit im LAN	163
4.2	Konfigurationsmöglichkeiten für ein privates LAN mit vertrauenswürdigen Benutzern	164
4.2.1	LAN-Zugriffe auf die Bastion-Firewall	166
4.2.2	LAN-Zugriffe auf andere LANs: Lokale Pakete zwischen mehreren lokalen Netzen weiterleiten	166
4.2.3	LAN-Zugriffe aufs Internet: Forwarding und Masquerading	167
4.3	Konfigurationsmöglichkeiten für ein größeres oder weniger vertrauenswürdiges LAN	169
4.3.1	Bildung mehrerer Subnetze	169
4.3.2	Selektiver interner Zugang nach Host, Adressbereich oder Port	170
4.3.3	Masquerading zwischen LAN und Internet	178
4.3.4	Portumleitung und transparente Proxies	181
4.3.5	Aus dem Internet ankommende Verbindungen zu internen Servern umleiten	181
4.4	Eine formale Firewall mit abgeschirmtem Subnetz	183
4.4.1	Symbolische Konstanten für die Firewall-Beispiele	185
4.4.2	Löschen alter Firewall-Regeln	187
4.4.3	Festlegung der voreingestellten Policy für die Choke	187
4.4.4	Einschalten des Loopback-Interfaces auf der Choke	187
4.4.5	Gefälschte Absender und andere fehlerhafte Adressen (Klassen A bis C)	188
4.4.6	ICMP-Nachrichten filtern	189
4.4.7	DNS (UDP- und TCP-Port 53)	194
4.4.8	Der auth-Service (TCP-Port 113)	200
4.4.9	E-Mail (SMTP: TCP-Port 25; POP: TCP-Port 110; IMAP: TCP-Port 143)	203
4.4.10	Usenet News (TCP: NNTP-Port 119)	214
4.4.11	telnet (TCP-Port 23)	216
4.4.12	ssh (TCP-Port 22)	218
4.4.13	ftp (TCP-Ports 21 und 22)	222

4.4.14	WWW	232
4.4.15	finger (TCP-Port 79)	243
4.4.16	whois (TCP-Port 43)	245
4.4.17	gopher (TCP-Port 70)	246
4.4.18	WAIS (TCP-Port 210)	247
4.4.19	RealAudio und QuickTime (Ports 554 u.a.)	249
4.4.20	Internet Relay Chat IRC (TCP-Port 6667)	253
4.4.21	CU-SeeMe (UDP-Port 7648, 7649 und 24032; TCP-Ports 7648 und 7649)	258
4.4.22	Quake (UDP-Ports 26000 sowie 1025 bis 1200)	262
4.4.23	Der Network Time Service NTP (UDP-Port 123)	266
4.4.24	Protokolle an einen anderen Computer schicken (UDP-Port 514)	269
4.4.25	Die Choke als lokaler DHCP-Server (UDP-Ports 67 und 68)	271
4.4.26	LAN-Zugriff auf die Choke	272
4.4.27	IP-Masquerading	272
4.4.28	Protokollierung	273
4.5	Zusammenfassung	273
Kapitel 5	Fehlersuche	275
5.1	Ein paar allgemeine Tipps für die Firewall-Entwicklung	276
5.2	Anzeigen der Firewall-Regeln	278
5.2.1	ipchains -L input	278
5.2.2	ipchains -L input -n	279
5.2.3	ipchains -L input -v	281
5.2.4	ipchains -L input -nv	283
5.3	Die Regeln für die input-, output- und forward-Chains	284
5.3.1	Die input-Chain	284
5.3.2	Die output-Chain	286
5.3.3	Die forward-Chain	289
5.4	Die Firewall-Regeln mit Einzelpaketen testen	290

5.5	Suche nach offenen Ports	292
5.5.1	netstat -a [-n -p -A inet]	292
5.5.2	strobe	295
5.5.3	nmap	296
5.6	Fehlersuche für ssh – ein Beispiel aus der Praxis	296
5.7	Zusammenfassung	299
Teil 3	Systemsicherheit und Systemüberwachung	301
Kapitel 6	Läuft das System wie erwartet?	303
6.1	Überprüfen der Netzwerkschnittstellen mit ifconfig	304
6.2	Überprüfen der Netzwerkverbindung mit ping	305
6.3	Überprüfen der Netzwerkprozesse mit netstat	307
6.4	Überprüfen aller laufenden Prozesse mit ps ax	308
6.5	Die Protokolldateien Ihres Systems	311
6.5.1	Was wird wohin geschrieben?	311
6.5.2	Konfiguration des syslog	312
6.5.3	Was bedeuten die Meldungen der Firewall?	314
6.5.4	Häufig gescannte Ports	316
6.5.5	Pakete zur automatisierten Protokollanalyse	321
6.6	Zusammenfassung	322
Kapitel 7	Ergänzende Maßnahmen durch die Unix-Systemadministration	323
7.1	Authentifizierung: Prüfung der Identität	324
7.1.1	Shadow-Passwörter	324
7.1.2	MD5-Hashes für Passwörter	325
7.1.3	Die rhost-Authentifizierung von Berkeley: hosts.equiv und .rhosts	326
7.1.4	Gemeinsamer Zugang zu zentralen Authentifizierungsdatenbanken: der Network Information Service (NIS)	326
7.2	Autorisation: Festlegung von Zugriffsrechten	327
7.2.1	Zugriff auf den root-Account	327
7.2.2	Verwendung von su einschränken	328

7.2.3	Die tcp_wrapper	328
7.2.4	Zugriffsrechte für Dateien und Verzeichnisse	332
7.3	Serverspezifische Konfiguration	334
7.3.1	telnet-Konfiguration	334
7.3.2	ssh-Konfiguration	334
7.3.3	SMTP-Konfiguration	335
7.3.4	DNS-Konfiguration	337
7.3.5	ftp-Konfiguration	359
7.3.6	Konfiguration eines POP-Servers	365
7.3.7	Konfiguration eines DHCP-Servers	366
7.3.8	NTP-Konfiguration	368
7.3.9	Konfiguration von CGI-Skripten	370
7.4	SOCKS: eine Proxy-Firewall auf Anwendungsebene	371
7.5	Diverse Systemaccounts in /etc/passwd und /etc/group	372
7.6	Die PATH-Variable	373
7.7	/etc/issue.net	374
7.8	Protokollierung auf andere Rechner	375
7.9	Halten Sie Ihre Software auf dem Laufenden!	376
7.9.1	Updates von Red Hat	376
7.9.2	Beispiel: Sicherheitslücke in mountd	376
7.10	Zusammenfassung	377
Kapitel 8	Entdecken von Eindringlingen und Melden der Vorfälle	379
8.1	Prüfung der Systemintegrität	381
8.1.1	COPS	381
8.1.2	crack	381
8.1.3	ifstatus	382
8.1.4	MD5	382
8.1.5	SATAN	382
8.1.6	tiger	383
8.1.7	tripwire	383
8.2	Welche Symptome deuten auf einen Eindringling hin?	383
8.2.1	Hinweise aus dem syslog	384
8.2.2	Änderungen der Systemkonfiguration	385

8.2.3	Änderungen am Dateisystem	385
8.2.4	Änderungen an Benutzer-Accounts	386
8.2.5	Hinweise der Überwachungswerkzeuge	386
8.2.6	Geschwindigkeitseinbußen	387
8.3	Reaktion auf einen Einbruch	387
8.4	Meldung eines Vorfalls	389
8.4.1	Warum sollten Sie Vorfälle melden?	389
8.4.2	Welche Vorfälle sollten Sie melden?	390
8.4.3	Wem melden Sie?	392
8.4.4	Was melden Sie?	393
8.5	Wo finden Sie nähere Informationen?	394
8.6	Zusammenfassung	395
Teil 4	Anhänge	397
Anhang A	Ressourcen rund um das Thema Sicherheit	399
A.1	Informationsquellen	400
A.2	Softwaresammlungen	401
A.3	Sicherheitstools	401
A.4	Firewall-Tools	404
A.5	Nachschlagewerke und FAQ-Listen	404
A.5.1	Sicherheit unter Unix	404
A.5.2	Rund um Firewalls	405
A.5.3	Web-Server	406
A.6	Online-Dokumentation	406
A.7	Web-Sites zu allgemeinen Themen	407
A.8	Bücher	408
Anhang B	Firewall-Beispiele und hilfreiche Skripten	409
B.1	Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3	410
B.2	Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3	431

Inhaltsverzeichnis 15

B.3	Optimierung der Firewall-Regeln	451
B.3.1	Optimierung der Reihenfolge der ipfwadm-Regeln	462
B.3.2	Optimierung der ipchains-Regeln mit benutzerdefinierten Chains	471
B.4	Spezialskripten	485
B.4.1	Alles erlauben	485
B.4.2	Alles sperren	485
B.4.3	Eine IP-Adresse blockieren	486
B.4.4	Eine blockierte IP-Adresse wieder zulassen	487

Anhang C	Glossar	489

	Der Autor	511

	Stichwortverzeichnis	513

Einführung

- E.1 Übersicht**
- E.2 Was dieses Buch nicht beschreibt**
- E.3 Einbruchsversuche: Dimension eines Problems**
- E.4 Was hat der Eindringling davon?**
- E.5 Was steht für Sie auf dem Spiel?**
- E.6 Firewalls und Black Hats in einer idealen Welt**



Linux erfreut sich bei Fans und kleinen Privatfirmen wachsender Beliebtheit. Gleichzeitig werden Internet-Flatrates über Kabelmodems und DSL-Verbindungen immer verbreiteter.

Unix ist nicht nur eine beliebte Serverplattform, vor allem für das WWW, sondern es eignet sich auch hervorragend als Gateway eines Hausnetzes. Hinter diesem Gateway befinden sich andere Maschinen, die rund um die Uhr an das Internet angeschlossen sind: Rechner unter Unix, Windows, NT, MacOS, oder auch vernetzte Drucker. Das bedeutet jedoch, dass die Anwender kleiner Systeme immer mehr mit Sicherheitsproblemen zu tun bekommen, an die sie vorher nicht einmal denken mussten.

Netzwerksicherheit ist besonders wichtig für Linux-Anwender, die einen direkten Internetzugang haben. Im Gegensatz zu einfacheren PC-Systemen ist Unix ein vollwertiges, mächtiges Betriebssystem. Es wurde entwickelt, damit man bei Forschung und Entwicklung auf gemeinsame Ressourcen zugreifen konnte. Dadurch wurde es groß, schwer verständlich und ziemlich ungeeignet für unerfahrene und ungeschützte Anwender.

Wenn man ein solches Unix-System an das Internet anschließt, ist das so ähnlich, als würde man seine Haustür sperrangelweit öffnen, für ein paar Wochen in den Urlaub fahren und das Ganze auch noch per Zeitungsinserat bekannt geben. Ergreift man keine zusätzlichen Vorsichtsmaßnahmen, wird man in beiden Fällen Eindringlinge zu Besuch haben, und zwar eher früher als später.

Der durchschnittliche Anwender eines Privat-PCs oder eines kleinen Firmenrechners hat nicht die Zeit, das Interesse und die Geduld, alle nötigen Feinheiten der Systemsicherheit zu erlernen. Dieses Buch will ihm helfen, innerhalb kürzester Zeit die notwendigen Sicherheitsmaßnahmen umzusetzen, ohne dass er zuerst ein Sicherheitsexperte für Computernetzwerke werden müsste. Diese Maßnahmen sind an sich gar nicht so schwierig, aber es kann ein Problem sein, alle relevanten Informationen zusammenzustellen und auch noch herauszufinden, wie man sie in der Praxis anwendet.

E.1 Übersicht

Für die Anwender kleiner Systeme bezieht sich »Sicherheit« fast ausschließlich auf die *externe Sicherheit*, also auf den Schutz vor ungewolltem Netzwerkzugriff von außen. Natürlich gibt es Ausnahmen. Zum Beispiel wollen manche Eltern kontrollieren, was ihre Kinder vom eigenen System oder vom Internet sehen dürfen. Normalerweise ist das aber alles. Im Allgemeinen gelten die Benutzer im eigenen Haus als vertrauenswürdig.

Dieses Buch zeigt den Nutzern von Privat- und kleinen Firmenrechnern, wie sie eine Paketfilter-Firewall entwerfen und implementieren. Eine Firewall ist allerdings nur *ein* Schritt in Richtung auf ein sicheres System. Weitere Maßnahmen auf höherer Ebene sind ebenso wichtig.

Computersicherheit sollte man immer auf mehreren Schichten sehen. Keine der Sicherheitsschichten ist für sich alleine ausreichend. Jede höhere Schicht verlässt sich ein Stück weit auf den Schutz durch die tieferen Schichten. Die Entscheidung, welche Dienste Sie auf Ihrem Rechner aktivieren, bildet – zusammen mit der Firewall selbst – die Grundlage für die Sicherheit Ihres Systems. Daher beschäftigt sich dieses Buch mit dem Deaktivieren unnötiger Dienste, mit der gezielten Auswahl derjenigen Dienste, die Sie öffentlich anbieten, sowie mit der Identifikation potenziell gefährlicher privater Dienste, die durch die Firewall geschützt werden müssen.

Das Buch untersucht, welche Formen des Firewall-Schutzes auf einem kleinen System leicht und sinnvoll realisiert werden können. Dabei geht es unter anderem um das Konzept einer Paketfilter-Firewall, um die Implementierung Ihrer persönlichen Firewall, um die Absicherung von Diensten in Bezug auf die Firewall und die Kommunikations-Protokolle, um IP-Masquerading, wodurch Sie die Rechner Ihres Privatnetzes verstecken können, wenn Sie damit auf das Internet zugreifen, und um die Überprüfung, ob die Firewall funktioniert.

Obwohl das nicht das eigentliche Thema des Buches ist, diskutiere ich in Teil III auch Formen der Zugriffskontrolle auf höheren Schichten. Behandelt werden u.a. Access-Control-Lists mit den TCP-Wrappern und Portmap, Serverkonfiguration, Proxies sowie allgemeine Praktiken in der Systemadministration.

Schließlich finden Sie auch noch Abschnitte zur Systemsicherheit und Integritätsprüfung, zur Erkennung von Abtastversuchen und unerlaubten Zugriffsversuchen, noch bevor tatsächlich jemand in das System eindringt. Beschrieben werden Utilities, die Zeichen eines Einbruchs entdecken, und Strategien nach einem erfolgten Einbruch.

Text und Beispiele in diesem Buch beziehen sich auf Red Hat Linux. Die Firewall-Beispiele sind für `ipchains` geschrieben. Weil der Übergang von `ipfwadm` zu `ipchains` aber noch nicht völlig abgeschlossen ist und sich auf heutigen Linux-Systemen beide Varianten finden, sind in Anhang B zusätzlich Beispiele für `ipfw-adm` abgedruckt.

E.2 Was dieses Buch nicht beschreibt

Die Sicherheitsstrategien eines großen Unternehmens stehen dem, was der Benutzer eines kleinen Systems beachten muss, fast diametral gegenüber. Die externe Sicherheit ist nur ein kleiner Bruchteil dessen, was einer größeren Firma an Sicherheitsproblemen begegnet. Man schätzt, dass ungefähr 90% der Vorfälle in Unternehmen vom eigenen Firmennetzwerk ausgehen, nicht vom Internet.

Dieses Buch versucht gar nicht erst eine Beschreibung von interner Systemsicherheit und Sicherheit in Multi-User-LANs, von komplexen Proxy-Konfigurationen, firmenweiten Authentifizierungsregeln und -technologien, VPNs, Verschlüsselungssoftware oder kommerziellen Firewall- und Netzwerklösungen.

E.3 Einbruchsversuche: Dimension eines Problems

Es ist nicht ganz einfach, eine aktuelle Schätzung für die Zahl von Einbruchsversuchen abzugeben. Vielleicht liegt das daran, dass vergebliche Versuche selten bemerkt werden. Viele Sites haben sich auch daran gewöhnt und betrachten sie als alltägliche Vorkommnisse auf dem Internet. Dokumente von CERT enthalten Schätzungen, die von »gleichmäßige Zunahme entsprechend dem Wachstum des Internets« bis zu »exponentielles Wachstum über das Jahr 1998 hinweg« reichen.

Wie hoch die tatsächlichen Zahlen auch sein mögen, es besteht kein Zweifel, dass die weltweiten Einbruchsversuche im Internet zunehmen und dass sie immer raffinierter werden. Portscans z.B. waren früher einfache Tests auf wenige, gut bekannte Sicherheitslücken. Heute werden alle Ports kompletter Domains gescannt. Die neuesten »Hacker-Tools« gehen über Websites, Mailinglisten und Newsgruppen um die Welt. Gruppen von Aficionados koordinieren Scans und Angriffe über den Internet Relay Chat und minimieren dabei das Risiko ihrer Entdeckung. Neu gefundene Sicherheitslücken verbreiten sich in Windeseile über das ganze Internet und werden sofort ausgenutzt. Hersteller und Sicherheitsorganisationen befinden sich in einem ständigen Wettrennen gegen die Black Hats.

E.4 Was hat der Eindringling davon?

Wer sind diese Black Hats überhaupt und was wollen sie? Auf diese Frage gibt es keine einfache Antwort.

Hinweis

Black Hats und Hacker

Um die Frage, wie man einen Eindringling in ein fremdes Computersystem nennt, rankt sich eine kontroverse Debatte. Der Begriff »Hacker« in diesem Kontext wird von den Leuten nicht gerne gesehen, die sich selbst im positiven Sinne als Hacker bezeichnen – halbe (oder ganze) Gurus, die mit den Innereien eines Computers auf Du und Du sind. Der Terminus »Cracker« passt da schon eher auf das zwielichtige Gesindel, vor dem wir uns hier schützen wollen. In letzter Zeit findet auch die Unterscheidung zwischen den »Black Hats« und den »White Hats« immer größere Verbreitung: Die einen knacken Computer aus mehr oder weniger kriminellen Beweggründen, die anderen dringen in die eigenen Systeme ein mit dem Ziel, ihre Sicherheit zu verbessern.

Hinter vielem, was als Einbruchversuch angesehen wird, verbergen sich in Wirklichkeit nur Neugier, Fehler, schlecht geschriebene Software oder schlecht konfigurierte Systeme. Vieles geht von Teenagern und Studenten aus, vieles von kompromittierten Systemen, vor allem an Universitäten. Der Eigentümer solcher

Systeme weiß oft gar nicht, dass sein Computer von einem ungebetenen Gast als Basislager benutzt wird. Es gibt Gruppen von Crackern, die lose zusammenarbeiten und die Einbrüche in fremde Systeme als eine tolle Freizeitbeschäftigung ansehen. Letztlich können Privatanwender auch das Ziel von Spionage aus finanziellen oder politischen Gründen sein.

Und was wollen die Eindringlinge? Manche suchen die Herausforderung, sie wollen »die Nuss knacken«. Andere prahlen gerne. Wieder anderen macht es Freude, einzubrechen und Dinge zu zerstören. Oder man sucht konkrete Vorteile: Eine neue Ausgangsbasis, von der aus man tun kann, was man will –, quasi anonym, denn alle Aktivitäten gehen jetzt ja vom kompromittierten Rechner aus – ist natürlich eine tolle Sache. Von so einem Computer aus kann man auch ungestraft Werbe-E-Mails verschicken. Unangenehmer wird es schon, wenn jemand über Ihren Rechner Raubkopien verbreiten will. Und schließlich gibt es auch Diebe, die Software oder anderes stehlen.

E.5 Was steht für Sie auf dem Spiel?

Wenn in einen Computer eingebrochen wird, erzeugt das beim durchschnittlichen Heimanwender vor allem einmal Unbehagen und Angst. Häufig kommt es zum Datenverlust, wenn ein unfähiger Gast es gerade einmal schafft, die Festplatte zu löschen. Datenverlust bezieht sich auch auf alle Dateien, von denen kein Backup existiert, denn ein solches System muss in jedem Fall neu installiert werden, egal ob weiterer Schaden entstanden ist oder nicht.

Ein anderes Problem besteht darin, dass man den Zugriff auf das Internet verliert. Der Internet-Provider wird den Zugang des Betroffenen typischerweise sperren, bis das Problem nachweislich behoben ist. Der Eigentümer muss sein Problem aber erst einmal verstehen und lernen, wie er seinen Rechner absichert. Das kostet Zeit. Für eine kleine Firma ist das nicht nur lästig, sondern es kann auch einen Umsatzverlust bedeuten.

Sie werden nicht nur beim eigenen Provider verdächtig, sondern verlieren auch Ihr Image bei allen, die durch den Eindringling auf Ihrem System belästigt worden sind. Wenn Ihr Internet-Provider nicht von Ihrer Unschuld überzeugt ist, sperrt er Ihren Anschluss möglicherweise endgültig. Wenn über Ihren Computer Raubkopien verteilt werden oder Ihr »Gast« sich mit den falschen Leuten anlegt, können durchaus auch rechtliche Schritte und soziale Schwierigkeiten auf Sie zukommen.

Eine weitere Gefahr ist, dass persönliche und geschützte Informationen gestohlen und verbreitet werden.

E.6 Firewalls und Black Hats in einer idealen Welt

Vom Konzept her könnten viele – wenn nicht sogar die meisten – Einbruchsversuche von den Internet- oder Gateway-Providern schon an der Quelle gestoppt werden. Einheitliche Filterregeln, in allen Routern und Gateways eingesetzt, würden die meisten dieser Versuche sofort zunichte machen. Leider funktioniert das nur in einer idealen Welt. Man müsste nicht nur alle Provider auf der ganzen Welt davon überzeugen, wie wichtig und verantwortungsvoll sie dabei sind, sondern man bräuchte auch Router, die mit der zusätzlichen Arbeit des Paketfilterns (in riesigem Umfang!) zurecht kämen. So gut ist unsere Hardware leider noch nicht.

Trotzdem können Heimanwender und kleine Firmen solche Filter auf ihren Rechnern leicht einsetzen, und zwar ohne spürbare Geschwindigkeitseinbußen. Damit erhalten sie nicht nur ein sichereres System, sondern sie schützen auch andere Leute vor ihren Fehlern.



Teil I

Vorüberlegungen

Kapitel 1 Grundlegende Konzepte für Paketfilter-Firewalls

Kapitel 1

Grundlegende Konzepte für Paketfilter-Firewalls



- 1.1 Das OSI-Referenzmodell
- 1.2 Ports: Tore zu den Programmen auf Ihrem Computer
- 1.3 Pakete: Botschaften im IP-Netz
- 1.4 Zusammenfassung

Eine kleine Site realisiert ihren Internetzugang z.B. über ein Kabelmodem, einen DSL-Anschluss, ISDN, oder oft auch über eine PPP-Verbindung zu einem Einwahlzugang. Den zentralen Angriffspunkt stellt dabei der Rechner dar, der direkt an das Internet angeschlossen ist. Ob Sie jetzt einen einzigen Computer oder ein kleines Local Area Network (LAN) anbinden, im Mittelpunkt der Überlegungen steht immer der Computer mit direktem Netzzugang. Dieser wird Ihre Firewall sein.

Der Begriff *Firewall* hat unterschiedliche Bedeutungen, je nach Implementierung und Einsatzzweck. Hier, ganz am Anfang des Buches, bedeutet »Firewall« zunächst einfach einmal die Maschine, die ans Internet angeschlossen ist und auf der Sie Ihre Sicherheitsregeln realisieren. Das externe Netzwerkinterface dieses Computers (z.B. die ISDN-Karte oder das DSL-Modem) ist der Verbindungspunkt, das *Gateway*, ins Internet. Das Ziel einer Firewall besteht darin, alles auf Ihrer Seite dieses Gateways vor allem zu schützen, was auf der anderen Seite liegt.

Ein Setup mit einer einfachen Firewall nennt man manchmal auch *Bastion-Firewall*, weil die Firewall die Hauptverteidigungseinrichtung gegen den Rest der Welt ist. Alle Sicherheitsmaßnahmen beziehen sich hierauf. Alles nur irgend Mögliche sollte daher getan werden, um dieses System zu schützen: Es ist ihre einzige »Bastion«.

Hinter der Firewall liegen ein oder mehrere weitere Computer. Die Firewall dient den anderen Maschinen im LAN möglicherweise einfach als Verbindung ins Internet. Dahinter laufen vielleicht private Dienste, die geschützt werden sollen, z.B. ein Netzwerkdrucker oder gemeinsam genutzte Dateisysteme. Eventuell wollen Sie von allen Ihren Computern Zugriff auf das WWW haben. Oder einer der Rechner erledigt Ihre private Buchführung; Sie wollen zwar von dort aus ins Internet, aber in der umgekehrten Richtung soll alles gesperrt sein. Später einmal bieten Sie vielleicht öffentliche Dienste an, z.B. Ihre private Web-Site. Ihre Sicherheitsstrategien hängen davon ab, wie Ihre Computer eingerichtet sind und was Sie damit tun.

Die Firewall soll die von Ihnen definierten Strategien umsetzen. Diese sind ein Abbild Ihrer Entscheidungen,

- welche Internet-Dienste von Ihren Computern aus genutzt werden dürfen,
- welche Dienste Sie der ganzen Welt anbieten,
- welche Dienste Sie ausgewählten anderen Nutzern oder Computern anbieten,
- und welche Dienste und Programme Sie lokal und privat laufen lassen.

Wenn ich hier von »Sicherheitsstrategien« spreche, ist damit Zugriffskontrolle gemeint, also der kontrollierte Zugriff auf private oder sonstwie geschützte Dienste, Programme und Dateien auf Ihren Computern.

Heimannwender und kleine Firmen müssen sich nicht mit allen Sicherheitsproblemen beschäftigen, die im professionellen Umfeld auftreten. Die Grundgedanken und ersten Schritte sind zwar gleich, es gibt aber nicht so viele Faktoren, die man in Betracht ziehen muss. Das Hauptaugenmerk liegt auf dem Schutz der eigenen Site vor unerwünschten Zugriffen aus dem Internet. (Eine Firma würde innere Sicherheit stärker betonen, was für Ihr Zuhause weniger wichtig ist.) Eine Paketfilter-Firewall ist ein beliebter Ansatz für die Realisierung von Netzwerksicherheit und Zugriffskontrolle gegenüber der Außenwelt. Sie ist aber nur ein Teil eines größeren Ganzen.

Bevor wir uns mit den Details des Firewall-Designs beschäftigen, führt dieses Kapitel zunächst ein paar grundlegende Konzepte und Mechanismen ein, auf denen Paketfilter basieren: Was ist Kommunikation in Netzwerken überhaupt? Wie unterscheidet man die einzelnen Netzwerk-Dienste? Was ist ein »Paket«? Was für Botschaften schicken sich die Computer auf einem Netzwerk gegenseitig zu?

1.1 Das OSI-Referenzmodell

Bei der Vorstellung eines begrifflichen Rahmenwerks für dieses Kapitel und den Rest des Buches verwende ich einzelne Fachbegriffe, noch bevor sie in den folgenden Abschnitten genauer definiert werden. Für Profis aus Informatik und Netzwerkdesign handelt es sich ohnehin um Altbekanntes, technisch weniger vorbelastete Leute hingegen werden hier und da wohl noch ein paar neue Konzepte finden. Wenn für Sie noch alles neu ist, dann machen Sie sich auch keine Sorgen – im Moment stelle ich nur ein paar Konzepte vor, die den später folgenden Definitionen vielleicht mehr Sinn verleihen.

Wenn Ihre akademische Ausbildung auch Netzwerke beinhaltete, kennen Sie sicherlich noch das OSI-Referenzmodell. OSI steht für »Open Systems Interconnection«, also Verbindung offener Systeme miteinander; dieses Modell wurde Ende der Siebziger, Anfang der Achtziger Jahre entwickelt, als man für Standards im Bereich der Vernetzung von Computern einen theoretischen Rahmen brauchte. Das OSI-Modell ist sehr formell und akademisch und auch gut durchdacht. Lehrbücher und Professoren unterhalten sich über Netzwerke vor dem konzeptuellen Hintergrund dieses Modells.

Damals wurden Computernetzwerke allmählich große Mode, und in den sieben Jahren der Entwicklung des OSI-Modells blieb die Zeit nicht stehen. Als TCP/IP der *de facto*-Standard für die Kommunikation zwischen Unix-Maschinen wurde, entstand ein zweites, informelleres Modell: das TCP/IP-Referenzmodell. Dieses ist weniger ein universitäres Ideal als vielmehr ein Abbild dessen, worauf sich Hersteller und Entwickler in Bezug auf die Kommunikation über das Internet letztlich einigten. Weil dieses Modell TCP/IP aus der praktischen, bodenständigen Perspektive eines Entwicklers sieht, ist es viel einfacher als das OSI-Modell.

Wo OSI noch zwischen sieben Schichten unterscheidet, spricht das TCP/IP-Modell nur noch von vier.

Dieses Buch beruht auf dem TCP/IP-Referenzmodell. Wie die meisten Leute aus der Informatik benutze ich meistens das OSI-Vokabular, gebrauche es konzeptuell aber im Rahmen des TCP/IP-Modells.

Kommunikation im Netzwerk stellt man sich in mehreren Schichten vor, wobei der eigentliche Datenaustausch zwischen vertikal angrenzenden Schichten auf dem gleichen Computer sowie zwischen horizontal einander entsprechenden Schichten auf miteinander kommunizierenden Computern erfolgt. Ganz oben, in der Anwendungsschicht, steht Ihr Programm (z.B. Ihr Web-Browser), das mit einem anderen Programm auf einem anderen Computer spricht (z.B. mit einem Webserver).

Wenn Ihr Browser eine Webseite vom Server anfordern will, muss er Bibliotheks- und Systemaufrufe durchführen. Diese nehmen die Daten vom Browser und verpacken sie in eine Nachricht, die sich für den Transport zwischen zwei Programmen eignet. Die entstehende Nachricht ist ein TCP-Segment oder ein UDP-Datagramm aus der *Transportschicht*. Wenn die Anwendungsschicht so eine Nachricht produzieren möchte, ruft sie die Transportschicht auf, damit sie das für sie erledigt. Die Nachrichten der Transportschicht werden zwischen dem Client (Browser) und dem Server (Webserver) ausgetauscht. Die Transportschicht weiß, wie sie eine Nachricht zwischen zwei Programmen auf zwei verschiedenen Computern im Netzwerk verschickt. Die Transportschicht heißt sowohl bei OSI als auch im TCP/IP-Modell so, wird im OSI-Modell aber noch funktionell in mehrere Unterschichten getrennt.

Um diese Nachrichten aus der Transportschicht zwischen zwei Programmen zu übertragen, müssen sie zwischen zwei Computern gesendet werden. Dazu ruft die Transportschicht Funktionen des Betriebssystems auf, welche die TCP- oder UDP-Nachricht nehmen und in ein Internet-Datagramm verpacken, das an den anderen Computer geschickt werden kann. Diese Datagramme sind die IP-Pakete. IP-Pakete werden zwischen zwei Computern über das Internet übertragen. Die Internet-Schicht weiß, wie sie mit dem Computer auf der anderen Seite des Netzwerks reden kann. Das TCP/IP-Modell nennt diese Schicht die *Internet-Schicht*. In der Praxis benutzt man dafür eher das OSI-Vokabular und spricht damit von der *Netzwerkschicht* bzw. *Vermittlungsschicht*. Diese Begriffe meinen aber ein und dasselbe Konzept.

Unterhalb der Netzwerk- oder Vermittlungsschicht liegt die *Subnetz-Schicht*. Das Paket wird hier nochmals verpackt, und zwar in einen Ethernet-Header. In der Subnetz-Schicht liegt die Nachricht jetzt als *Ethernet-Frame* vor. Aus der Perspektive von TCP/IP ist die Subnetz-Schicht einfach alles, was nötig ist, um ein IP-Paket an den nächsten Computer zu schicken. Dazu gehören alle Details, die Adressierung und Auslieferung betreffen, das Routing von einem Computer zum nächsten, bis der Empfänger gefunden ist. Diese Schicht ist auch dafür zuständig, den Netzwerkframe unterwegs zwischen unterschiedlichen Netzwerk-Arten zu

übersetzen. Die meisten Netzwerke der heutigen Zeit sind Ethernet-Netzwerke, aber unterwegs gibt es ATM-Netze, FDDI, Token-Ring etc., was auch immer bei der Datenübertragung zum Einsatz kommt. Hierher gehören auch die Hardware, die Kabel zwischen den Computern, sowie die Signale, die Spannungsänderungen, die die einzelnen Bits eines Frames sowie die Kontrollinformationen zur Abgrenzung eines einzelnen Bytes repräsentieren.

In der Zusammenschau (siehe Bild 1.1) spiegelt also die Anwendungsschicht den Datenaustausch zwischen zwei Programmen wider. Die Transportschicht beschreibt, wie diese Daten zwischen zwei Programmen ausgeliefert werden. In dieser Schicht werden Programme durch Zahlen identifiziert, die man *Ports* nennt. Die Netzwerk- oder Vermittlungsschicht ist dafür zuständig, die Daten zwischen den zwei Computern an jedem Ende der Verbindung zu übertragen. Computer bzw. ihre einzelnen Netzwerkkarten werden auch hier durch Zahlen identifiziert, und zwar durch die *IP-Adressen*. Die Subnetz-Schicht überträgt die Daten auf dem Weg von einem Computer zum nächsten. Auf einem Ethernet sind die Netzwerkkarten wiederum durch Zahlen identifiziert, durch die *Ethernet-* bzw. *MAC-Adressen*.

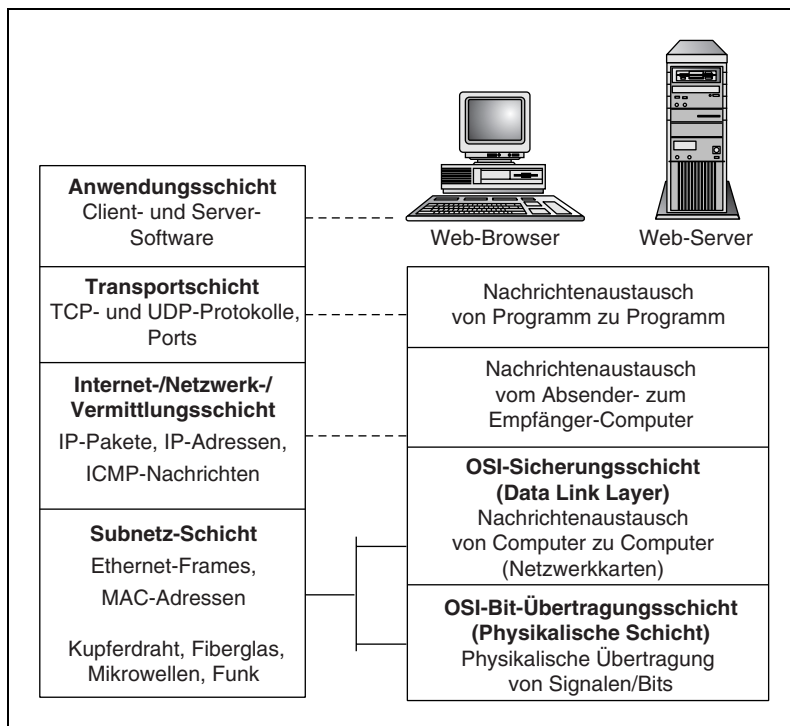


Bild 1.1: Das TCP/IP-Referenzmodell

1.2 Ports: Tore zu den Programmen auf Ihrem Computer

Netzwerkbasierte Dienste sind Programme auf einem Computer, auf die andere Computer über das Netzwerk zugreifen können. *Ports* identifizieren diese Programme sowie die einzelnen Sessions und Verbindungen. Ports sind numerische Namen für die unterschiedlichen Netzwerkdienste. Sie dienen gleichzeitig als Kennzeichner für die beiden Endpunkte jeder Verbindung. Die Port-Zahlen liegen im Bereich von 0 bis 65535.

Server-Programme oder *Dämonen* warten auf dem ihnen zugewiesenen Port auf eingehende Verbindungen. Von alters her liegen alle wichtigen Netzwerkdienste auf bekannten Ports im unteren Bereich von 1 bis 1023. Die Internet Assigned Numbers Authority (IANA) koordiniert diese Zuordnung zwischen Portnummern und Diensten; es handelt sich dabei um einen weltweit anerkannten Standard.

Die Ports im unteren Bereich nennt man auch *privilegierte Ports*. Sie gehören zu Programmen, die mit elevierten Privilegien laufen, d.h. als *superuser* oder *root*. Damit will man ein bisschen sicherer gehen, dass der Client tatsächlich den gewünschten und angebotenen Dienst erreicht. Das ist zumindest die Absicht – eine Garantie ist es keineswegs. Man kann sich niemals absolut sicher sein, dass der Computer bzw. der Dienst am anderen Ende wirklich der ist, als der er sich ausgibt.

»Angeboten« ist jeder Dienst, der über das Internet auf dem korrekten Port erreichbar ist. Wenn Ihre Maschine einen bestimmten Dienst nicht anbietet und jemand trotzdem versucht, eine Verbindung zum entsprechenden Port herzustellen, passiert gar nichts. Derjenige klopft sozusagen an die Tür, aber es ist niemand zu Hause. Zum Beispiel sind Web-Server dem Port 80 zugeordnet. Wenn auf Ihrem Computer kein Web-Server läuft und jemand eine Verbindung zu Ihrem Port 80 aufbauen will, erhält er eine Fehlermeldung von Ihnen, dass der betreffende Dienst nicht angeboten wird.

Die oberen Ports von 1024 bis 65535 heißen *unprivilegierte Ports*. Sie werden zweifach verwendet. Zum einen, und das ist der häufigere Fall, wird der Client-Seite einer Verbindung dynamisch ein Port aus diesem Bereich zugewiesen. Jede Verbindung lässt sich mit einer Kombination aus den Client- und Server-Portnummern plus dem verwendeten Transportprotokoll und den jeweiligen IP-Adressen eindeutig identifizieren.

Zum anderen hat die IANA viele Ports zwischen 1024 und 49151 registriert. Diese Ports können zwar weiterhin als ganz normale unprivilegierte Ports verwendet werden; darüber hinaus sind ihnen aber auch bestimmte Dienste wie z.B. SOCKS oder die X-Window-Server zugeordnet. Ursprünglich war das so gedacht, dass Dienste auf oberen Ports nicht mit *root*-Rechten laufen. Darauf sollte man sich aber nicht immer verlassen.

Hinweis

Die Zuordnung zwischen Dienst-Namen und -Ports

Linux-Distributionen enthalten eine Liste der wichtigen Ports. Diese befindet sich in der Datei `/etc/services`.

Jede Zeile dieser Datei enthält den symbolischen Namen eines Dienstes, die entsprechende Portnummer, das verwendete Protokoll (TCP oder UDP) sowie etwa gebräuchliche alternative Namen. Die folgende Tabelle zeigt ein paar häufige Zuordnungen dieser Art:

Symbolischer Name	Port und Protokoll	Alternativer Name
ftp	21/tcp	
telnet	23/tcp	
smtp	25/tcp	mail
whois	43/tcp	nicname
domain	53/tcp	nameserver
domain	53/udp	nameserver
finger	79/tcp	
pop-3	110/tcp	
nntp	119/tcp	readnews
www	80/tcp	http
auth	113/tcp	ident
ntp	123/udp	
https	443/tcp	

Tab. 1.1: Auszug aus `/etc/services`

Achtung: Die symbolischen Namen, die den einzelnen Ports zugeordnet sind, können sich bei zwei unterschiedlichen Unices, bei zwei Distributionen, ja sogar bei zwei Versionen ein und derselben Distribution unterscheiden. Die Portnummern hingegen bleiben immer gleich.

Beachten Sie weiterhin, dass zu einem Port immer auch ein bestimmtes Protokoll gehört. Die IANA hat versucht, TCP und UDP immer die gleiche Portnummer zuzuordnen, egal ob ein Dienst beide Protokolle benutzt oder nicht. Die meisten Dienste verwenden jedoch nur ein Protokoll. `domain` ist eine Ausnahme, es benutzt sowohl UDP als auch (selten) TCP.

1.3 Pakete: Botschaften im IP-Netz

Der Begriff *Paket* bezieht sich auf eine Nachricht im *Internet Protocol* (IP). Der IP-Standard definiert den Aufbau einer Nachricht, die sich zwei Computer über das Netz zusenden. Ein Paket ist eine einzelne Nachricht oder Informationseinheit, die über dieses Netzwerk läuft. Es enthält einen Informationskopf oder Header sowie den eigentlichen Nachrichtentext oder Body, also die übertragenen Daten. Der Body eines IP-Paketes – die enthaltenen Daten – ist entweder eine komplette Nachricht eines übergeordneten Protokolls oder aber ein Teil davon, ein *Fragment*.

Der IP-Firewall-Mechanismus (IPFW) von Linux unterstützt drei IP-Nachrichtentypen: ICMP, UDP und TCP:

Ein ICMP-Paket (Internet Control Message Protocol) gehört in die Netzwerkschicht und stellt eine Kontroll- oder Statusnachricht dar. Es enthält Informationen über die Kommunikation zwischen zwei Computern.

Ein UDP-Paket (User Datagram Protocol) enthält Daten der Transportschicht, die zwischen zwei Programmen transportiert werden. Dabei ist nicht garantiert, ob das Paket tatsächlich ankommt und ob die Reihenfolge der Pakete während der Übertragung beibehalten wird, d.h. ob das zuerst gesendete Paket wirklich auch als erstes den Empfänger erreicht. Ein UDP-Paket ist vergleichbar mit einer Postkarte an ein anderes Programm.

Ein TCP-Paket (Transmission Control Protocol) überträgt ebenfalls Daten der Transportschicht. Der Header des Paketes enthält allerdings zusätzliche Informationen, die die Aufrechterhaltung einer kontinuierlichen und verlässlichen Verbindung ermöglichen. In dem bildlichen Vergleich mit der Post entspricht ein TCP-Paket einem Ausschnitt aus einem Telefongespräch zwischen zwei Programmen.

Die meisten Internet-Dienste verwenden TCP, nicht UDP. Mit anderen Worten: Die meisten Dienste im Internet basieren auf dem Konzept einer kontinuierlichen Verbindung zwischen zwei Programmen, einem Client und einem Server, wobei Kommunikation in beide Richtungen möglich ist.

Alle IP-Pakete enthalten im Header die IP-Adresse des Absenders und des Empfängers sowie die Art des Protokolls, zu dem das Paket gehört (ICMP, UDP oder TCP). Darüber hinaus speichert der Header je nach Protokoll etwas unterschiedliche Felder. ICMP-Pakete haben ein Typ-Feld, das die Art der Kontroll- oder Statusnachricht angibt, sowie ein weiteres Code-Feld, das die Nachricht genauer bestimmt. UDP- und TCP-Pakete enthalten die Portnummern von Absender und Empfänger. In einem TCP-Paket sind noch weitere Informationen über den Zustand der Verbindung sowie eindeutige Bezeichner für jedes Paket gespeichert. (Der Header umfasst noch mehr Felder. Die anderen Felder sind aber entweder unsichtbar oder für einen Paketfilter auf IP-Ebene wenig brauchbar.) (Definitionen der übrigen Felder im Header finden Sie in jedem beliebigen Buch über TCP/IP, z.B. in »TCP/IP Clearly Explained« von Pete Loshin, verlegt bei Academic Press.)

1.3.1 IP-Nachrichtentypen: ICMP

Ein ICMP-Paket ist eine IP-Kontroll- und Status-Nachricht der Netzwerkschicht. Der Header enthält die IP-Adressen von Absender und Empfänger, die Kennung für das ICMP-Protokoll, sowie den ICMP-Nachrichtentyp. Dieser zeigt an, ob es sich bei dem Paket um ein Kommando handelt, um eine Antwort auf ein Kommando, um Statusinformationen oder um einen Fehler. Die einzelnen ICMP-Nachrichtentypen werden in Kapitel 3 genauer beschrieben.

ICMP-Nachrichten kennen keine Portnummern für Absender oder Empfänger. Es handelt sich bei ihnen ja nicht um Nachrichten zwischen zwei Programmen, sondern zwischen zwei Computern.

Ein Beispiel für eine ICMP-Nachricht wäre eine Fehlermeldung: Wenn Sie auf einen Webserver zugreifen wollen, aber auf dem betreffenden Computer läuft gar keiner, schickt Ihnen der angesprochene Computer einen Hinweis, dass dieser Dienst nicht existiert. Bild 1.2 zeigt die wesentlichen Felder aus dem Header.

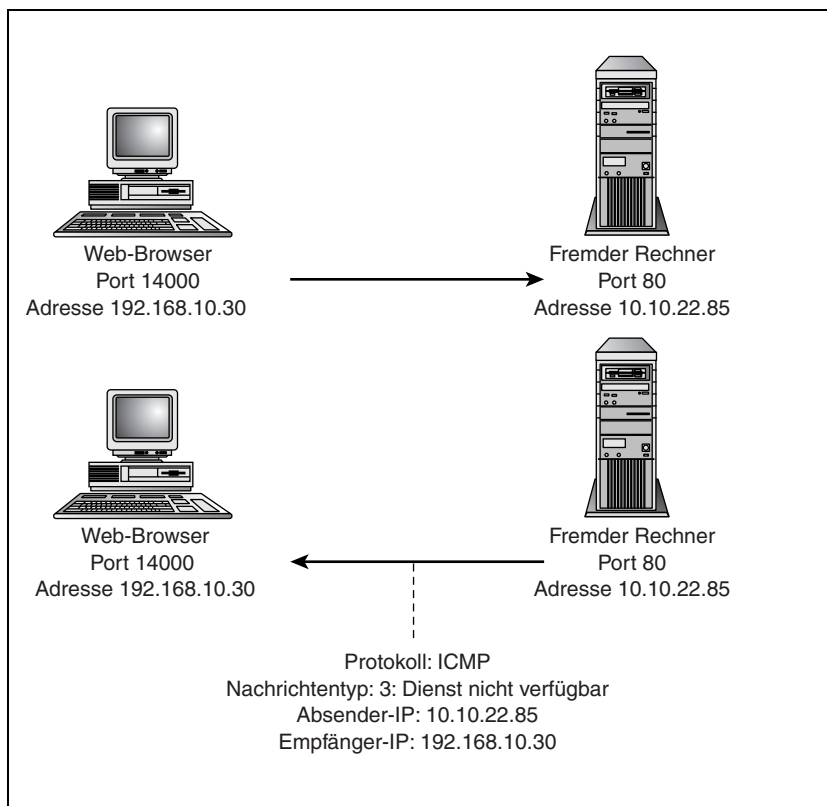


Bild 1.2: Eine ICMP-Fehlermeldung kommt an

1.3.2 IP-Nachrichtentypen: UDP

UDP ist *stateless* (»zustandsfrei«, kommt also ohne Verbindungen aus) und unverlässlich. Ein Programm schickt eine UDP-Nachricht ab, die ankommen kann, aber nicht muss. Vielleicht antwortet jemand darauf. Es gibt keine Empfangsbestätigung und es existiert auch keine Flusskontrolle. Wenn ein UDP-Datagramm unterwegs nicht verarbeitet werden kann, wird es stillschweigend weggeworfen.

Der Header eines UDP-Paketes enthält die IP-Adressen von Absender und Empfänger, den UDP-Protokolltyp, sowie die Portnummern von Absender und Empfänger.

Hinweis

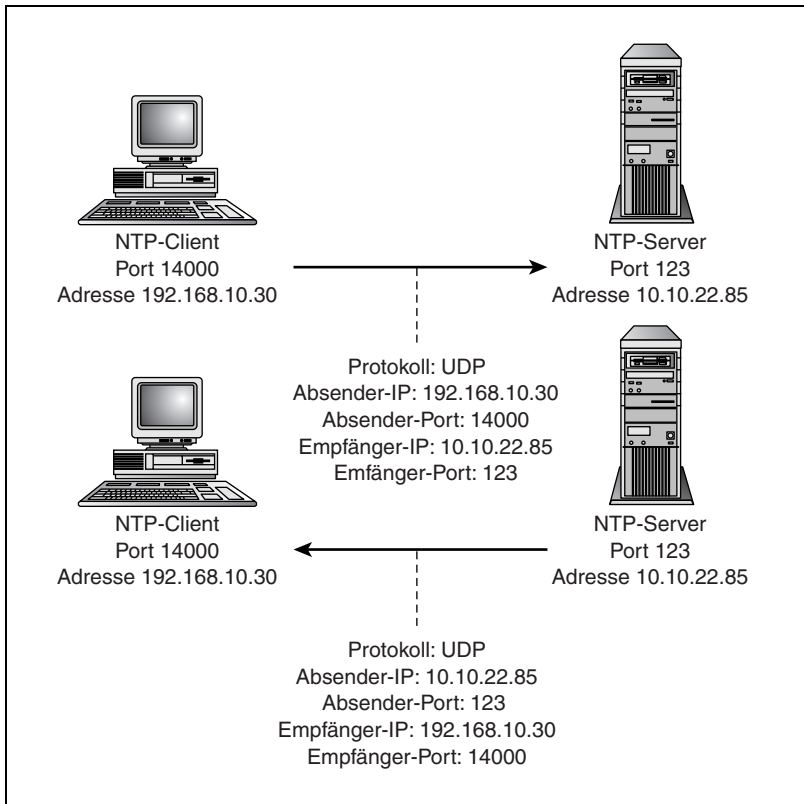
Ist UDP unzuverlässig?

UDP wird als »unverlässlicher Datagramm-Dienst« bezeichnet. »Unverlässlich« heißt aber nicht »unzuverlässig«; damit ist keinerlei negative Aussage über die Verwendbarkeit von UDP verbunden. »Unverlässlich« heißt in diesem Fall einfach nur, dass die Zustellung nicht garantiert wird. Da nicht erst – wie bei TCP – aufwändig eine Verbindung hergestellt werden muss, um die Zuverlässigkeit der Übertragung zu sichern, ist die Datenübertragung über UDP um ein Mehrfaches schneller als die über TCP. UDP ist daher hervorragend geeignet für einfache Arten von Kommunikation, wenn man nur schnell eine Frage stellen und die Antwort zurückschicken will.

Beispielsweise könnten Sie Ihren Computer so einrichten, dass er regelmäßig auf einen NTP-Server (Network Time Protocol) zugreift. Der NTP-Server läuft auf Port 123. Er antwortet auf Anfragen mit der aktuellen Uhrzeit. Ihre Systemuhr wird dann nach der Uhr des (genaueren) NTP-Servers gestellt. Wenn die Anfrage keinen Erfolg hat, z.B. weil auf dem abgefragten Computer gar kein NTP-Server läuft oder der Computer selbst gerade ausgeschaltet ist, wird stattdessen die ICMP-Fehlermeldung »Service Or Host Not Available« (Dienst oder Computer nicht verfügbar) zurückgegeben.

Wie Sie in Bild 1.3 sehen, formuliert das Client-Programm auf Ihrem Computer die Anfrage. Dieser Anfrage wird automatisch ein unprivilegiertes Port zugeordnet. Das losgeschickte Paket enthält als Absender-Port diesen unprivilegierten Port und als Empfänger-Port den Standard-Port für NTP, nämlich 123.

Der NTP-Server antwortet mit der aktuellen Uhrzeit in einem UDP-Paket. Ihr Computer erhält also eine UDP-Antwort von der IP-Adresse des Servers, Port 123, an Ihre IP-Adresse und die gerade vergebene unprivilegierte Portnummer.

**Bild 1.3:** UDP-Anfrage und -Antwort

1.3.3 IP-Nachrichtentypen: TCP

Die allermeisten Netzwerkdienste arbeiten mit TCP. Nachrichten in beide Richtungen werden als Teil einer kontinuierlichen Verbindung zwischen zwei Programmen zuverlässig ausgeliefert. Fehler treten nicht auf, Daten gehen nicht verloren (und werden auch nicht vervielfacht), und alle Pakete kommen in der korrekten Reihenfolge an. Auf jedes TCP-Segment folgt eine Empfangsbestätigung. Jedes Segment wird durch eine eindeutige Sequenznummer gekennzeichnet. Spezielle Flags beschreiben den Zustand der Verbindung.

Wenn ein IP-verkapseltes TCP-Segment größer ist als die Maximum Transmission Unit (MTU) des benutzten Netzwerks, wird es in Fragmente aufgeteilt. (Im häufigen Falle eines Ethernets liegt die MTU bei 1500 Byte pro Ethernet-Frame.) Die Fragmente werden als Teil eines bestimmten Segments gekennzeichnet und separat abgeschickt. Der Empfänger-Computer setzt sie wieder zum ursprünglichen TCP-Segment zusammen.

Beim Aufbau einer Verbindung zu einem Server wählt die Client-Seite einen unprivilegierten Port aus. Zusammen mit der IP-Adresse des Clients und dem Protokolltyp (TCP) definiert dieser das Client-*Socket*. Umgekehrt besteht das Server-Socket aus der Host-IP, der Portnummer des Servers und dem Protokolltyp. Die Verbindung zwischen Client und Server ist durch Angabe der beiden Sockets eindeutig definiert.

Allgemein: Jede beliebige Verbindung zwischen einem Client und einem Server, die ja – z.B. im Falle eines Web-Servers – möglicherweise nur eine von mehreren gleichzeitig bestehenden Verbindungen darstellt, wird durch IP-Adresse und Portnummer des Clients, IP-Adresse und Portnummer des Servers sowie das verwendete Transportprotokoll eindeutig bestimmt.

Ein TCP-Header enthält die IP-Adressen von Absender und Empfänger, den TCP-Nachrichtentyp, die Ports von Absender und Empfänger, Sequenz- und Bestätigungsnummern sowie Kontrollflags, durch die Aufbau und Aufrechterhaltung einer in beiden Richtungen zuverlässig nutzbaren Verbindung ermöglicht werden.

Eine typische TCP-Verbindung: Besuch bei einer Webseite

Ein typisches Beispiel für eine TCP-Verbindung: Ihr Netscape-Browser baut eine Verbindung zu einer Website (also zu einem Web-Server) auf. Dieser Abschnitt demonstriert die Aspekte des Verbindungsaufbaus und der laufenden Kommunikation, die später beim Filtern von IP-Paketen von Bedeutung sein werden.

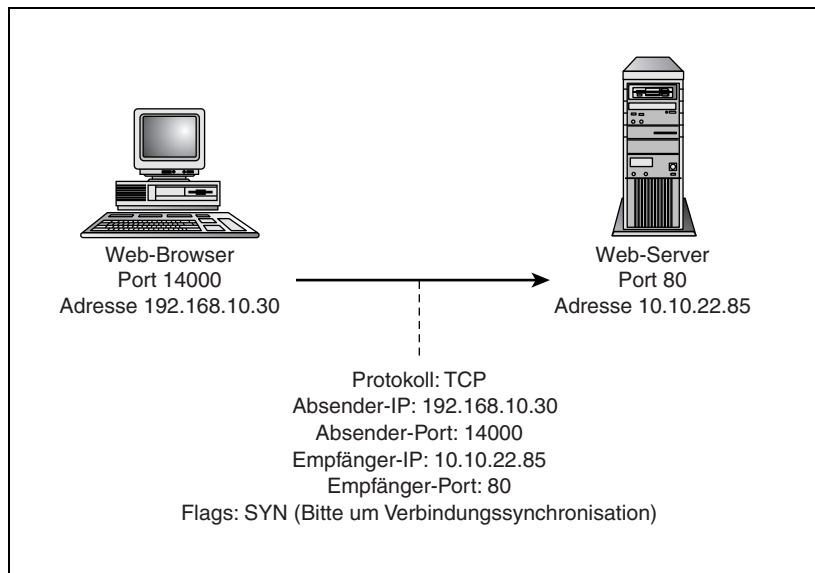


Bild 1.4: TCP: Client erbittet Verbindungsaufbau

Was passiert? Bild 1.4 zeigt einen Webserver, der auf dem TCP-Port 80 auf eine Verbindungsanfrage wartet. Sie klicken in Netscape auf eine URL. Diese URL wird u.a. in einen Hostnamen zerlegt; dieser wird in eine IP-Adresse übersetzt; und Ihrem Browser wird vom Betriebssystem ein unprivilegierten Port (z.B. 14000) für die Verbindung zugewiesen. Er schreibt eine HTTP-Nachricht für den Web-Server. Diese wird in eine TCP-Nachricht verpackt, mit einem IP-Header versehen und auf den Weg geschickt. Die für unsere Zwecke wichtigen Felder im Header sind in Bild 1.4 gezeigt.

Der Header enthält weitere Informationen, die auf der Ebene des Paketfilters allerdings nicht sichtbar sind. Trotzdem hilft die Beschreibung der Sequenznummern, die mit SYN und ACK verwendet werden, den dreiteiligen Handshake zu verstehen.

Wenn der Client das erste Paket des Verbindungsaufbaus abschickt, ist darin, zusammen mit dem SYN-Flag, auch eine Sequenznummer enthalten. Der Client wünscht eine Verbindung mit dem Server. Er übergibt eine Sequenznummer, auf deren Basis er alle nachfolgenden Nachrichten nummerieren wird.

Das Paket kommt auf der Server-Maschine an und gelangt zum Port 80. Der Server wartet auf diesem Port auf Anfragen und wird folglich über die von Absender-IP und -Port erhaltene Aufforderung zum Verbindungsaufbau (SYN-Flag gesetzt) informiert. Auf Server-Seite wird ein neues Socket reserviert und mit dem Client-Socket assoziiert. Der Web-Server schickt nun eine Bestätigung (ACK) der SYN-Nachricht zurück, dazu eine eigene Synchronisationsaufforderung (SYN) – siehe Bild 1.5. Die Verbindung ist jetzt *halb-offen*.

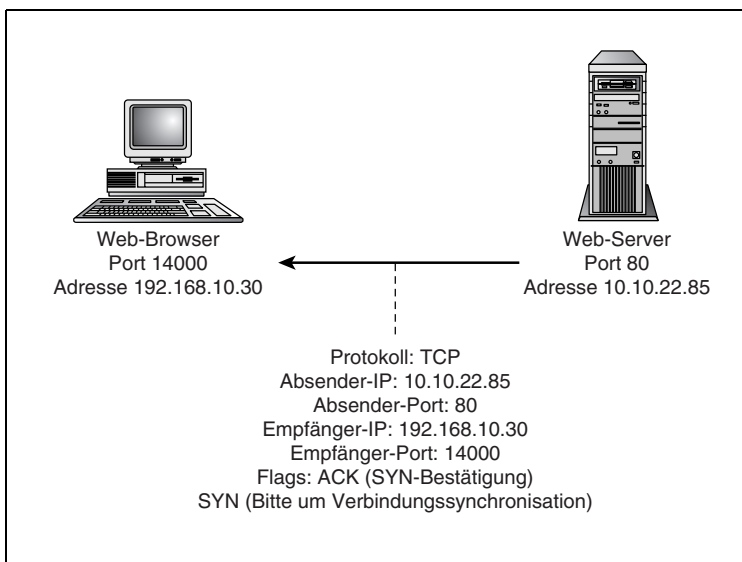


Bild 1.5: TCP: Bestätigung der Bitte um Verbindungsaufbau

Der SYN-ACK-Header enthält zwei Felder, die für den Paketfilter nicht sichtbar sind:

Mit dem ACK-Flag wird die vom Client erhaltene Sequenznummer plus eins zurückgegeben. Der Client hatte die Nachricht mit seiner Sequenznummer gekennzeichnet. Der Server bestätigt das mit der Sequenznummer plus eins und sagt damit, dass er die Nachricht erhalten hat und dass er als Nächstes eine Nachricht mit der Sequenznummer plus eins erwartet. Der Client kann sein Original-Paket mit der SYN-Nachricht jetzt wegwerfen, weil der Server dessen Erhalt bestätigt hat.

Gleichzeitig setzt der Server auch das SYN-Flag. Wie bei der ersten Nachricht des Clients, wird auch hierbei eine Sequenznummer für die Synchronisation übergeben. Der Server gibt seine eigene Sequenznummer für seine Seite der Verbindung an.

Dieses erste Paket ist das einzige, in dem der Server das SYN-Flag setzt. Dieses und auch alle folgenden Pakete haben zudem ein gesetztes ACK-Flag. Dass in *allen* Server-Nachrichten ein ACK-Flag gesetzt ist, in der *ersten* Nachricht des Clients aber nicht, wird später ein ganz wichtiges Unterscheidungskriterium sein, wenn es darum geht, welche Informationen wir beim Aufbau einer Firewall benutzen können.

Ihr Computer empfängt das Paket vom Server und bestätigt es seinerseits, wodurch die Verbindung hergestellt ist (siehe Bild 1.6). Ab diesem Punkt setzen Client und Server in jeder Nachricht das ACK-Flag. Keiner von beiden wird das SYN-Flag während dieser Verbindung noch einmal verwenden.

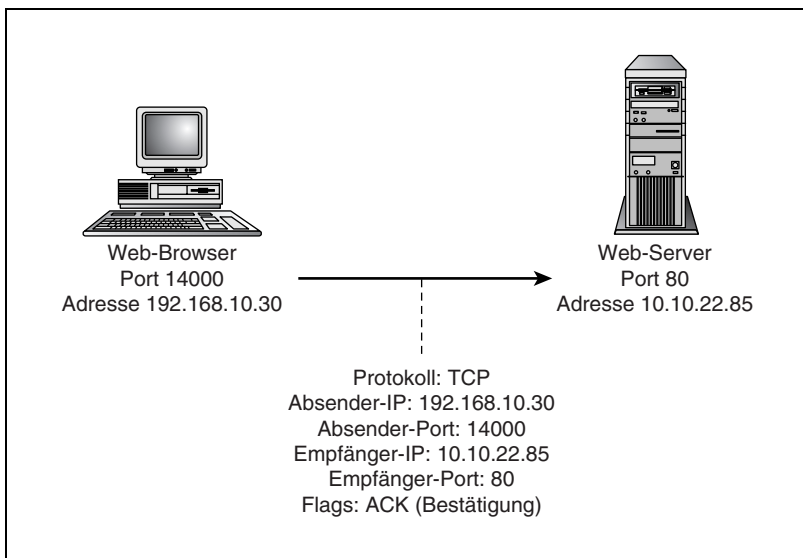


Bild 1.6: TCP: Verbindung wurde hergestellt

Bei jeder Bestätigung (jedem ACK) erhöhen Client und Server die Sequenznummer des jeweiligen Partners um die Zahl der seit dem letzten ACK in korrekter Folge erhaltenen Pakete. So bestätigen sie den Empfang der Pakete und teilen dem Partner mit, welche Nachrichten-Nummer sie als Nächstes erwarten.

Während Ihr Browser den HTML-Code empfängt, erhält Ihre Maschine Datenpakete vom Web-Server, mit Headern wie in Bild 1.7.

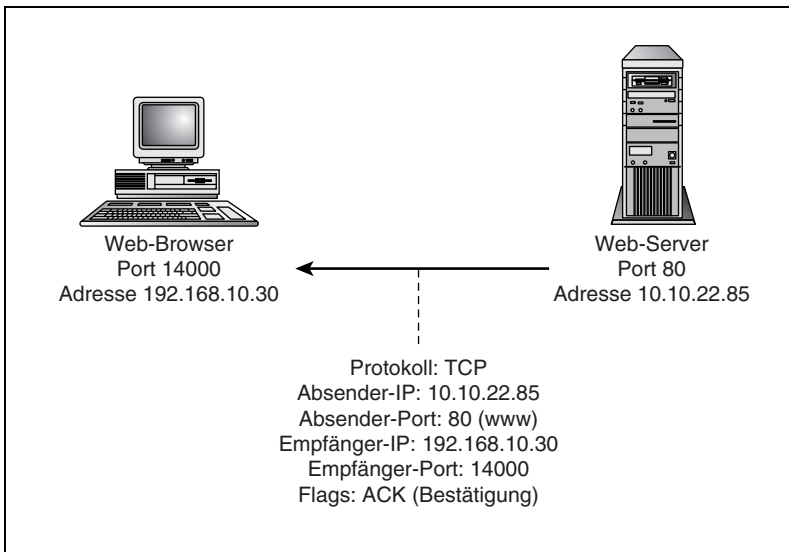


Bild 1.7: TCP: laufende Übertragung vom Server zum Client

Am Ende einer Verbindung werden weitere Nachrichten mit speziellen Flags für den Verbindungsabbau verschickt. Diese Flags stehen einem Paketfilter aber nicht zur Verfügung.

Die wichtigsten Flags sind SYN und ACK. SYN ist gesetzt, wenn Client und Server während des Verbindungsaufbaus ihre ersten Pakete austauschen. Bei allen folgenden Nachrichten zwischen Client und Server ist das ACK-Flag gesetzt.

1.4 Zusammenfassung

Die einfachen Beispiele aus diesem Kapitel veranschaulichen die Konzepte, auf denen Paketfilter-Firewalls basieren. Kapitel 2 baut auf dieser Einführung auf und beschreibt, wie man mit Hilfe der Nachrichtentypen ICMP, UDP und TCP sowie der Portnummern einen Paketfilter definiert.



Teil II

Paketfilter und grundlegende Sicherheitsmaßnahmen

Kapitel 2	Konzepte für Paketfilter
Kapitel 3	Gestaltung und Installation einer Firewall
Kapitel 4	LANs, mehrfache Firewalls und Perimeter-Netze
Kapitel 5	Fehlersuche

Kapitel 2

Konzepte für Paketfilter



- 2.1 Eine Paketfilter-Firewall
- 2.2 Auswahl der Voreinstellung für den Umgang mit Paketen
- 2.3 REJECT und DENY: Pakete ablehnen oder verwerfen?
- 2.4 Filtern ankommender Pakete
- 2.5 Filtern abgehender Pakete
- 2.6 Private und öffentliche Dienste im Netz
- 2.7 Zusammenfassung

Der Begriff *Firewall* wird mit unterschiedlichen Bedeutungen gebraucht, je nachdem, welche Mechanismen für die Firewall benutzt werden, auf welcher Schicht des TCP/IP-Stacks die Firewall aufsetzt und welche Netzwerk- und Routing-Architektur sie benutzt. Die drei häufigsten Bedeutungen sind die eines Paketfilters, die eines Application Gateway (auch »abgeschirmter Host« genannt) und die eines Circuit Gateway auf der Anwendungsschicht (häufiger einfach als »Proxy-Firewall« bekannt).

Ein Paketfilter ist normalerweise schon innerhalb des Betriebssystems implementiert und arbeitet auf den Netzwerk- und Transportprotokoll-Schichten des Internet Protocols. Er schützt das System, indem er die Pakete zunächst nach Informationen im Paket-Header filtert und danach über das Routing entscheidet.

Ein Application Gateway oder abgeschirmter Host wird auf den Ebenen von Netzwerkarchitektur und Systemkonfiguration verwirklicht. Netzwerkverkehr wird niemals durch das Gateway hindurchgeleitet. Zugriffe von innen gelangen nur bis zum Gateway; auch Zugriffe von außen erfolgen nur bis auf das Gateway. Lokale Benutzer loggen sich auf dem Gateway ein und können erst von dort aus auf das Internet zugreifen. Das Gateway kann zusätzlich durch Paketfilter auf seinen internen und externen Interfaces geschützt werden.

Eine Proxy-Firewall besteht typischerweise aus separaten Anwendungen für jeden Dienst, für den ein Proxy eingerichtet wurde. Jedes Proxy-Programm benimmt sich gegenüber dem Client-Programm als Server und gegenüber dem eigentlichen Server als Client. Spezielle Client-Programme (oder speziell konfigurierte Clients) greifen statt auf den eigentlichen Server auf den Proxy-Server zu. Der Proxy stellt für den Client die Verbindung zum echten Server her, wobei aber die Absender-Adresse des Clients durch die des Proxies ersetzt wird. Die Proxy-Programme können die Datenintegrität sichern – also sicherstellen, dass die ausgetauschten Daten auch wirklich dem genutzten Dienst entsprechen –, Virens Scanner implementieren und auf höchster Ebene detaillierte Strategien für die Zugriffskontrolle durchsetzen.

In diesem Buch geht es primär um die Überlegungen, die hinter einem Paketfilter stehen. Alle drei beschriebenen Ansätze kontrollieren, wer auf welche Dienste zugreifen darf. Jeder Ansatz hat seine eigenen Stärken und Vorteile, weil auf unterschiedlichen Schichten des TCP/IP-Modells unterschiedliche Informationen verfügbar sind. Große kommerzielle Firewalls integrieren oft eine Kombination aus Paketfiltern, abgeschirmten Hosts und Proxy-Funktionalität in ein mehrschichtiges Sicherheitspaket.

In Kapitel 1 wurden die Konzepte und Informationsquellen eingeführt, auf denen eine Firewall basiert. Dieses Kapitel beschreibt, wie anhand dieser Informationen Firewall-Regeln implementiert werden. Zu einem wesentlichen Teil geht es dabei um das Filtern eingehender Pakete: welche Paketarten werden herausgefiltert, und warum. Das Filtern abgehender Pakete wird nur am Rande gestreift. Wir nehmen einmal an, dass man sich auf einer kleinen Site gegenseitig halbwegs vertraut und dass die zusätzlichen internen Sicherheitsprobleme, mit denen eine große Firma

kämpfen muss, insofern nicht existieren. In der Tat sind nicht einmal alle kommerziellen Firewalls in der Lage, abgehende Pakete zu filtern. Teilweise liegt das sicher an der ganz nennenswerten Performance-Belastung, die das Paketfiltern einem großen Router aufbürdet. Die Hardware ist im Moment noch nicht schnell genug.

Sowohl in Kapitel 1 als auch in diesem Kapitel werden immer wieder die Ports für TCP- und UDP-Dienste erwähnt. Der letzte Abschnitt dieses Kapitels, 2.6 *Private und öffentliche Dienste im Netz*, beschreibt die typischen Netzwerkdienste einer Unix-Maschine und enthält Empfehlungen, ob Sie diese Dienste auf einer Firewall aktivieren sollten.

2.1 Eine Paketfilter-Firewall

Eine IPWF Paketfilter-Firewall besteht aus einer Liste von Annahme- und Ablehnungskriterien. Diese Regeln definieren ganz genau, welche Pakete durch das Netzwerkinterface hindurch dürfen und welche nicht. Die Firewall entscheidet anhand der in Kapitel 1 beschriebenen Felder des Paket-Headers, ob ein Paket in Richtung des Empfängers weitergeleitet, kommentarlos verworfen oder – mit einer Fehlermeldung an den Absender – blockiert wird. Dabei berücksichtigt sie das Netzwerkinterface, über das das Paket läuft, sowie seine IP-Adresse, dazu die Absender- und Empfängerinformationen aus der Netzwerkschicht, die TCP- bzw. UDP-Ports aus der Transportschicht, einige der TCP-Flags zum Verbindungszustand, die ICMP-Nachrichtentypen der Netzwerkschicht sowie die Richtung, in die das Paket läuft (ob es kommt oder geht).

Auf das TCP/IP-Netzwerkmodell bezogen, arbeitet eine Paketfilter-Firewall also auf der Netzwerk- und der Transportschicht (siehe Bild 2.1).

Sie müssen sehr sorgfältig kontrollieren, welche Daten zwischen dem Internet und der direkt an das Internet angeschlossenen Maschine übertragen werden. Auf dem externen Interface zum Internet filtern Sie so genau und detailliert wie möglich, was von außen ankommt und was nach außen geschickt wird.

Die Firewall filtert ankommende (Input) und abgehende Daten (Output) separat. Für Input-Filter und Output-Filter können völlig unterschiedliche Regeln gelten. Die Regellisten heißen *Chains*, »Ketten«, weil alle Pakete nacheinander mit den Regeln aus der Liste verglichen werden, bis entweder eine Regel gefunden wird, die für das jeweilige Paket zutrifft, oder bis das Ende der Liste erreicht ist – siehe Bild 2.2.

Das hört sich nach einem sehr mächtigen System an, und das ist es auch, aber es ist keine ultimative Lösung. Es ist nur eine von vielen Schichten, die eine gute Sicherheitsstrategie umfassen sollte. Nicht alle Kommunikationsprotokolle, die von den diversen Anwendungen eingesetzt werden, eignen sich für Paketfilter. Diese Art von Filter arbeitet viel zu grob, als dass genaue Authentifizierung und Zugriffskontrolle möglich wären. Derartige Sicherheitsmaßnahmen müssen auf

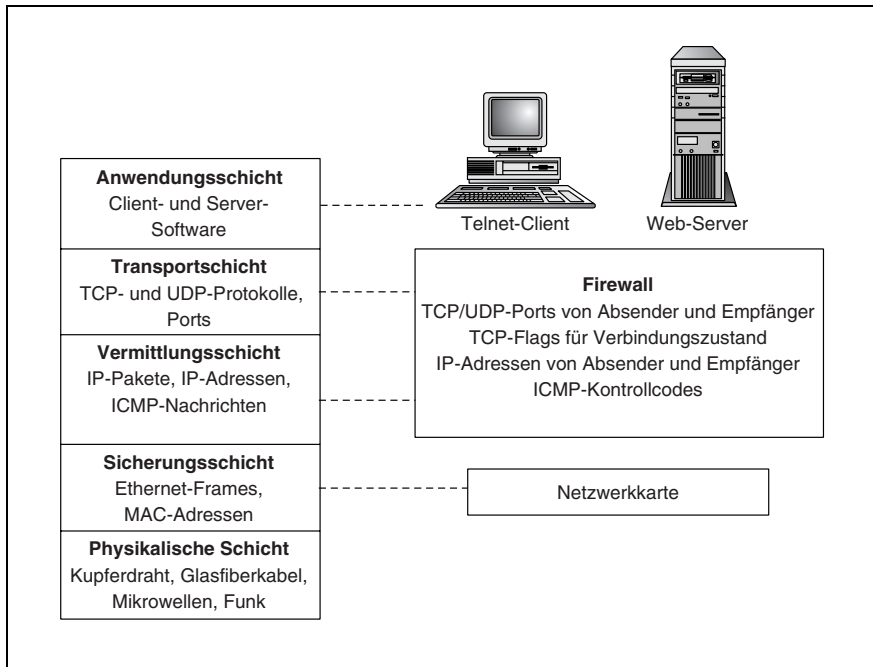


Bild 2.1: Die Firewall im TCP/IP-Referenzmodell

höheren Ebenen implementiert werden. Das Internet Protocol kann nicht überprüfen, ob der Absender auch wirklich der ist, als der er sich ausgibt. Auf dieser Ebene ist die einzige verfügbare Identifikation die Absenderadresse im IP-Header des Paketes. Diese lässt sich sehr einfach verfälschen. Darüber hinaus können weder die Netzwerk- noch die Transportschicht überprüfen, ob die übertragenen Daten stimmen. Trotz allem erlaubt die Paket-Ebene im Vergleich zu höheren Ebenen eine viel mächtigere und auch einfachere Kontrolle, wer welche Ports benutzen und welche Daten mit welchen Protokollen übertragen darf.

Ohne Paketfilter sind Filter- und Proxy-Mechanismen auf höheren Ebenen entweder nur eingeschränkt nützlich oder gänzlich uneffektiv. Zumindest in einem gewissen Maße stützen sie sich auf die Integrität des zugrunde liegenden Kommunikationsprotokolls. Jede Schicht Ihrer Sicherheitsstrategie liefert ein Mosaiksteinchen, das andere Schichten nicht so ohne weiteres bieten könnten.

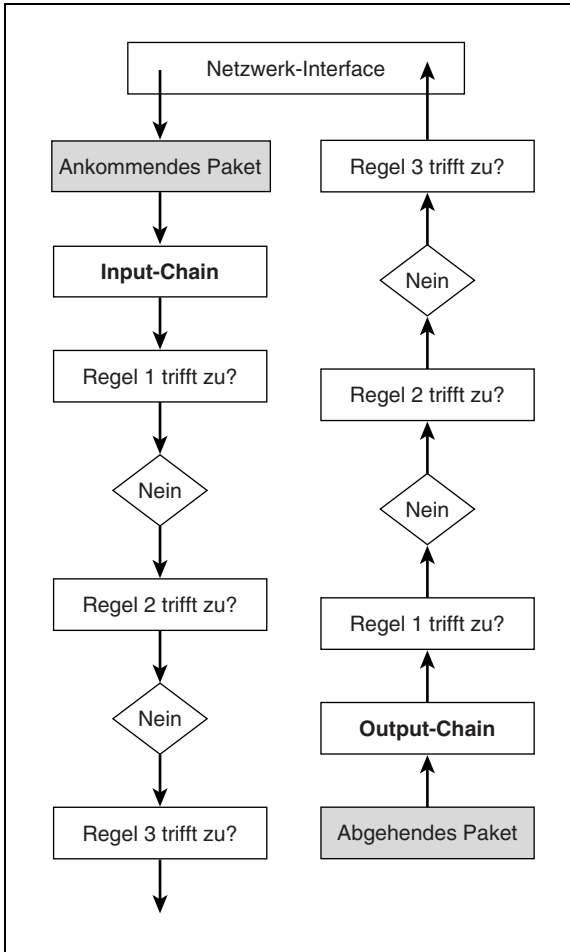


Bild 2.2: Input- und Output-Chains

2.2 Auswahl der Voreinstellung für den Umgang mit Paketen

Jede Chain der Firewall hat eine Voreinstellung für den Umgang mit Paketen, die so genannte *Policy*, sowie eine Sammlung von Maßnahmen, die in Reaktion auf bestimmte Nachrichtentypen ergriffen werden. Jede dieser Regeln wird nacheinander auf ein Paket angewandt, bis eine passende Regel gefunden ist. Wenn gar keine Regel auf ein Paket zutrifft, fällt es durch, und die voreingestellte Policy gilt.

Es gibt zwei grundsätzlich verschiedene Strategien, wie man Firewalls implementiert:

- Per Voreinstellung wird alles abgewiesen; nur explizit ausgewählte Pakete dürfen passieren.
- Per Voreinstellung darf alles passieren; nur explizit ausgewählte Pakete werden abgewiesen.

Empfehlenswert ist nur die erste Strategie, normalerweise alles abzuweisen. Eine unter dieser Prämisse eingerichtete Firewall wird mit höherer Wahrscheinlichkeit sicher sein; allerdings müssen alle gewünschten Dienste einzeln erlaubt werden (siehe Bild 2.3). Sie müssen also die Kommunikationsprotokolle für jeden Dienst, den Sie aktivieren, verstehen. Anfangs erfordert dieser Ansatz viel mehr Arbeit, um den Zugriff auf das Internet zu ermöglichen. Einige kommerzielle Firewall-Produkte gestatten nur diese Strategie.

Die zweite angesprochene Sicherheitsstrategie, alle Pakete zunächst einmal passieren zu lassen, wenn sie nicht explizit verboten sind, ist bei der Einrichtung natürlich viel einfacher. Sie müssen dann aber *a priori* alle denkbaren Zugriffstypen kennen, die Sie vielleicht einmal verhindern wollen (siehe Bild 2.4). Es ist denkbar, dass Sie einen gefährlichen Zugriff erst erkennen, wenn es zu spät ist, oder dass Sie später einen unsicheren Dienst einrichten, ohne daran zu denken, den Zugriff darauf von außen zu blockieren. Unter dem Strich ist die Entwicklung einer sicheren Firewall nach dieser Strategie viel zeitaufwändiger, schwerer und letztlich auch fehleranfälliger.

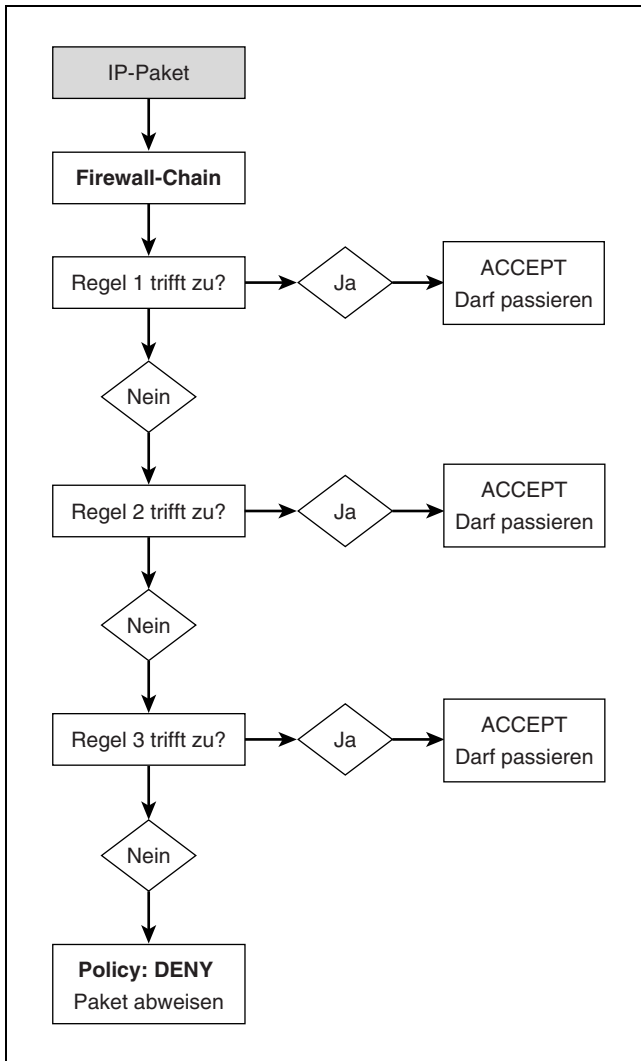


Bild 2.3: Policy: Per Voreinstellung wird alles abgewiesen

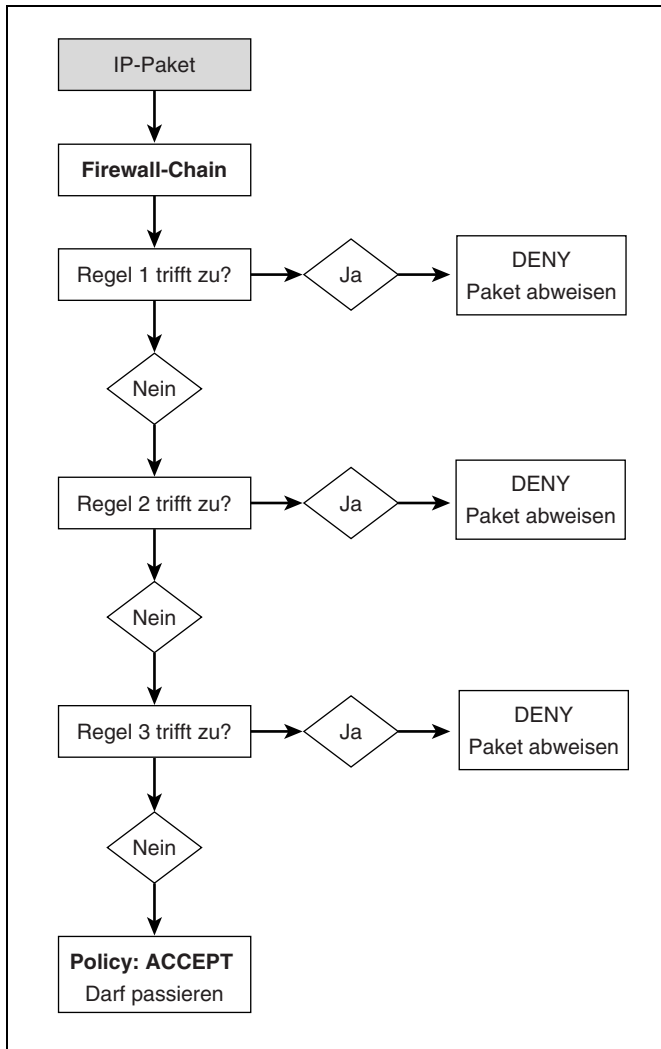


Bild 2.4: Policy: Per Voreinstellung wird alles angenommen

2.3 REJECT und DENY: Pakete ablehnen oder verwerfen?

Die Firewall-Mechanismen von IPFW und `ipchains` überlassen Ihnen die Entscheidung, ob Sie ein Paket ablehnen (REJECT) oder verwerfen (DENY) wollen. Wo liegt der Unterschied?

Wenn ein Paket durch REJECT abgelehnt wird, löscht Ihr Computer das Paket und schickt dem Absender eine ICMP-Fehlermeldung (siehe Bild 2.5). Wenn er ein

Paket durch **DENY** verwirft, löscht er es ohne einen solchen Hinweis an den Absender.

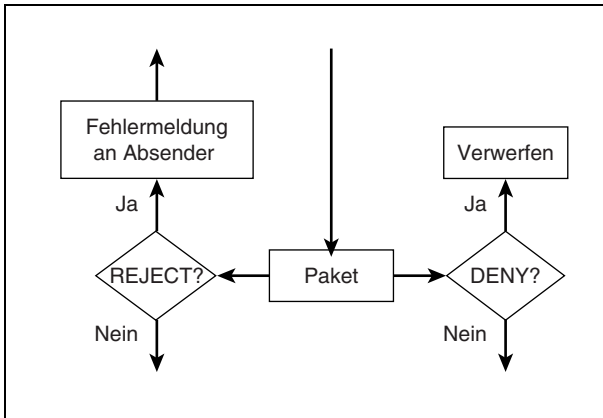


Bild 2.5: REJECT und DENY

Das kommentarlose Verwerfen durch **DENY** ist fast immer die bessere Wahl, und zwar aus drei Gründen: Erstens verdoppelt eine Fehlermeldung den Traffic auf dem Netz. Die allermeisten Pakete werden verworfen, weil sie bösartig sind, nicht weil jemand ganz unschuldig auf einen Dienst zugreifen wollte, den Sie eben zufällig nicht anbieten. Zweitens kann jedes Paket, auf das Sie antworten, als Teil eines Denial-of-Service-Angriffes eingesetzt werden. Drittens liefert jede Antwort, selbst eine Fehlermeldung, dem Möchtegern-Angreifer potenziell nützliche Informationen.

2.4 Filtern ankommender Pakete

Im Hinblick auf die Sicherheit Ihrer Site sind die Pakete, die auf Ihrem externen Interface ankommen – die Input-Chain – interessanter als die abgehenden Pakete. Wie zuvor schon erwähnt, können Sie nach Absender- und Empfänger-IP-Adresse, nach Absender- und Empfänger-Port und nach den TCP-Status-Flags filtern. Früher oder später werden wir in den folgenden Abschnitten alle diese Informationen benutzen.

2.4.1 Filtern nach Absender-IP

Auf der Paketebene kann man den Absender eines Paketes nur über die Absender-IP-Adresse im Paket-Header bestimmen. Das öffnet gefälschten Absender-Adressen natürlich Tür und Tor. Dabei setzt der Absender statt seiner eigenen IP-Adresse irgendeine andere Adresse in das Absender-Feld ein. Die verwendete IP existiert möglicherweise gar nicht, oder sie gehört zu jemand anderem. Unfreund-

liche Zeitgenossen können auf diese Weise in Ihr System einbrechen, sich beim Angriff auf andere Computer als Ihr System ausgeben, sich beim Angriff auf Ihr System als ein anderer Computer verkleiden oder Sie sonstwie über den wahren Ursprung eingehender Nachrichten in die Irre führen.

Gefälschte und illegale Absender-Adressen

Es gibt sechs Hauptgruppen von Absender-IPs, die Sie auf Ihrem externen Interface auf jeden Fall abweisen sollten. Das betrifft alle eingehenden Pakete, die angeblich von folgenden IP-Adressen ausgehen:

- Ihre eigene IP-Adresse: Niemals wird ein reguläres eingehendes Paket von Ihrer eigenen Maschine kommen. Dabei handelt es sich um eine der wenigen Formen der IP-Fälschung, die Sie schon auf der Paketebene erkennen können. Eingehende Pakete, die vorgeben, von Ihrem eigenen Computer abgeschickt worden zu sein, sind mit Sicherheit gefälscht. Bei allen anderen eingehenden Paketen können Sie sich nicht ganz sicher sein, ob die angegebene Absender-IP stimmt.
- Die *privaten* IP-Adressen der Klassen A, B und C: Eine Untermenge der IP-Adressen der Klassen A, B und C ist für den Gebrauch in privaten Netzwerken reserviert. Diese Adressen sollten auf dem Internet nicht benutzt werden. Dadurch kann jedermann solch eine IP intern verwenden, ohne offizielle IP-Adressen einkaufen zu müssen. Vom Internet sollten Sie niemals Pakete von einer dieser Absenderadressen erhalten.

(Auf den Subnetzen einiger Internet-Provider sieht man diese privaten Adressen durchaus. Dabei handelt es sich entweder um falsch konfigurierte Systeme oder um absichtlich gefälschte Adressen. Wenn jemand solche Adressen ins Internet lässt, sehen Sie sie natürlich.)

- Private Adressen der Klasse A liegen im Bereich von 10.0.0.0 bis 10.255.255.255.
- Private Adressen der Klasse B liegen im Bereich von 172.16.0.0 bis 172.31.255.255.
- Private Adressen der Klasse C liegen im Bereich von 192.168.0.0 bis 192.168.255.255.
- Multicast-Adressen der Klasse D: IP-Adressen der Klasse D sind für den Gebrauch als Empfänger-IPs bei Multicast-Broadcasts reserviert, z.B. bei Audio- und Video-Übertragungen. Sie liegen im Bereich von 224.0.0.0 bis 239.255.255.255. Diese Adressen sollten auf dem Internet nicht als Absender vorkommen.
- Reservierte Adressen der Klasse E: IP-Adressen der Klasse E sind zukünftigen und experimentellen Anwendungen vorbehalten und werden nicht öffentlich vergeben. Sie liegen im Bereich von 240.0.0.0 bis 247.255.255.255. Sie sollten niemals ein Paket von einer Absender-Adresse aus diesem Bereich zu Ge-

sicht bekommen, und die Chancen stehen gut, dass dies nicht geschieht – die Netzwerke der amerikanischen Militärs und des Geheimdienstes sind sehr darauf bedacht, ihre Daten nicht ins Internet zu lassen.

- **Loopback-Adressen:** Das Loopback-Interface ist ein privates Netzwerkinterface, das von Unix für lokale netzwerkbasierende Dienste eingesetzt wird. Statt lokalen Netzwerkverkehr durch die Treiber des Netzwerkinterfaces zu schicken, nimmt das Betriebssystem das Loopback-Device als Abkürzung und verbessert dadurch die Performance. Definitionsgemäß ist Loopback-Verkehr für das System gedacht, das ihn erzeugt. Er erscheint nie auf dem Netzwerk. Der Loopback-Bereich reicht von 127.0.0.0 bis 127.255.255.255. Normalerweise kennen Sie ihn als 127.0.0.1, als `localhost` oder als das Loopback-Interface `lo`.
- **Fehlerhafte Broadcast-Adressen:** Broadcast-Adressen sind besondere IPs, die für alle Maschinen eines Netzwerks gelten. Eine typische Broadcast-Absender-Adresse ist 0.0.0.0. DHCP-Clients können eingehende Broadcast-Pakete von 0.0.0.0 sehen. Ich kenne aber keine andere Situation, in der ein legitimes Paket von Adresse 0.0.0.0 ausgehen könnte. Auf alle Fälle ist es keine gültige Absenderadresse für die Kommunikation zwischen zwei definierten Computern. Wenn so eine Adresse außerhalb des Kontextes einer Broadcast-Anwendung auftritt, ist sie gefälscht.

Problematische Sites herausfiltern

Seltener, aber immer noch recht oft, filtert man nach der Absenderadresse, um alle Zugriffe von einer ausgewählten Maschine oder – häufiger – den IP-Adressen eines ganzen Netzwerkes zu sperren. Auf diese Weise geht die Internet-Gemeinschaft mit problematischen Sites und Internet-Providern um, die ihren Benutzern alles erlauben. Wenn eine Site erst einmal den Ruf erwirbt, ein schlechter Internet-Nachbar zu sein, neigen andere Sites dazu, sie komplett zu sperren.

Individuell ist eine Zugriffssperre für ein ausgewähltes Netzwerk praktisch, wenn einzelne Benutzer dieses Netzes immer wieder als Störenfriede auftreten.

Eingehende Pakete nur von ausgewählten Hosts annehmen

Bestimmte Arten eingehender Pakete wollen Sie möglicherweise nur von handverlesenen anderen Sites oder Benutzern annehmen. In diesen Fällen definieren die Firewall-Regeln entweder die genaue IP-Adresse oder einen begrenzten Bereich von IPs, von denen solche Pakete akzeptiert werden.

Eingehende Pakete kommen einerseits von fremden Servern, die auf Ihre Anfragen antworten. Obwohl Sie auf manche Dienste, z.B. Web oder FTP, ganz regulär überall zugreifen wollen, laufen andere Server nur auf Computern Ihres Internet-Providers oder auf gezielt ausgewählten Rechnern. Beispiele für Dienste, die vermutlich nur Ihr ISP anbietet, sind POP für das Abholen von Mail, DHCP für die dynamische Zuweisung von IP-Adressen, und vielleicht auch der Domain Name Service DNS.

Andererseits kommen eingehende Pakete von fremden Clients, die Server auf Ihrem Computer benutzen wollen. Auch hier gilt, dass einige dieser Anfragen von überall her gestellt werden dürfen – z.B. Verbindungen zu Ihrem Web-Server –, während Sie andere Dienste nur wenigen Ihrer Freunde anbieten wollen, z.B. telnet, ssh und finger.

2.4.2 Filtern nach Empfänger-IP

Das Filtern nach der Empfänger-IP ist bei eingehenden Paketen nicht besonders wichtig. Ihre Netzwerkkarte ignoriert sowieso alle normalen Pakete, die nicht an sie gerichtet sind. Die einzige Ausnahme sind Broadcast-Pakete, die an alle Computer auf einem Netzwerk geschickt werden.

Die allgemeine Broadcast-Empfänger-Adresse ist 255.255.255.255. Die Broadcast-Adresse kann aber auch genauer definiert werden und besteht dann aus Ihrer Netzwerkadresse, wobei alle weiteren Bits der IP auf 1 gesetzt sind. Wenn beispielsweise Ihr Internet-Provider die Netzwerkadresse 192.168.0.0 benutzt und Ihre eigene IP 192.168.10.30 ist, dann sehen Sie von Ihrem ISP möglicherweise Broadcast-Pakete an die Adressen 192.168.255.255 und 255.255.255.255.

Broadcasts an die Empfänger-Adresse 0.0.0.0 muss man so ähnlich interpretieren wie zielgerichtete Pakete, die angeblich von einer Broadcast-Adresse abgeschickt wurden – siehe die Beschreibung weiter oben im Abschnitt *Gefälschte und illegale Absender-Adressen*. Hierbei werden Broadcast-Pakete an die Absender-Adresse 0.0.0.0 statt an die korrekte Empfänger-Adresse 255.255.255.255 gerichtet. In diesem Fall ist die Absicht des Paketes ganz eindeutig: Man versucht, Ihren Computer als Unix-Maschine zu identifizieren. Aus historischen Gründen reagiert Netzwerkcode, der von BSD-Unix abgeleitet wurde, auf die Verwendung von 0.0.0.0 als Broadcast-Empfänger mit einer ICMP-Fehlermeldung Typ 3. Andere Betriebssysteme ignorieren solche Pakete einfach. Dies ist übrigens auch ein gutes Beispiel, warum man unerwünschte Pakete lieber kommentarlos verwerfen (DENY) als mit einer Fehlermeldung ablehnen (REJECT) sollte: Die Fehlermeldung ist in diesem Fall genau das, was der Angreifer sehen möchte.

2.4.3 Filtern nach Absender-Port

Der Absender-Port eines eingehenden Paketes ist abhängig von dem Programm auf dem anderen Computer, welches das Paket erzeugt hat. Alle Zugriffe fremder Clients auf die von Ihnen angebotenen Dienste folgen dabei einem Muster, und alle Antworten fremder Server auf die Anfragen Ihrer Clients entsprechen einem anderen Muster.

Alle eingehenden Anfragen und Verbindungen fremder Clients für Server auf Ihrer Site haben einen Absender-Port im unprivilegierten Bereich. Wenn Sie einen Web-Server betreiben, sollten alle Anfragen an diesen Web-Server einen Absender-Port zwischen 1024 und 65535 haben.

Antworten fremder Server, auf die Sie zugreifen, kommen von einem Absender-Port, der dem jeweils benutzten Dienst entspricht. Wenn Sie z.B. eine Verbindung zu einer Web-Site aufbauen, setzt dieser Server den Absender-Port aller Pakete auf 80, die Portnummer von `http`.

2.4.4 Filtern nach Empfänger-Port

Der Empfänger-Port eingehender Pakete bestimmt, für welches Programm bzw. welchen Dienst auf Ihrem Computer das Paket bestimmt ist. Wie beim Absender-Port kann man auch hier unterscheiden nach eingehenden Anfragen fremder Clients an lokale Server einerseits, und eingehenden Antworten fremder Server an lokale Clients andererseits.

Bei eingehenden Anfragen und Verbindungen fremder Clients für Server auf Ihrem Computer ist der Empfängerport auf die Portnummer des jeweils genutzten Dienstes gesetzt. Z.B. hat ein Paket, das für Ihren Web-Server gedacht ist, einen Empfänger-Port von 80 entsprechend `http`.

Antwortet ein fremder Server auf eine Anfrage von Ihnen, liegt der Empfänger-Port im unprivilegierten Bereich. Wenn Sie auf eine Web-Site im Internet zugreifen, werden die Antworten an einen Empfänger-Port zwischen 1024 und 65535 gesandt.

2.4.5 Filtern nach TCP-Status-Flags

Die Regeln, welche eingehenden Pakete angenommen werden sollen, können auch die Flags berücksichtigen, mit denen der Zustand der TCP-Verbindung angezeigt wird. Alle TCP-Verbindungen benutzen die gleiche Gruppe möglicher Verbindungszustände. Wegen des dreiteiligen Handshakes beim Verbindungsaufbau unterscheiden sich die Flags bei Clients und Servern voneinander.

Wenn ein fremder Client eine Verbindung aufbaut, ist im ersten Paket des Handshakes das SYN-Flag gesetzt, das ACK-Flag aber nicht. Bei allen Paketen, die danach ankommen, ist nur noch das ACK-Flag gesetzt. Die Firewall-Regeln für einen Server sollten alle eingehenden Pakete zulassen, unabhängig davon, welchen Wert SYN oder ACK haben.

Von einem fremden Server ankommende Pakete sind immer Antworten auf einen Verbindungsaufbau, den Ihr Client-Programm initiiert hat. Bei jedem Paket ist also das ACK-Flag gesetzt. Firewall-Regeln für einen Client sollten Pakete vom Server daher nur mit gesetztem ACK-Flag zulassen. Ein normaler Server wird normalerweise nicht selbst eine Verbindung zum Client aufbauen.

2.4.6 Abtastversuche und Scans

Ein *Abtastversuch* – eine »Probe« – ist ein Versuch, eine Verbindung zu einem bestimmten Port aufzubauen oder zumindest irgendeine Reaktion davon zu erhal-

ten. Ein *Portscan* oder kurz *Scan* ist eine ganze Serie von Abtastversuchen, mit denen verschiedene Ports untersucht werden. Scans sind oft vollautomatisiert.

An und für sich sind Abtastversuche und Scans harmlos. Auf dem Internet können Sie eigentlich nur durch eine Probe herausfinden, ob eine Site einen bestimmten Dienst anbietet. Wenn Sie die URL zu einer Site nicht haben, woher wissen Sie dann, ob ein Web-Server dazu existiert? Sie versuchen einfach, die Homepage aufzurufen. Mit anderen Worten, Sie führen einen Abtastversuch des *http*-Ports durch.

Leider sind heute die wenigsten Probes und Scans harmlos. Meistens gehören sie zum ersten Teil eines Einbruchversuchs, wenn man Informationen über möglicherweise interessante Schwächen Ihres Systems gewinnen will. Gerade seit 1998 beobachtet man weltweit einen exponentiellen Anstieg der Scans. Automatisierte Scan-Tools sind weit verbreitet, und oft koordinieren Black Hats ihre Bemühungen in dieser Hinsicht.

Komplette Portscans

Ein kompletter Portscan überprüft ungezielt einen großen Bereich von Ports, möglicherweise sogar alle zulässigen Ports (siehe Bild 2.6). Diese Scans gehen meistens von älteren Netzwerktools wie *satan* oder *strobe* aus und werden immer seltener. Raffiniertere Tools wie *mscan*, *sscan* und *nscan* gehen gezielt vor und verbreiten sich zusehends.

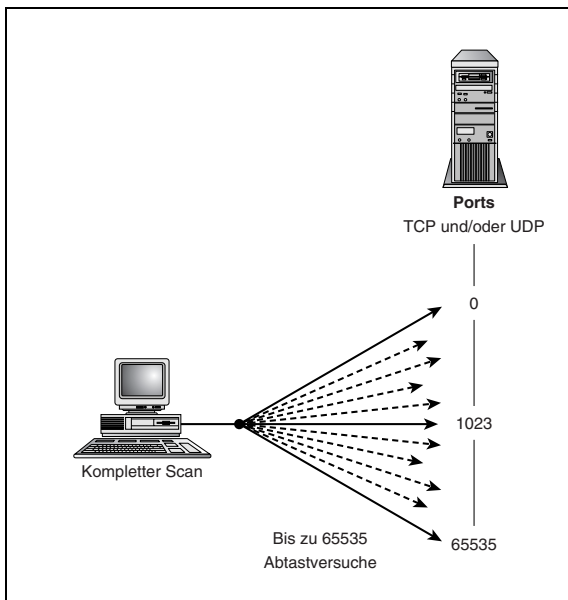


Bild 2.6: Ein kompletter Portscan

Gezielte Portscans

Gezielte Portscans suchen nach ganz bestimmten Sicherheitslücken (siehe Bild 2.7). Die neueren und ausgeklügelten Tools wollen herausfinden, welche Versionen von Hardware, Betriebssystem und Software installiert sind. Sie konzentrieren sich damit gezielt auf bekannte Sicherheitslücken einzelner Opfer.

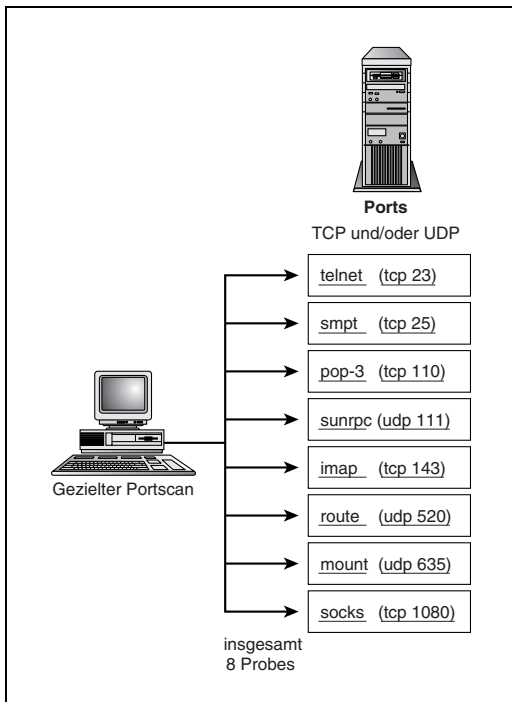


Bild 2.7: Ein gezielter Portscan

Typische Target-Ports

Einige Ports werden besonders oft untersucht oder in einem Portscan abgefragt. Der potenzielle Eindringling sucht dabei möglicherweise nach einer bestimmten Sicherheitslücke, z.B. nach einem unsicheren Mail-Server oder einem ungeschützten RPC-Portmapper.

Eine ausführliche Portliste finden Sie in Kapitel 6, Abschnitt 6.5 *Die Protokolldateien Ihres Systems*. Hier werden der Anschaulichkeit halber nur ein paar häufige Ports erwähnt:

- Eingehende Pakete vom reservierten Port 0 sind immer verdächtig. Dieser Port wird nicht legitim verwendet.
- Verbindungsversuche zu den TCP-Ports 0 bis 5 sind typisch für das Programm `sscan`.

- Beliebte Target-Ports sind telnet (23/tcp), smtp (25/tcp), pop-3 (110/tcp), sun-rpc (111/udp/tcp), imap (143/tcp), snmp (161/udp), route (520/udp) und mount (635/udp). Dabei handelt es sich um die potenziell verwundbarsten Angriffspunkte eines Systems. Weil diese Dienste so häufig sind, sollten Sie sie entweder gar nicht nach außen anbieten oder zumindest den Zugriff von außen sehr sorgfältig kontrollieren.
- Probes nach NetBIOS (TCP und UDP 137 und 138, TCP 139), Netbus (12345/tcp) und BackOrifice (31789) sind lästig, stellen aber keine Gefahr für ein Unix-System dar. Opfer kann in diesem Fall nur Windows sein.

Vorsicht Paranoia – Wie man auf Portscans reagiert

Praktisch täglich zeigen die Firewall-Logs in `/var/log/messages` alle möglichen fehlgeschlagenen Verbindungsversuche.

Versuchen die Leute wirklich so oft, Ihr System zu hacken? Ja. Hat jemand in Ihr System eingebrochen? Nein. Die Ports sind gesperrt. Die Firewall tut ihren Job. Es handelt sich um *fehlgeschlagene* Verbindungsversuche, die von der Firewall abgewiesen wurden.

Ab wann sollten Sie unerwünschte Zugriffsversuche melden? Ab welchem Punkt ist die Sache auch wichtig genug, dass Sie die entsprechende Zeit investieren? Wann sagen Sie »genug ist genug« und kümmern sich um wichtigere Dinge? Sollten Sie vielleicht nicht doch jedesmal an `abuse@irgendwo.com` schreiben?

Es gibt keine »richtige« Antwort. Wie Sie reagieren, ist Ihre persönliche Entscheidung. Das ganze Thema kann Kreuzzüge und heilige Kriege auslösen. Es gibt einfach keine klare und eindeutig richtige Erwiderung auf Probes und Portscans. Es hängt alles von Ihrer Persönlichkeit ab, von Ihrem Gefühl, davon, wie Sie für sich einen ernst zu nehmenden Portscan definieren, und irgendwo auch von ihrem sozialen Verantwortungsbewusstsein.

Wenn Sie all das im Hinterkopf behalten, darf ich trotzdem ein paar Richtlinien vorschlagen.

Ursache ist am häufigsten ein vollautomatisierter Scan, ein Benutzerfehler, ein historisch korrekter Zugriff, Dummheit, Neugier oder fehlerhafte Software.

Sie sollten individuelle, isolierte, einzelne Zugriffsversuche auf telnet, ssh, ftp, finger und ähnlich beliebte Dienste, die Sie eben nicht anbieten, einfach ignorieren. Probes und Scans gehören zum Leben auf dem Netz, sie sind häufig und in der Regel keine Gefahr. Sie gehören in die gleiche Schublade wie Vertreter, Leute, die Ihnen am Telefon eine Versicherung verkaufen wollen oder die sich einfach verwählt haben, oder Werbepost. Der Tag ist einfach nicht lang genug, um sich um das alles einzeln zu kümmern.

Andererseits sind manche Versuche auch hartnäckiger. Möglicherweise ist eine zusätzliche Firewall-Regel nützlich, die den Angreifer aussperrt, oder sogar seinen ganzen Adressbereich, wenn seine Domain einen schlechten Ruf hat oder er unterschiedliche IPs benutzt.

Wenn jemand nur ein paar ausgesuchte Ports untersucht, die alle bekannte Sicherheitslöcher beherbergen, und er zu allem Überfluss auch noch einen offenen Port findet, dann wird in der Regel ein Einbruchversuch auf dem Fuße folgen. Weitergehende Scans wollen oft eine ganze Domain oder ein Subnetz auf mögliche Lücken abtasten. Hacker-Tools scannen diese Ports auch nacheinander.

Von Zeit zu Zeit werden Ihnen ernsthafte Einbruchversuche begegnen. Allerspätstens jetzt sollten Sie aktiv werden. Schicken Sie dem Eindringling einen Brief (per Post, nicht per E-Mail). Melden Sie ihn. Prüfen Sie Ihre Sicherheitseinstellungen doppelt und dreifach. Beobachten Sie, was der Eindringling tut. Sperren Sie ihn aus. Sperren Sie seinen ganzen IP-Block aus. Wenn nötig, nehmen Sie Ihr System vom Netz.

Hinweis***Historisch gefährliche Ports***

Weitere Informationen zu Ports, die in der Vergangenheit gefährlich waren, liefert der Artikel »Packet Filtering for Firewall Systems«, den Sie von www.cert.org erhalten.

Manche Administratoren nehmen jeden einzelnen Vorfall ernst. Sie sagen: Selbst wenn mein eigenes System sicher ist, wird jemand anders möglicherweise doch betroffen sein. Ein anderer merke vielleicht nicht einmal, was da vorgehe. Solche Versuche zu melden, sagen sie, sei allgemeine Bürgerpflicht.

Wie sollten Sie auf Portscans reagieren? Wenn Sie diesen Leuten schreiben, oder ihrem Postmaster, dem Netzwerkzentrum ihres Internet-Providers, oder dem Koordinator des Adressblocks, dann bleiben Sie höflich. Zeigen Sie, dass Zweifel bestehen. Wer überreagiert, ist meistens im Unrecht. Was Ihnen wie ein ernsthafter Einbruchversuch erscheint, ist vielleicht ja nur ein neugieriges Kind, das mit einem neuen Programm herumspielt. Ein paar höfliche Zeilen an `abuse`, `root` oder `postmaster` können Wunder bewirken. Es ist nützlicher, jemandem die Netiquette beizubringen, als ihn aus dem Netz zu vertreiben. Und vielleicht erweist sich der andere ja wirklich als unschuldig.

Es kommt auch vor, dass das angreifende System seinerseits gehackt worden ist. Der Eigentümer weiß in diesem Fall gar nicht, was geschieht, und wird Ihnen dankbar sein, wenn Sie ihn über die Sachlage unterrichten.

2.4.7 Denial-of-Service-Angriffe

Denial-of-Service-Angriffe (»Dienstverweigerung«, oft als »DoS« abgekürzt) wollen Ihr System so mit Paketen überfluten, dass Ihre Internet-Anbindung unterbrochen oder massiv gestört wird, Ihre Server so sehr beansprucht werden, dass normale Anfragen nicht mehr bearbeitet werden können, oder – im schlimmsten Fall – Ihr System komplett abstürzt. Meistens wird Ihr System zu sehr belastet,

als dass Sie noch etwas sinnvolles damit tun könnten, und kritische Ressourcen werden aufgebraucht.

Ein vollständiger Schutz gegen Denial-of-Service-Angriffe ist nicht möglich. Sie können in jeder Form auftreten, die der Angreifer sich nur ausdenken kann. Alles, was irgendeine Antwort Ihres Systems auslöst, was irgendwelche Ressourcen auf Ihrem System beansprucht, was einen anderen Computer dazu bringt, die Kommunikation mit Ihnen einzustellen – all das kann einen Teil eines Denial-of-Service-Angriffes darstellen.

Einige wichtige und klassische DoS-Angriffe sind TCP-SYN-Flooding, ping-Flooding, UDP-Flooding und ICMP-Redirect-Bomben.

Hinweis

Denial-of-Service-Angriffe

Weitere Informationen zu Denial-of-Service-Angriffen finden Sie im Artikel »Denial of Service«, der von www.cert.org erhältlich ist.

TCP-SYN-Flooding

Eine TCP-SYN-Flut verbraucht Ihre Systemressourcen, bis keine weiteren TCP-Verbindungen mehr aufgebaut werden können. Der Angriff benutzt den grundlegenden dreiteiligen TCP-Handshake zum Verbindungsaufbau, zusammen mit gefälschten IP-Absenderadressen.

Der Angreifer beginnt mit gefälschter IP-Adresse einen Verbindungsaufbau zu einem Ihrer TCP-basierten Dienste. Als Client, der eine TCP-Verbindung aufbauen will, schickt er Ihnen ein SYN-Paket. Ihr Computer reagiert mit einer Bestätigung, also SYN-ACK. Allerdings ist die Adresse, an die diese Bestätigung geschickt wird, in diesem Fall nicht die Adresse des Clients – sie ist ja gefälscht, vermutlich existiert sie gar nicht. Der letzte Schritt des Verbindungsaufbaus, in dem Ihr Computer auf ein ACK als Antwort wartet, wird also niemals stattfinden.

Die begrenzten Netzwerkressourcen werden dadurch verbraucht. Die Verbindung bleibt halboffen, bis der Timeout für den Verbindungsaufbau eintritt. Der Hacker überflutet Ihren Port jetzt mit SYN um SYN, viel schneller, als die Timeouts die entsprechenden Ressourcen wieder freigeben. Irgendwann sind alle Ressourcen verbraucht, und der Aufbau weiterer Verbindungen ist nicht möglich. Wenn sich der Angriff gegen Ihren smtp-Port richtet, empfangen Sie keine E-Mail mehr. Wenn Ihr http-Port betroffen ist, können die Leute Ihren Web-Server nicht mehr benutzen.

Anwenden von Linux stehen mehrere Abhilfen zur Verfügung.

Die erste ist das bereits beschriebene Filtern nach Absender-IP. Dadurch werden die am häufigsten gefälschten IP-Adressen schon einmal herausgefiltert, aber natürlich garantiert niemand, dass Ihr konkreter Angreifer tatsächlich eine dieser Adressen benutzt.

Der zweite Schritt in der Verteidigung besteht darin, den Kernel mit eingeschalteten SYN-Cookies zu übersetzen. Dabei handelt es sich um eine sehr spezifische Abwehr gegen SYN-Flooding. SYN-Cookies sind in RedHat 6.0 und höher sowie den meisten anderen aktuellen Linux-Distributionen bereits aktiviert; Sie brauchen in diesem Fall nichts weiter zu tun. Bei früheren Versionen müssen Sie diese Option gegebenenfalls explizit mit `make config`, `make menuconfig` oder `make xconfig` einschalten und den Kernel dann neu übersetzen und installieren.

Hinweis***SYN-Flooding und Fälschen von IP-Adressen***

Zusätzliche Informationen zu den Themen SYN-Flut und IP-Fälschung bietet das CERT-Advisory CA-96.21, »TCP SYN Flooding and IP Spoofing Attacks«, erhältlich unter ftp://info.cert.org/pub/cert_advisories/CA-96.21.tcp_syn_flooding.

ping-Flooding

Jede Nachricht, die eine Antwort Ihres Computers auslöst, kann zur Verschlechterung Ihrer Netzwerkanbindung benutzt werden. Das System ist dann die meiste Zeit nur mit Antworten beschäftigt. Ein typisches Beispiel ist die ICMP Echoanforderung (*echo request*), die `ping` sendet.

Bei einer älteren Sicherheitsschwäche namens *Ping of Death* (»Todesping«) werden sehr große `ping`-Pakete verschickt. Anfällige Computer stürzen dann ab. Linux ist nicht verwundbar, ebenso wie viele andere aktuelle Unix-Betriebssysteme. Wenn Ihre Firewall aber ältere Systeme schützen soll, können diese durchaus angreifbar sein.

Der Ping of Death zeigt, wie auch das einfachste Protokoll und der einfachste Datenaustausch von kreativen Hackern ausgenutzt werden können. Nicht alle Black Hats wollen wirklich in Ihren Computer einbrechen. Manchmal geht es nur um Destruktion. In diesem Fall will der Angreifer die Maschine abstürzen lassen.

`ping` ist ein sehr nützliches und grundlegendes Netzwerktool. Vermutlich wollen Sie es nicht völlig unterbinden. Im heutigen Internet empfehlen konservative Leute das Sperren eingehender `ping`-Anfragen oder zumindest die rigorose Beschränkung der Hosts, die solche Anfragen durchführen dürfen. Wegen der langen Geschichte von `ping` als Teil von Denial-of-Service-Angriffen reagieren viele Sites überhaupt nicht mehr auf `pings` von außen.

UDP-Flooding

Das UDP-Protokoll eignet sich ganz besonders gut für Denial-of-Service-Angriffe. Im Gegensatz zu TCP ist UDP *stateless*, es kennt keine Verbindungen, die geöffnet und wieder geschlossen werden. Eine wirksame Flusskontrolle existiert nicht. Es gibt keine Flags, die den Zustand einer Verbindung anzeigen könnten, keine Sequenznummern für Pakete, keine Information darüber, welches Paket

als nächstes erwartet wird. Es ist ziemlich einfach, einen Computer mit den Antworten auf UDP-Anfragen so zu beschäftigen, dass für den regulären Netzwerkverkehr keine Bandbreite mehr übrig bleibt.

Weil UDP-Dienste inhärent unsicherer sind als TCP-Dienste, schalten viele Sites alle UDP-Ports ab, die nicht unbedingt benötigt werden. Wie schon früher erwähnt, basieren alle üblichen Dienste im Internet ohnehin auf TCP. Die Firewall, die wir in Kapitel 3 bauen werden, begrenzt UDP sehr sorgfältig auf die Rechner, die lebenswichtige UDP-Dienste zur Verfügung stellen.

Hinweis

Denial-of-Service-Angriffe gegen UDP-Ports

Eine Beschreibung eines Denial-of-Service-Angriffes, der UDP-Dienste ausnutzt, finden Sie im CERT-Advisory CA-96.01, »UDP Port Denial-of-Service Attack«, das Sie von www.cert.org erhalten.

ICMP-Redirect-Bomben

Die ICMP Nachricht Typ 5 *Redirect* (»Umleitung«) weist den Empfänger an, seine Routingtabellen zugunsten einer kürzeren Route umzustellen. Wenn bei Ihnen *routed* oder *gated* läuft und Sie *Redirect*-Nachrichten akzeptieren, kann ein Black Hat Ihr System dazu bringen, dass es die angreifende Maschine für einen Teil Ihres LANs oder einen Rechner Ihres Internet-Providers hält. Im Extremfall wird Ihr Computer allen Internet-Traffic an einen anderen Computer weiterleiten!

Denial-of-Service-Angriffe und andere Ressourcen

Die Anbindung ans Internet ist nicht das einzige Problem bei Denial-of-Service-Angriffen. Ein paar weitere Gebiete, die Sie bei der Konfiguration Ihres Systems im Auge behalten sollten, sind folgende:

- Zu viele Fehlermeldungen in den Logdateien oder der Empfang zahlreicher riesiger E-Mails könnten Ihre Festplatte füllen. Hier sind gegebenenfalls Ressourcen-Beschränkungen (siehe die Manual-Page zu `getrlimit(2)`) sowie separate Partitionen für schnell wachsende oder sich rasch ändernde Dateisysteme geboten.
- Server können abstürzen, wenn zu große Datenmengen das Ende der Eingabepuffer überschreiben (*Buffer Overflow*), oder wenn sie unerwartete Daten erhalten. Wenn Sie nicht besondere Vorsichtsmaßnahmen ergreifen, sind CGI-Skripten besonders anfällig. Viele der aktuellen Sicherheitslücken in Servern hängen mit Buffer-Overflows zusammen. Es ist sehr wichtig, immer am Ball zu bleiben und die neuesten Patches und Versionen Ihrer Software zu installieren.
- Hauptspeicher, Einträge in den Prozesstabellen, Prozessorzeit und andere Ressourcen können durch den schnellen und wiederholten Aufruf von Netzwerkdiensten erschöpft werden. Dagegen kann man nicht viel tun, außer eventuell

konfigurierbare Beschränkungen für einzelne Dienste auszunutzen, SYN-Cookies einzuschalten und Pakete an gesperrte Ports abzuweisen, statt sie mit einer Fehlermeldung zurückzuschicken.

Hinweis***Denial-of-Service-Angriffe über E-Mail***

Der Artikel »Email Bombing and Spamming« enthält eine Beschreibung eines Denial-of-Service-Angriffes über E-Mail. Sie finden ihn auf www.cert.org.

Hinweis***Denial-of-Service-Angriffe über CGI***

Der Artikel »How to Remove Meta-characters from User-Supplied Data in CGI Scripts«, den Sie von www.cert.org erhalten, beschreibt einen Denial-of-Service-Angriff, bei dem Schwächen in CGI-Skripten ausgenutzt werden. Sehr nützlich ist in diesem Zusammenhang auch »The World Wide Web Security FAQ« von www.w3.org.

2.4.8 Weitere Überlegungen zum Filtern von Paketen

Vom Absender geroutete Pakete (so genanntes *Source-Routing*) und Fragmentierung sind in der IPFW-Implementierung keine Probleme, die auf der Ebene des Paketfilters angegangen werden, aber es handelt sich auch dabei um Sicherheitsprobleme im Zusammenhang mit Paketen. Beide werden unter Linux auf der Ebene des Betriebssystems konfiguriert.

Vom Absender geroutete Pakete (Source-Routing)

Source-Routing ist eine selten benutzte Option des Internet Protocols, bei der der Absender eines Paketes bestimmt, welchen Weg es zwischen zwei Maschinen nehmen soll. Normalerweise entscheiden das die Router unterwegs. Wie bei den ICMP-Redirects kann ein Black Hat damit ihr System in die Irre führen, sodass es denkt, es kommuniziere statt mit dem angreifenden Rechner mit einem Computer auf dem eigenen LAN, auf dem Netz des Internet-Providers, oder mit einem aus irgendwelchen anderen Gründen vertrauenswürdigen System.

RedHat 6.0 ist, ebenso wie die meisten modernen Linux-Distributionen, so eingestellt, dass es Pakete mit dieser Option normalerweise ignoriert. Bei älteren Distributionen muss man den Kernel evtl. neu übersetzen, um das zu erreichen. Auf alle Fälle sollten Sie sicherstellen, dass Ihr System Pakete verwirft, bei denen der Absender versucht, über das Routing zu entscheiden. Das Source-Routing hat nur sehr wenige legitime Einsatzmöglichkeiten. Einige Router ignorieren die entsprechende Option übrigens völlig.

Fragmentierung

Bei verschiedenen physikalischen Netzwerktechnologien (z.B. Ethernet, ATM, Token-Ring) existieren unterschiedliche Begrenzungen für die maximal zulässige Länge eines Frames. Wenn ein Paket auf dem Weg vom Absender zum Empfänger von Router zu Router weitergereicht wird, muss ein Gateway es möglicherweise in mehrere kleinere Pakete (oder *Fragmente*) aufteilen, bevor es das Paket in ein Netzwerk mit anderer Technologie weiterleitet. Dabei enthält nur das erste Fragment die üblichen Absender- und Empfänger-Ports. In den folgenden Fragmenten fehlen sie.

Firewalls und Rechner, die für andere Computer IP-Masquerading betreiben, sollten solche Fragmente vor der Weiterleitung zu den ursprünglichen Paketen zusammensetzen. Bei RedHat 6.0 und vielen anderen aktuellen Distributionen ist dieses Feature bereits eingeschaltet. In älteren Versionen sollten Sie die Defragmentierungs-Funktion nachträglich in den Kernel einkompilieren.

2.5 Filtern abgehender Pakete

Wenn Sie den Benutzern und Computern Ihrer unmittelbaren Umgebung halbwegs vertrauen, ist das Filtern bei abgehenden Paketen nicht so kritisch wie bei ankommenden. Auf eingehende Anfragen, die Ihre Firewall herausfiltert, antwortet Ihr Computer gar nicht erst. Trotz allem ist symmetrisches Filtern einfach sicherer. Nebenbei schützt es auch andere Leute – und Sie selbst! – vor Fehlern, die Sie eventuell einmal begehen könnten.

Was kann denn auf Ihrer Seite alles schief laufen? Im schlimmsten Fall gelingt es einem Black Hat, einen Account auf Ihrem System komplett zu knacken. In diesem Fall bietet das Filtern abgehender Pakete zumindest einen gewissen Schutz, wenigstens bis sich der Eindringling *root*-Rechte verschafft und herausfindet, wie man die Firewall ausschaltet.

Wenn Sie abgehende Pakete filtern, verhindern Sie damit auch, dass lokale Pakete Ihrer LAN-Netzwerkdienste ins Internet geraten, wo sie nicht hingehören. Es handelt sich ja nicht nur darum, den Zugriff auf LAN-Dienste von außen zu unterbinden, sondern Sie wollen auch die Informationen Ihres eigenen Systems nicht öffentlich ins Internet hinaussenden. Beispiele wären intern genutzte *dhcpd*-, *timed*-, *routed*- oder *rwhod*-Server. Ebenso sollen *wall*- und *syslogd*-Nachrichten nicht öffentlich sichtbar sein.

Gleichzeitig unterbinden Sie unerwünschte Tätigkeiten, die von Ihrem eigenen System ausgehen. Noch vor einem Jahr vertrat ich in einer Diskussion im Usenet einen etwas großzügigeren Standpunkt, was Filter für abgehende Pakete anging. Jemand schrieb mir daraufhin, ich hätte wohl keine pubertierenden Kinder ...

Schließlich gibt es, zumindest bei meinem eigenen ISP, manchmal auch Schwierigkeiten beim Ausprobieren von Software. Jemand bringt ein Programm von der Arbeit mit nach Hause, und es benimmt sich nicht ganz so, wie es vielleicht sollte.

In diese Ecke gehört auch alte PC-Software, die manchmal die im Internet üblichen Protokolle und reservierten Ports völlig ignoriert. Das ist so, als würde man reine LAN-Software im Internet benutzen.

2.5.1 Filtern nach Absender-IP

Abgehende Pakete nach der Absender-IP zu filtern, ist leicht. Bei einer kleinen Site oder einem einzelnen Computer ist die Absenderadresse normalerweise immer die IP-Adresse Ihres eigenen Computers. Abgehende Pakete mit einer anderen Absenderadresse sind nicht erlaubt.

Wenn Sie Ihre IP-Adresse dynamisch über DHCP erhalten, gibt es während der Adresszuweisung eine kurze Ausnahme. Diese ist spezifisch für DHCP und wird in Kapitel 3 genau beschrieben.

Nicht ganz so leicht haben es Leute mit einem LAN und mehreren öffentlich zugänglichen Servern, von denen jeder über seine eigene statische IP-Adresse verfügt. Die Besonderheiten der Firewall-Konfiguration in einem LAN sind Thema von Kapitel 4.

Wenn die Firewall-Maschine Ihres LANs eine dynamische IP-Adresse hat, sollten Sie abgehende Pakete unbedingt filtern und nur Absender-Adressen zulassen, die mit dieser dynamischen IP übereinstimmen. Dadurch schützen Sie sich vor einigen häufigen Konfigurationsfehlern, bei denen ein anderer Rechner den Eindruck gewinnt, Sie würden Ihre Adresse fälschen oder eine unzulässige IP benutzen. Siehe Kapitel 4.

2.5.2 Filtern nach Empfänger-IP

Wie bei den ankommenden Paketen kann es auch für manche Arten abgehender Pakete sinnvoll sein, sie auf bestimmte Empfänger-Netzwerke oder -Maschinen zu begrenzen. Die Firewall-Regeln geben in diesen Fällen an, welche Empfänger – einzelne IP-Adressen oder ganze Adressbereiche – zulässig sind.

Hierbei gilt es wieder zu unterscheiden, ob ein Paket an einen fremden Server gerichtet ist, auf den Sie zugreifen, oder ob es an einen fremden Client geschickt wird, der einen Dienst Ihres Computers nutzt.

1. Pakete an einen fremden Server: Auf manche Dienste, z.B. Web- oder FTP-Server, greifen Sie überall im Internet zu. Andere Dienste bietet normalerweise nur Ihr eigener Internet-Provider oder ein anderweitig bekannter und vertrauenswürdiger Host an. Zu dieser zweiten Gruppe gehören z.B. POP3 für das Abholen von E-Mail, DHCP für die dynamische Zuweisung Ihrer IP-Adresse und NNTP für das Usenet.
2. Pakete an einen fremden Client: In gleicher Weise gilt, dass manche Clients überall sein dürfen, z.B. wenn jemand Ihren Web-Server benutzt. Andere Dienste bieten Sie vermutlich nur ein paar handverlesenen Rechnern an, z.B.

telnet, ssh und finger. Für diese privaten Dienste sollten Ihre Firewall-Regeln nicht nur eingehende Verbindungen abweisen, sondern gleichzeitig etwaige Antworten Ihres Computers blockieren.

2.5.3 Filtern nach Absender-Port

Explizites Filtern abgehender Pakete nach dem Absenderport an Ihrem Ende der Verbindung ist in zweifacher Hinsicht wirksam, einmal für Ihre Clients, einmal für Ihre Server. Indem Sie die in gesendeten Paketen erlaubten Absender-Ports angeben, zwingen Sie Ihre eigenen Programme zu korrektem Verhalten und schützen andere Leute vor lokalem Traffic, der nicht auf das Internet gehört.

Wenn einer Ihrer Clients eine Verbindung aufbaut, liegt der Absender-Port fast immer im unprivilegierten Bereich. Erzwingen Sie das in den Firewall-Regeln, so schützen Sie Fremde vor potenziellen Fehlern auf Ihrer Seite.

Ihre eigenen Server-Programme senden abgehende Pakete immer vom zugehörigen Service-Port. Wenn die Firewall nur die korrekten Ports als Absender-Ports zulässt, garantiert sie in dieser Hinsicht die richtige Funktion Ihrer Server. Gleichzeitig verhindert sie den externen Zugriff auf private Dienste, die nur für lokale Anwender gedacht sind. Auch werden andere Leute nicht mit Netzwerktraffic belästigt, der eigentlich auf Ihre Computer begrenzt sein sollte.

2.5.4 Filtern nach Empfänger-Port

Client-Software muss sich mit den Servern, auf deren Dienste sie zugreift, immer auf dem korrekten Port verbinden. Sie erzwingen auf der Protokollebene ein sauberes Verhalten Ihrer Programme, indem Sie Verbindungen nur zu den gewünschten Ports zulassen. Gleichzeitig dient eine solche Beschränkung aber noch anderen Zwecken: Zum einen verhindert sie, dass Clients für lokale Dienste versehentlich auf Server im Internet zugreifen. Zum anderen erschwert sie Fehler, Portscans und anderes unerwünschtes Verhalten, das unter Umständen von Ihrem Computer ausgehen könnte.

Wenn Sie eigene Server betreiben, werden diese fast immer an Verbindungen mit unprivilegierten Ports beteiligt sein. Die Firewall-Regeln für Server beschränken abgehende Nachrichten auf unprivilegierte Empfänger-Ports.

2.5.5 Filtern nach TCP-Status-Flags

Wie in den Regeln für ankommende Pakete können Sie auch in denen für abgehende Pakete die Flags benutzen, die den Zustand einer TCP-Verbindung anzeigen. Dabei sollten Sie wieder zwischen Client und Server unterscheiden:

- Wenn einer Ihrer Clients eine Verbindung aufbaut, ist im ersten Paket des dreiteiligen Handshakes das SYN-Flag gesetzt, nicht aber das ACK-Flag. Danach ist nur noch das ACK-Flag gesetzt. In den Regeln sollten Sie also Pakete erlauben, die entweder das SYN- oder das ACK-Flag enthalten.

- Ihre Server hingegen senden Pakete nur als Antwort auf eine Verbindung, die ein fremder Client initiiert hat. Dabei ist immer das ACK-Flag gesetzt. Die Firewall sollte also nur Pakete mit ACK-Flag zulassen.

2.6 Private und öffentliche Dienste im Netz

Am einfachsten laden Sie ungewollte Eindringlinge auf Ihren Computer ein, indem Sie für ein LAN entwickelte Netzwerkdienste für den Zugriff von außen freigeben. Einige Dienste – wenn sie denn angeboten werden müssen – sollten eben niemals die Grenze zwischen Ihrem privaten LAN und dem großen bösen Internet überschreiten. Manche belästigen Ihre Nachbarn, manche verbreiten Informationen, die Sie besser für sich behalten hätten, und manche sind einfach nur riesige Sicherheitslücken.

Einige der frühesten Netzwerkdienste, besonders die BSD-Kommandos für den Zugriff auf fremde Maschinen, wurden für den bequemen Zugriff auf gemeinsame Ressourcen und gemeinsam genutzte Computer in einer vollkommen sicheren Umgebung – dem eigenen Labor – entwickelt. Später entstanden Dienste, die zwar schon für den Einsatz im Internet gedacht waren, aber noch zu einer Zeit, als das Internet im Wesentlichen eine große Gemeinde von Wissenschaftlern und Forschern war. Das Internet war offen und relativ sicher. Erst mit seiner Entwicklung zu einem globalen Netzwerk mit öffentlichem Zugang verlor es seine Vertrauenswürdigkeit.

Viele Netzwerkdienste von Unix liefern den Computern eines lokalen Netzes Informationen über die Benutzer auf dem Rechner, über die laufenden Programme und verwendeten Ressourcen, über den Zustand des Systems, der Netzwerkanbindung etc. Nicht alle diese Info-Dienste sind an und für sich schon Sicherheitslöcher. Ein Black Hat kann mit ihrer Hilfe nicht etwa direkt in Ihr System eindringen. Allerdings erhält er durch sie Informationen über Ihr System und die dort vorhandenen Benutzer, die ihm bei der Suche nach bekannten Sicherheitslücken helfen. Weiterhin verbreiten solche Dienste möglicherweise Daten wie z.B. Name, Adresse, Telefonnummer etc., die Sie vermutlich nicht einfach so an irgendwen weitergeben wollen.

Gefährlichere Dienste erlauben den Zugriff auf gemeinsam genutzte Dateisysteme und Geräte, z.B. einen Drucker oder ein Faxgerät.

Bei manchen Diensten ist es schwer, sie richtig zu konfigurieren, bei manchen ist es schwer, sie sicher zu konfigurieren. Ganze Bücher widmen sich einzelnen komplizierten Unix-Diensten. Solche konkreten Konfigurationsprobleme liegen jenseits dieses Buches.

Ein paar Dienste sind auf einem PC oder in einem kleinen Büro schlicht und ergreifend sinnlos, z.B. wenn es um das Management großer Netzwerke geht, um Routing-Dienste für das Internet, große Datenbankdienste, bestimmte Formen von Verschlüsselung und Authentifizierung usw.

2.6.1 Unsichere lokale Dienste schützen

Der beste Schutz besteht darin, einen Dienst nicht anzubieten. Aber wenn Sie ihn doch brauchen? Nicht alles lässt sich mit einem Paketfilter adäquat schützen. Services mit mehreren Verbindungen, z.B. RealAudio oder ICQ, sowie die UDP-basierten RPC-Dienste sind auf dieser Ebene berichtigt.

Einen einfachen Schutzmechanismus verwirklichen Sie dadurch, dass der ans Netz angeschlossene Computer die problematischen Dienste nicht anbietet. Wenn der Service nicht verfügbar ist, kann jemand aus dem Internet auch nicht auf ihn zugreifen. Auf kleinen Sites hat man leider aber keine größere Menge von Computern zur Verfügung, auf die man die einzelnen Dienste angemessen verteilen könnte. Hier muss man Kompromisse eingehen.

Eine Firewall stellt eine weitere Sicherheitsschranke dar. Eine Paketfilter-Firewall schützt Ihre Systeme auf der Ebene der Ports. Bei vielen der unsicheren lokalen Dienste – v.a. bei TCP-basierten Diensten – wird dadurch die Gefahr eines Zugriffs von außen deutlich reduziert. Gleichzeitig reguliert sie, wer auf welchen Dienst zugreifen darf. Indem sie die Einhaltung der grundlegenden Protokolle erzwingt, trägt sie auch ein Stück weit dazu bei, dass Ihre Programme nur mit den gewünschten Kommunikationspartnern sprechen.

Ein Paketfilter bietet allerdings keine ultimative Sicherheit. Manche Programme benötigen Sicherheitsmaßnahmen auf höheren Ebenen, als ein Paketfilter sie durchführen kann. Manche Programme bedeuten schlicht und ergreifend ein zu hohes Risiko, als dass man sie mit gutem Gewissen auf einer Firewall betreiben dürfte.

2.6.2 Auswahl der Dienste

Letztlich entscheiden Sie selbst darüber, welche Dienste Sie benötigen und wünschen. Der erste Schritt beim Absichern Ihres Systems ist die Entscheidung, welche Dienste und Dämonen auf der Firewall laufen sollen. Jeder Dienst hat sein eigenes Potenzial für Sicherheitsprobleme. Eine gute Faustregel unter Unix ist, nur diejenigen Dienste laufen zu lassen, die Sie brauchen und auch verstehen. Bevor Sie einen Dienst starten, sollten Sie sich wirklich darüber im Klaren sein, was er tut und für wen er es tut – vor allem auf Maschinen mit einer Internet-Anbindung.

Die folgenden Abschnitte enthalten die wichtigsten Netzwerkdienste, die Ihnen unter Red Hat Linux zur Verfügung stehen. Die Dienste sind danach gegliedert, wer sie zu welchem Zeitpunkt aufruft; mit anderen Worten, nach dem Mechanismus, nach dem Unix diese Dienste startet. Diese Aufteilung entspricht der Organisation der einzelnen Dienste vom Standpunkt des Systemadministrators.

Netzwerkdienste werden unter Unix auf drei verschiedene Arten gestartet:

1. Grundlegende Dienste stehen unter der Kontrolle des Runlevel-Managers, der sie während des Bootvorgangs automatisch aufruft, und zwar über Shell-Skripte aus den Verzeichnissen unter `/etc/rc.d`.
2. Für speziellere Dienste ist `inetd` zuständig. Die zugehörigen Server werden erst dann aufgerufen, wenn ein Client sie benötigt. `inetd` wird in der Datei `/etc/inetd.conf` konfiguriert.
3. Einige Netzwerkdienste gibt es nur auf Ihrem speziellen System; für diese sind möglicherweise Ihre eigenen Konfigurationsdateien verantwortlich. Solche Dienste sind normalerweise nicht Teil der Linux-Distribution, sondern Programme, die Sie von irgendwo geholt und manuell installiert haben.

Der Runlevel-Manager

Der Begriff *Runlevel* entstammt dem System V und beschreibt den Zustand des Systems während des Bootvorganges oder des laufenden Betriebs. Red Hat unterscheidet sieben unterschiedliche Runlevel. Die Unterverzeichnisse von `/etc/rc.d` sind mit den einzelnen Runleveln assoziiert und enthalten symbolische Links zu den eigentlichen Konfigurationsdateien in `/etc/rc.d/init.d`.¹ Die Namen der symbolischen Links zeigen Ihnen, ob ein bestimmter Dienst beim Übergang in den zugehörigen Runlevel angehalten (Link beginnt mit *K*) oder gestartet (Link beginnt mit *S*) wird. Die Zahl nach diesem Buchstaben bestimmt die Reihenfolge, in der `init` die Programme aufruft.

Sie werden nicht wirklich alle sieben Runlevel benutzen. Normalerweise arbeitet das System in einem der Runlevel 2, 3 oder 5. Runlevel 3 ist der normale Zustand des Systems. Runlevel 2 ist fast identisch, NFS (Network File System) ist allerdings deaktiviert. Runlevel 5 entspricht Runlevel 3, wobei zusätzlich `xm` (der X Window Display Manager) aktiv ist, der einen grafischen Login ermöglicht. (Auf einer Firewall sollte `xm` im Idealfall inaktiv sein.)

Für Neugierige: Die übrigen Runlevel repräsentieren besondere Zustände des Systems. Runlevel 0 beschreibt die Aufräumarbeiten, die vor dem Ausschalten des Computers durchgeführt werden müssen. In ähnlicher Weise entspricht Runlevel 6 den Aktionen vor einem Neustart. Runlevel 1 definiert, was beim Übergang in den Single-User-Modus geschieht. Der Runlevel 4 wird nicht benutzt. Sie könnten hier einen eigenen Systemzustand definieren.

Bei der Erstinstallation des Computers sowie auch später über den Runlevel-Manager, der sich im `control-panel` verbirgt, entscheiden Sie, welche Dienste

1. SuSE weicht in dieser Hinsicht vom SysV-Standard ab. Man betrachtet die Runlevel-Files hier nicht als Konfigurationsdateien, sondern als Binaries; dementsprechend finden Sie die zugehörigen Verzeichnisse unter SuSE in `/sbin/init.d`. Immerhin existieren symbolische Links nach `/etc`, sodass der Zugriff über die gewohnten Pfade möglich ist. Wundern Sie sich jedoch nicht, wenn in Ihrem Backup von `/etc` diese entscheidenden Dateien fehlen.

beim Booten automatisch starten. Die hierbei verwendeten Tools sind grafische Interfaces zur Manipulation der Dateien in `/etc/rc.d`.

Technischer Hintergrund: Wenn das System bootet, liest der »Hauptprozess« `init` die Konfigurationsdatei `/etc/inittab`. Dort stehen Anweisungen, verschiedene Skripts in `/etc/rc.d` auszuführen, u.a. `rc.sysinit` und `rc.` `/etc/rc.d/rc.sysinit` wird beim Booten einmalig ausgeführt; es initialisiert das System, lädt notwendige Module, überprüft die Dateisysteme usw. `/etc/rc.d/rc` wird jedesmal aufgerufen, wenn sich der Runlevel ändert. Es hält zunächst alle Dienste an, die im neuen Runlevel nicht mehr benötigt werden, und startet anschließend etwa definierte neue Services. `/etc/inittab` legt auch den voreingestellten Runlevel fest, z.B. `id:3:initdefault`. Für eine Firewall ist Runlevel 2 die beste Wahl.

Die folgende Tabelle 2.1 beschreibt die wichtigsten Netzwerkdienste, die direkt von `init` gestartet werden.

	amd
Beschreibung	NFS Automount-Dämon, amd
Kommentar	NFS wurde für lokale Netze entwickelt. Bei der Verwendung über das Internet drohen zahlreiche Sicherheitsprobleme. Erlauben Sie vom Internet keinen Zugang zu NFS-Dateisystemen. Aktivieren Sie NFS nicht einmal auf Ihrer Firewall. NFS besteht aus mehreren Dämonen; im Runlevel-Editor von Red Hat heißen die zugehörigen Skripts <code>amd</code> , <code>nfs</code> und <code>nfsfs</code> .
Empfehlung	amd hat auf einer Firewall nichts verloren.
	arpwatch
Beschreibung	Der <code>arpwatch</code> -Dämon erstellt eine Datenbank der Zuordnungen zwischen den Ethernet- und IP-Adressen, die er auf einem LAN-Interface sieht.
Kommentar	Richtig konfiguriert, ist der <code>arpwatch</code> -Dämon selber kein Sicherheitsproblem. Er benutzt ein privates Unix-Domain-Socket. Allerdings schaltet er Ihr externes Netzwerk-Interface in den <i>Promiscuous-Mode</i> , d.h. dass es auch solche Pakete untersuchen kann, die nicht an Ihren Computer gerichtet sind. Eine Netzwerkkarte in diesem Modus ist normalerweise ein sicheres Zeichen für einen erfolgten Einbruch. Wenn die Netzwerkkarte Ihres Computers schon im Normalzustand im <i>Promiscuous-Mode</i> ist, verlieren Sie dieses Indiz für einen Einbruch. Ohnehin sagt <code>arpwatch</code> Ihnen nichts über Ihr LAN, das Sie nicht schon wüssten.
Empfehlung	Deaktivieren Sie <code>arpwatch</code> auf einer Firewall.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind

autofs	
Beschreibung	Automatisches mounten von Netzwerkdateisystemen über automount
Kommentar	NFS ist ein RPC-basierter Dienst. NFS ist, wie auch jeder andere Service, der portmap benutzt, ein massives potenzielles Sicherheitsproblem und sollte aus dem Internet nicht zugänglich sein. Im Idealfall sollten diese Server nicht auf einer Firewall laufen. Zudem verlässt sich automount auf den Network Information Service NIS, sofern dieser verfügbar ist. NIS ist ein weiterer LAN-Dienst, der auf einer Firewall nichts verloren hat.
Empfehlung	Deaktivieren Sie automount auf der Firewall.
bootparamd	
Beschreibung	Boot Parameter Server
Kommentar	Der bootparamd-Dämon versorgt Workstations ohne eigenes Bootmedium mit Informationen, die sie für den Bootvorgang benötigen. In einem privaten LAN oder einer kleinen Firma existieren mit hoher Wahrscheinlichkeit keine solchen »diskless« Workstations. So oder so, dieser Server sollte auf einer Firewall nicht ausgeführt werden.
Empfehlung	Deaktivieren Sie bootparamd.
dhcpcd	
Beschreibung	Lokaler DHCP-Server
Kommentar	dhcpcd weist anderen Client-Computern dynamische IP-Adressen zu. ISPs oder LANs größerer Firmen benötigen so etwas. Wenn umgekehrt Ihr Internet-Provider Ihnen eine dynamische IP-Adresse zuteilt, dann wird er es nicht besonders gerne sehen, wenn Sie selbst einen DHCP-Server aufsetzen, und Ihren Zugang mit ein paar unfreundlichen Worten deaktivieren. Sicherlich, manche Leute brauchen so etwas, aber dabei ist bei der Konfiguration sowohl des Servers selbst als auch der Firewall Sorgfalt geboten.
Empfehlung	Aktivieren Sie dhcpcd nur, wenn Sie ihn wirklich benötigen und erst, wenn Sie seine Konfiguration gut verstehen. Selbst dann ist eine vorgeschaltete Firewall obligatorisch.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

gated	
Beschreibung	Gateway Routing Daemon
Kommentar	Der <code>gated</code> -Dämon ist für die Routing-Protokolle im Netz verantwortlich. In einem kleinen LAN verwalten Sie das Routing besser und sicherer mit statischen IP-Adressen. Eine Firewall sollte in jedem Fall nur statisches Routing einsetzen. Wenn Sie allerdings einen Routing-Dämon brauchen, ist <code>gated</code> eine bessere Wahl als der ältere und unsichere <code>routed</code> : <code>gated</code> unterstützt das moderne OSPF-Routing-Protokoll, <code>routed</code> lediglich das ältere RIP. Sowohl RIP als auch OSPF sind Protokolle für <i>lokale</i> Netze.
Empfehlung	Firewall-Computer sollten nicht zugleich als dynamische Router agieren; <code>gated</code> gehört nicht auf eine Firewall.
httpd	
Beschreibung	Apache Web-Server
Kommentar	Die detaillierte Behandlung der Absicherung eines Web-Servers würde den Rahmen dieses Buches sprengen. Grundsätzlich ist zu sagen: Begrenzen Sie den Zugriff auf die Teile Ihres Dateisystems, die Sie wirklich veröffentlichen wollen. Lassen Sie die Server als unprivilegierte Benutzer laufen. Vermeiden Sie CGI-Skripten vollständig, sofern Sie sich nicht über die massiven Sicherheitsrisiken, die damit einhergehen können, im Klaren sind. Weil CGI-Skripten beliebige weitere Programme auf Ihrem Computer ausführen dürfen, müssen sie alle Benutzereingaben sorgfältig überprüfen, als unprivilegierter Benutzer laufen und immer vollständige Pfade angeben, wenn sie andere Programme oder Skripten ausführen.
Empfehlung	Wenn Sie eine Web-Site betreiben wollen, aktivieren Sie den <code>httpd</code> erst später, nachdem Sie sich eingehend mit der Server-Dokumentation und den Kommentaren in den mitgelieferten Konfigurationsdateien beschäftigt haben.
inet	
Beschreibung	Der <code>inetd</code> spielt für viele Netzwerkdienste eine zentrale Rolle. Dabei muss nicht mehr mindestens ein Dämon kontinuierlich im Speicher bleiben, egal, ob der zugehörige Dienst benutzt wird oder nicht. <code>inetd</code> ersetzt diese Dämonen durch ein einzelnes Programm, nämlich sich selbst. <code>inetd</code> wartet auf eine eingehende Verbindung, entscheidet, zu welchem Dienst diese gehört, ruft möglicherweise ein Hilfsprogramm für die Zugriffskontrolle auf und stellt schließlich den Client zum richtigen Server-Programm durch.
Kommentar	Sie benötigen <code>inetd</code> , wenn Sie einige beliebte Dienste wie <code>ftp</code> oder <code>telnet</code> lokal einsetzen oder öffentlich anbieten wollen.
Empfehlung	Aktivieren Sie <code>inetd</code> .

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

innd	
Beschreibung	Lokaler Usenet-News-Server
Kommentar	Durch eine Firewall geschützt, ist ein News-Server kein allzu großes Sicherheitsrisiko. Auf den meisten kleinen Sites besteht allerdings kein Bedarf hierfür. Zudem ist die Einrichtung von <code>innd</code> nicht ganz einfach. Wenn Sie einen eigenen Server brauchen, denken Sie daran, den Zugriff ausschließlich von einzelnen vertrauenswürdigen Sites zu erlauben.
Empfehlung	Deaktivieren Sie <code>innd</code> .
linuxconf	
Beschreibung	Erlaubt die Konfiguration eines RedHat-Servers über einen Web-Browser.
Kommentar	<code>linuxconf</code> ist dem TCP-Port 98 zugeordnet. In der Voreinstellung reagiert es ausschließlich auf Anfragen von Loopback, also dem eigenen Computer. In diesem Fall ist der Dienst ungefährlich. Ändern Sie aber keinesfalls die Konfiguration so, dass Zugriffe von außen möglich sind.
Empfehlung	keine
lpd	
Beschreibung	Druck-Server
Kommentar	Auf den ersten Blick wirken Drucker wie rein lokale Geräte. Unix behandelt sie jedoch automatisch als Netzwerkgeräte. Wenn ein Drucker angeschlossen ist, sollten Sie den Zugriff über das Netz sowohl in der Druckerkonfiguration als auch über die Firewall sperren.
Empfehlung	Wenn Sie einen Drucker mit mehreren Unix-Maschinen teilen, kommen Sie um <code>lpd</code> wohl kaum herum.
mars-nwe	
Beschreibung	Ein File- und Print-Server für Novell NetWare-kompatible Windows-Clients auf dem lokalen Netz.
Kommentar	File- und Printserver für das lokale Netz sind auf einer Firewall deplatziert. Für Netware-Unterstützung müssten Sie die IPX-Netzwerkschicht, die SPX-Transportschicht und das NCP-Dateisystem in den Linux-Kernel mit einkompilieren. Wie heißt es im Hilfetext zur Kernelkonfiguration so schön: »Wenn Sie nicht verstehen, was damit gemeint ist, schalten Sie es auch nicht ein.«
Empfehlung	Deaktivieren Sie <code>mars-nwe</code> auf der Firewall.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runlevels verfügbar sind (Forts.)

mcserv	
Beschreibung	Der File-Server zum Midnight Commander
Kommentar	mcserv regelt den Zugriff zum Netzwerkdateisystem von Midnight Commander. Dabei handelt es sich um einen unsicheren Dateiserver, der für den Betrieb im lokalen Netz entwickelt wurde. Erlauben Sie keinen externen Zugang aus dem Internet. mcserv ist ein RPC-basierter UDP-Dienst und benutzt den portmap-Dämon.
Empfehlung	Deaktivieren Sie mcserv auf der Firewall.
named	
Beschreibung	Bei named handelt es sich um einen DNS-Server, der zwischen symbolischen Maschinennamen und den zugehörigen numerischen IP-Adressen übersetzt. Der entsprechende Client – Resolver genannt – ist kein separates Programm, sondern ein Bestandteil der Netzwerk-Bibliotheken, die in Ihre Programme hineinkompiliert werden.
Kommentar	Vermutlich betreibt Ihr Internet-Provider bereits einen Nameserver. Ein lokaler Nameserver kann die Performance im Netzwerk verbessern. Richtige Konfiguration vorausgesetzt, stellen einfache DNS-Server kein wesentliches Sicherheitsrisiko dar.
Empfehlung	Wenn Sie den Server und seine Konfiguration – die ziemlich kompliziert sein kann – verstehen, aktivieren Sie named. Bis dahin benutzen Sie die Nameserver Ihres Providers.
netfs	
Beschreibung	Mountet Dateisysteme vom Netzwerk über NFS, Samba oder NetWare.
Kommentar	netfs ist kein Dämon, sondern ein Shell-Skript, das einmalig aufgerufen wird und Netzwerkdateisysteme auf Ihrem Rechner mountet. Trotzdem gilt auch hier die Faustregel, dass man vernetzte Dateisysteme auf einer Firewall nicht benutzen sollte.
Empfehlung	Deaktivieren Sie netfs auf der Firewall.
network	
Beschreibung	Das Skript network aktiviert während des Bootvorganges die konfigurierten Netzwerk-Interfaces. Es ist kein Server.
Kommentar	Sie benötigen dieses Skript.
Empfehlung	Aktivieren Sie network.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

nfs	
Beschreibung	Aktiviert die NFS-Dienste.
Kommentar	NFS wurde als reiner LAN-Dienst entwickelt. Beim Einsatz im Internet drohen zahlreiche Sicherheitsrisiken. Verboten Sie den Internet-Zugang zu NFS-Dateisystemen; benutzen Sie NFS nicht auf der Firewall. NFS besteht aus mehreren Dämonen; im Runlevel-Editor von RedHat heißen die zugehörigen Init-Skripte <code>amd</code> , <code>nfs</code> und <code>nfsfs</code> .
Empfehlung	Deaktivieren Sie <code>nfs</code> auf der Firewall.
nscd	
Beschreibung	<i>Name Switch Cache Daemon</i>
Kommentar	<code>nscd</code> ist ein Hilfs-Service zu NIS, der die Passwörter und die Gruppenzugehörigkeit der Benutzer zwischenspeichert. NIS ist ein inhärent unsicherer LAN-Dienst, der auf einer Firewall-Maschine nichts zu suchen hat.
Empfehlung	Deaktivieren Sie <code>nscd</code> auf der Firewall.
portmap	
Beschreibung	<code>portmap</code> ist der RPC-Portmap-Manager.
Kommentar	Der RPC- <code>portmap</code> besitzt eine gewisse Ähnlichkeit zu <code>inetd</code> . Er verwaltet Verbindungen zu RPC-basierten Diensten wie NFS und NIS. Wenn Sie diese Dienste nicht benutzen, deaktivieren Sie <code>portmap</code> . Ansonsten sperren Sie unbedingt in Ihrer Firewall den Zugriff auf <code>portmap</code> von außen. Moderne Versionen von <code>portmap</code> berücksichtigen die Zugriffsregeln aus <code>/etc/hosts.allow</code> und <code>/etc/hosts.deny</code> .
Empfehlung	Deaktivieren Sie <code>portmap</code> auf der Firewall.
postgresql	
Beschreibung	SQL-Datenbank-Server
Kommentar	<code>postgresql</code> ist ein TCP-Dienst auf Port 5432. Der Server, <code>postmaster</code> , lässt sich so konfigurieren, dass er entweder Internet-Domain-Sockets oder lokale Unix-Domain-Sockets benutzt. Als ein Service, der Zugriff auf lokale Dateien erlaubt, sollte er ohne ganz besondere Vorsichtsmaßnahmen nicht von einer Firewall angeboten werden. Nähere Informationen bieten die Manual-Pages zu <code>postmaster(1)</code> , <code>postgres(1)</code> und <code>psql(1)</code> sowie das PostgreSQL-HOWTO.
Empfehlung	Deaktivieren Sie <code>postgresql</code> auf der Firewall.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

routed	
Beschreibung	Der <code>routed</code> -Dämon ermöglicht dynamische Updates für die Routing-Tabellen des Kernels.
Kommentar	Sowohl <code>routed</code> als auch <code>gated</code> sind massive Sicherheitslücken, <code>routed</code> in noch höherem Ausmaß als <code>gated</code> . Es ist äußerst unwahrscheinlich, dass Sie die Routing-Tabelle Ihrer Firewall mit RIP – einem Dienst Ihres Providers – verwalten müssen. Verwenden Sie stattdessen statische IP-Adressen für die Firewall.
Empfehlung	Deaktivieren Sie <code>routed</code> .
rstatd	
Beschreibung	Der <code>rstatd</code> -Dämon liefert anderen Maschinen auf dem Netzwerk Systeminformationen.
Kommentar	Informationen über den Status Ihres Systems sollten Sie anderen Computern im Internet nicht mitteilen.
Empfehlung	Deaktivieren Sie <code>rstatd</code> auf der Firewall.
ruserd	
Beschreibung	Der RPC-basierte »Remote User Locator Service« liefert Informationen über die Benutzer, die gerade auf den Computern eines Netzwerkes angemeldet sind.
Kommentar	Eine kleine Site wird diesen Dienst kaum verwenden. Außerdem benötigt der Dämon <code>rpc.rusersd</code> RPC-Dienste, die auf einer Firewall nicht eingesetzt werden sollten.
Empfehlung	Deaktivieren Sie <code>ruserd</code> auf der Firewall.
rwallld	
Beschreibung	Der RPC-basierte Dämon <code>rpc.rwallld</code> ermöglicht es den Benutzern, Nachrichten an jeden anderen zu verschicken, der gerade auf irgendeinem Computer des Netzwerks eingeloggt ist.
Kommentar	Eine kleine Site wird diesen Dienst kaum einsetzen. Außerdem benötigt der Dämon <code>rpc.rwallld</code> RPC-Dienste, die auf einer Firewall nicht eingesetzt werden sollten.
Empfehlung	Deaktivieren Sie <code>rwallld</code> auf der Firewall.
rwhod	
Beschreibung	Der Dämon <code>rwhod</code> ermöglicht die LAN-Dienste <code>rwho</code> und <code>ruptime</code> , d.h. er liefert Informationen, wer gerade eingeloggt ist, was er tut, welche Computer überhaupt laufen und an das Netzwerk angeschlossen sind etc.
Kommentar	Auf einer kleinen Site wird solch ein Dienst wohl kaum gebraucht.
Empfehlung	Deaktivieren Sie <code>rwhod</code> auf der Firewall.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

sendmail	
Beschreibung	E-Mail-Dienst
Kommentar	Wenn Sie Ihren eigenen E-Mail-Dienst betreiben, benötigen Sie <code>sendmail</code> (oder einen anderen Mail-Server). Richtig konfiguriert, ist <code>sendmail</code> heutzutage relativ sicher. Trotzdem bleibt es ein häufiges Angriffsziel für Einbruchsversuche. Sobald Probleme entdeckt und behoben werden, erscheinen Sicherheits-Updates. SMTP-Server waren in der Vergangenheit immer wieder für Sicherheitslücken verantwortlich, womit in diesem Zusammenhang nicht nur der Einbruch ins System gemeint ist, sondern auch die Verwendung eines Computers zum Versand von Werbe-E-Mails (Spam). In die Sicherheit der jüngsten <code>sendmail</code>-Versionen ist viel Arbeit investiert worden. Die mit den meisten heutigen Linux-Distributionen gelieferte Konfiguration ist zumindest im Hinblick auf Spam recht sicher.
Empfehlung	Aktivieren Sie <code>sendmail</code> , wenn Sie – unabhängig von Ihrem Internet-Provider – eigene E-Mail-Dienste wünschen. Falls Sie alles über Ihren Provider empfangen und verschicken, deaktivieren Sie <code>sendmail</code> .
smb	
Beschreibung	Gemeinsame Nutzung von Dateisystemen und Druckern über Samba
Kommentar	Die gemeinsame Nutzung von Dateisystemen und Druckern fällt in den Bereich der reinen LAN-Dienste und sollte auf einer Firewall nicht angeboten werden. Samba sollte unter keinen Umständen vom Internet aus erreichbar sein.
Empfehlung	Deaktivieren Sie <code>smb</code> auf einer Firewall.
snmpd	
Beschreibung	Netzwerkadministration über SNMP (Simple Network Management Protocol)
Kommentar	SNMP dient der Verwaltung eines LANs. Als UDP-Dienst sollte <code>snmpd</code> aus Sicherheitsgründen nicht auf einer Firewall laufen. Auf einem kleinen Netzwerk ist es mit größter Wahrscheinlichkeit ohnehin unnötig. Falls Sie es wirklich benötigen, betrachten Sie es als gefährlichen Dienst und sperren Sie SNMP-Pakete zwischen dem Internet und Ihrem eigenen Netzwerk in beiden Richtungen. Außenstehende sollen Ihr Netzwerk nicht administrieren, und in gleicher Weise werden Fremde nicht sehr glücklich sein, wenn sie von Ihnen SNMP-Pakete erhalten.
Empfehlung	Deaktivieren Sie <code>snmpd</code> .

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

squid	
Beschreibung	Der <i>Squid Internet Object Cache</i> dient als lokaler HTTP-Proxy und Web-Cache. Vor Gebrauch müssen Sie ihn konfigurieren.
Kommentar	Bei angemessener Konfiguration drohen von squid keine besonderen Gefahren.
Empfehlung	Wenn Sie einen Cache für Web-Seiten wünschen und dafür nicht Apache benutzen, aktivieren Sie squid.
syslog	
Beschreibung	Das Init-Skript syslog startet die Dämonen syslogd und klogd. Diese übernehmen Status- und Fehlermeldungen vom syslog(2) und verteilen sie auf die entsprechenden Dateien oder Programme oder geben sie an andere Computer im LAN weiter.
Kommentar	Das syslog ist essentiell.
Empfehlung	Aktivieren Sie syslog.
xfs	
Beschreibung	Der <i>X Window Font Server</i> versorgt X-Server auf dem eigenen Computer und auf fremden Maschinen mit Bildschirm-Fonts.
Kommentar	xfs kann zwar so konfiguriert werden, dass es Verbindungsanfragen über ein TCP-Socket entgegennimmt, aber das ist nicht die Voreinstellung. Normalerweise benutzt es nur ein privates Unix-Domain-Socket. Damit ist es in der Regel kein besonderes Sicherheitsrisiko. Ihr X-Server benötigt xfs.
Empfehlung	Trotzdem sollten Sie xfs auf einer Firewall deaktivieren, wenn irgend möglich.
xntpd	
Beschreibung	Ein Server, der anderen Computern die Synchronisation mit der aktuellen Systemzeit erlaubt.
Kommentar	Hinter einer Firewall ist xntpd kein Sicherheitsproblem. Manche Leute setzen die Systemzeit regelmäßig via ntpd oder ntpdate. Der xntpd-Server gibt die Zeit dann an andere Rechner im lokalen Netz weiter.
Empfehlung	Falls Sie die Systemzeit der Computer in Ihrem LAN synchronisieren möchten, lesen Sie die Dokumentation zu xntpd und aktivieren danach den Dämon.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

ypbind	
Beschreibung	Macht einen Computer zu einem NIS-Client
Kommentar	NIS umfasst die Boot-Skripten <code>ypbind</code> , <code>ypasswdd</code> und <code>ypserv</code> . NIS ist ein LAN-Service für die zentralisierte Verwaltung von Netzwerk-, Benutzer- und Host-Informationen. Auf einem kleinen Netzwerk ist es vermutlich überflüssig. Falls Sie sich für seinen Einsatz entscheiden, betrachten Sie NIS als »gefährlich« und blockieren Sie den Austausch von NIS- und RPC-Daten zwischen Ihrem LAN und dem Internet.
Empfehlung	Deaktivieren Sie <code>ypbind</code> auf einer Firewall.
ypasswdd	
Beschreibung	NIS-Passwort-Server
Kommentar	NIS umfasst die Boot-Skripten <code>ypbind</code> , <code>ypasswdd</code> und <code>ypserv</code> . NIS ist ein LAN-Service für die zentralisierte Verwaltung von Netzwerk-, Benutzer- und Host-Informationen. Auf einem kleinen Netzwerk ist es vermutlich überflüssig. Falls Sie sich für seinen Einsatz entscheiden, betrachten Sie NIS als »gefährlich« und blockieren Sie den Austausch von NIS- und RPC-Daten zwischen Ihrem LAN und dem Internet.
Empfehlung	Deaktivieren Sie <code>ypasswdd</code> auf einer Firewall.
ypserv	
Beschreibung	NIS-Master-Server
Kommentar	NIS umfasst die Boot-Skripten <code>ypbind</code> , <code>ypasswdd</code> und <code>ypserv</code> . NIS ist ein LAN-Service für die zentralisierte Verwaltung von Netzwerk-, Benutzer- und Host-Informationen. Auf einem kleinen Netzwerk ist es vermutlich überflüssig. Falls Sie sich für seinen Einsatz entscheiden, betrachten Sie NIS als »gefährlich« und blockieren Sie den Austausch von NIS- und RPC-Daten zwischen Ihrem LAN und dem Internet.
Empfehlung	Deaktivieren Sie <code>ypserv</code> auf einer Firewall.

Tab. 2.1: Netzwerkdienste, die auf den einzelnen Runleveln verfügbar sind (Forts.)

Hinweis

Weitere Informationen zur Sicherheit von Web-Servern

Informationen zur Sicherheit von Web-Servern finden Sie u.a. auf der Webseite von Apache, www.apache.org, in »Apache Server for Dummies« von Ken Coar, und im Artikel »How to Remove Meta-characters from User-Supplied Data in CGI Scripts« von www.cert.org.

Hinweis

Weitere Informationen zu DNS

Informationen zu DNS finden Sie u.a. im O'Reilly-Buch »DNS und BIND« von Paul Albitz und Cricket Liu, im DNS-HOWTO von Nicolai Langfeldt – auf Ihrem Computer vermutlich irgendwo unter `/usr/doc/` abgelegt – sowie in den Manual-Pages zu `named(8)`, `resolver(5)` und `hostname(7)`.

Hinweis

Häufige Sicherheitslücken

`portmap` gehört zu den am häufigsten ausgenutzten Sicherheitslücken eines Unix-Systems.

Von inetd verwaltete Dienste

Nachdem Sie jetzt wissen, wie der Runlevel-Manager funktioniert und welche Systemdienste beim Booten automatisch gestartet werden, gehen wir zu den Netzwerkdiensten über. Diese können sowohl lokal als auch öffentlich sein. Die meisten von ihnen werden durch `inetd` gestartet, den man deshalb auch als »Superserver« bezeichnet. In der Konfigurationsdatei `/etc/inetd.conf` geben Sie an, welche Dienste `inetd` verwalten soll.

Fast alle diese Dienste wurden für lokale Netze entworfen und sollten auf einer Firewall deaktiviert sein. Dabei ist aber zu beachten, dass sich die voreingestellte `inetd`-Konfiguration von Distribution zu Distribution und auch innerhalb der einzelnen Versionen einer Distribution häufig ändert. So war z.B. die Konfiguration von RedHat 6.0 sehr unsicher und für ein ans Internet angeschlossenes System nicht zu gebrauchen. Bei RedHat 6.2 waren sich die Entwickler der Sicherheitsrisiken in höherem Ausmaß bewusst. Indem Sie auf die Installation von `inetd` sowie der diversen `server`-Pakete verzichten, erhalten Sie zumindest in diesem Bereich einen relativ gut abgesicherten Computer. Achtung: Gegebenfalls wird auf Ihrem System in diesem Fall gar keine Datei `/etc/inetd.conf` mehr existieren.

Typische Services aus `inetd.conf` werden in den folgenden Abschnitten beschrieben, und zwar in der Reihenfolge, in der sie meistens aufgeführt sind.

Dienste für Netzwerk-Tests

Die erste Gruppe von Diensten ist intern im `inetd`-Dämon implementiert und dient der Durchführung von lokalen Netzwerk-Tests und der Fehlersuche. Mit größter Wahrscheinlichkeit wird kein einziger Leser dieses Buches sie jemals benötigen:

```
#echo          stream  tcp      nowait  root    internal
#echo          dgram   udp      wait    root    internal
#discard       stream  tcp      nowait  root    internal
```


#discard	dgram	udp	wait	root	internal
#daytime	stream	tcp	nowait	root	internal
#daytime	dgram	udp	wait	root	internal
#chargen	stream	tcp	nowait	root	internal
#chargen	dgram	udp	wait	root	internal
#time	stream	tcp	nowait	root	internal
#time	dgram	udp	wait	root	internal

Wenn Sie sich diese Services ansehen, wirken sie auf den ersten Blick ziemlich harmlos. Dennoch waren z.B. die UDP-Formen von `discard` und `chargen` an einem ziemlich weit verbreiteten Denial-of-Service-Angriff beteiligt. Weiter oben, in Abschnitt *UDP-Flooding*, finden Sie einen Hinweis auf das CERT-Advisory, das diesen Angriff beschreibt.

Standarddienste

`inetd` ruft auch eine Reihe von Unix-Standarddiensten auf, namentlich die `ftp`- und `telnet`-Server:

- `ftp` ist eine der verbreitetsten Methoden, Dateien im Internet zu veröffentlichen. Allerdings tummeln sich in den `ftp`-Servern zahlreiche Sicherheitslücken, und entsprechend oft wird in diesen Dienst bei schlechter Konfiguration eingebrochen. Die korrekte Konfiguration ist kompliziert und verlangt viel Sorgfalt.

```
#ftp      stream tcp      nowait root    /usr/sbin/tcpd  in.ftpd -l -a
```

Wenn Sie einen öffentlichen FTP-Server aufsetzen wollen, lesen Sie zuvor die einschlägige Dokumentation sowie einige der Aufsätze von CERT, in denen die Sicherheitsprobleme von FTP beschrieben werden. Wenn Sie anonymen Benutzern nur ein paar Dateien zur Verfügung stellen wollen, sind Sie vermutlich mit einem Web-Server besser beraten. Anonymes FTP ist im Hinblick auf Sicherheitslücken viel gefährdeter als FTP mit regulären Benutzernamen und Passwörtern. Ich würde an Ihrer Stelle das Paket für anonymes FTP gar nicht erst installieren. Aber auch reguläres FTP sollte im Idealfall nicht von einer Firewall aus angeboten werden.

Wenn Sie FTP nur im LAN anbieten wollen, aktivieren Sie den Dienst in `/etc/inetd.conf`. Sperren Sie den Zugriff von außen in der `ftp`-Konfiguration, über die `tcp_wrapper` sowie in der Firewall selbst. Dokumentation zu den `tcp_wrappern` und verwandten Themen befindet sich in den Man-Pages `tcpd(8)`, `hosts_access(5)`, `inetd.conf(5)`, `hosts_options(5)` und `syslog.conf(5)`.

- Mit `telnet` kann man sich über das Internet und lokal in andere Systeme einloggen, unabhängig davon, ob man Unix oder ein anderes Betriebssystem einsetzt. In einem LAN werden Sie diesen Dienst vermutlich einschalten wollen, außer, Sie können `ssh` auf allen Computern benutzen. Wenn Sie auf Ihren Computer aus dem Internet zugreifen wollen, benötigen Sie `telnet` ebenfalls, sofern nicht ein sicherer Dienst wie `ssh` zur Verfügung steht.

```
#telnet   stream tcp      nowait root    /usr/sbin/tcpd  in.telnetd
```

Hinweis**Sicherheitsprobleme mit FTP**

Sicherheitsprobleme mit FTP werden u.a. in den folgenden Artikeln von www.cert.org besprochen: »Anonymous FTP Configuration Guidelines«, »Anonymous FTP Abuses« sowie »Problems with the FTP PORT Command«.

telnet über das Internet ist unsicher, weil dabei alle Informationen – einschließlich Benutzername und Passwort – im Klartext übertragen werden. Mit einem Packet-Sniffer kann man diese Informationen mithören.

Wenn Sie telnet zwar intern, aber nicht über das Internet benötigen, beschränken Sie den Zugriff über die Firewall, über die tcp_wrapper und in /etc/security/access.conf.

- gopher gehört ebenfalls zu den Standarddiensten. Er wird heute nur noch von Wenigen benutzt und ist im Wesentlichen durch Web-Server und entsprechende Suchmaschinen ersetzt worden, sodass die meisten Systeme ihn nicht mehr anbieten. Seit RedHat 6.0 erscheint gopher nicht mehr in /etc/inetd.conf.

```
#gopher stream tcp nowait root /usr/sbin/tcpd gn
```

BSD-Dienste für Fernzugriff

Die BSD-Dienste wurden als Teil der Berkeley-Unix-Distribution entwickelt, um den Zugriff auf eigene Accounts auf anderen Maschinen im LAN zu erleichtern.

- Die »R-Dienste« umfassen die Programme rsh und rlogin sowie die Library-Funktion rexec. In /etc/inetd.conf heißen sie shell, login und exec. Diese LAN-Dienste erleichtern einem Benutzer den Zugriff auf eigene Accounts innerhalb des LANS: er muss nicht jedesmal ein Passwort eintippen, wenn er die Maschine wechselt.

Diese Dienste haben auf einer Firewall nichts verloren. Wenn Sie nicht auf die genannten Programme verzichten können oder wollen, deaktivieren Sie zumindest – über die Firewall-Regeln sowie die tcp_wrapper – jeden Zugriff von außen. Lassen Sie *niemals* Leute aus dem Internet auf diese drei Dienste zugreifen, oder man wird Ihr System in kürzester Zeit knacken.

```
#shell stream tcp nowait root /usr/sbin/tcpd in.rshd
#login stream tcp nowait root /usr/sbin/tcpd in.rlogind
#exec stream tcp nowait root /usr/sbin/tcpd in.rexecd
```

- Die talk-Dienste sind an und für sich nicht unsicher, aber wollen Sie wirklich, dass jemand im Internet auf Ihr Terminal schreiben kann oder auch nur erfährt, welche Accounts auf Ihrem System existieren? Eine kleine Site kommt intern

sehr gut ohne talk und Co. aus. IRC, Instant Messenger und ähnliche haben die talk-Dienste weitgehend abgelöst.

```
#comsat dgram udp wait root /usr/sbin/tcpd in.comsat
#talk dgram udp wait root /usr/sbin/tcpd in.talkd
#ntalk dgram udp wait root /usr/sbin/tcpd in.ntalkd
#dtalk stream tcp wait nobody /usr/sbin/tcpd in.dtalkd
```

E-Mail-Server

Hinweis

Eindringlinge in sunrpc und mountd

Beschreibungen von erfolgten Einbrüchen:

- *sunrpc (111/udp/tcp) und mountd (635/udp)* – www.cert.org/advisories/CA-98.12.mountd.html
- *imap (143/tcp)* – www.cert.org/advisories/CA-98.09.imapd.html
- *pop-3 (110/tcp)* – www.cert.org/advisories/CA-98.08.qpopper_vul.html

Die E-Mail-Server POP und IMAP stellen bei ungenügender Konfiguration notorische Sicherheitslücken dar. Zusammen mit portmap und mountd gehören sie zu den am häufigsten missbrauchten Servern.

- pop-2 ist heute weitgehend durch das neuere pop-3-Protokoll abgelöst worden. Es gibt kaum einen guten Grund, diesen Service noch anzubieten.

```
#pop-2 stream tcp nowait root /usr/sbin/tcpd ipop2d
```

- pop-3 müssen Sie möglicherweise anbieten, in der Regel wird das aber nicht der Fall sein. Manche Leute beschränken den POP-Zugriff auf wenige User, oder benutzen POP nur lokal, um E-Mail von einem zentralen Server abzuholen. Auf alle Fälle sollten Sie, wenn Sie einen POP-Server aktivieren, den Zugriff darauf mit Hilfe der Firewall, der tcp_wrapper und der POP-Konfigurationsdateien einschränken.

```
#pop-3 stream tcp nowait root /usr/sbin/tcpd ipop3d
```

- Wie pop-3 benötigen Sie eventuell auch imap.

```
#imap stream tcp nowait root /usr/sbin/tcpd imapd
```

uucp

Deaktivieren Sie uucp! UUCP war einmal eine sehr verbreitete Methode, mit der man Dateien zwischen einzelnen Computern austauschte. Manche News-Server erhalten ihre News weiterhin über UUCP, und ein paar Leute benutzen es auch sonst noch – unter strengen Sicherheitsvorkehrungen – als Alternative zu FTP, aber uucp stirbt eindeutig aus.

```
#uucp      stream tcp      nowait uucp   /usr/sbin/tcpd  /usr/lib/uucp/  
uucico -l
```

Boot-Dienste

Die Boot-Dienste vergeben im LAN IP-Adressen und helfen Maschinen ohne eigene Laufwerke beim Booten.

- **tftp:** Halten Sie sich an das, was der Kommentar in `/etc/inetd.conf` empfiehlt, und lassen Sie diesen Dienst deaktiviert! Ein Computer ohne eigene Festplatte oder ein Router kann mit Hilfe dieses Dienstes von einem lokalen und geschützten Netzwerk booten. Dummerweise halten ein paar Leute `tftp` für eine sichere Alternative zu `ftp`, weil `tftp` ohne Authentifizierung funktioniert. Natürlich ist genau das Gegenteil der Fall.

```
#tftp      dgram    udp      wait    root    /usr/sbin/tcpd  in.tftpd
```

- **bootps:** Ein Computer ohne eigenes Bootmedium fordert über BOOTP seine IP-Adresse sowie die des Boot-Servers an und initialisiert den Bootvorgang über `tftp`. Keine normale Installation zu Hause oder im Büro wird das verwenden. Lassen Sie es inaktiv.

```
#bootps dgram    udp      wait    root    /usr/sbin/tcpd  bootpd
```

Hinweis

RARP, BOOTP und DHCP

bootpd und dhcpcd teilen sich in `/etc/services` zwar den gleichen Port, sind aber dennoch zwei unterschiedliche Dienste.

Ein Computer ohne Bootmedium kennt zwar seine »Hardware-Adresse«, die MAC-Adresse seiner Netzwerkkarte, nicht aber die »Software-Adresse«, nämlich seine IP-Adresse. Über RARP (Reverse Address Resolution Protocol) kann so eine Maschine bei einem Server die zu einer MAC-Adresse passende IP erfragen.

RARP wurde später durch BOOTP (Bootstrap Protocol) abgelöst. BOOTP liefert nicht nur eine IP-Adresse, sondern zusätzlich die Adresse des Fileservers, von dem die Maschine über `tftp` booten kann.

BOOTP wurde nicht ersetzt, hat sich aber zu DHCP (Dynamic Host Configuration Protocol) weiterentwickelt. DHCP bietet alle Leistungen von BOOTP, liefert darüber hinaus aber noch zusätzliche Informationen über Router und Nameserver und vergibt dynamische IP-Adressen automatisch neu.

Informationsdienste

Die »Informationsdienste« aus `/etc/inetd.conf` liefern Daten über Benutzer, Prozesse, Netzwerkverbindungen und Konfiguration. Falls Sie diese Services verwenden wollen, sollten Sie sie auf Maschinen im eigenen LAN beschränken. Diese Dienste benutzen zwei verschiedene Sockets: `inet-Domain-Sockets` für Anfragen von fremden Rechnern sowie `Unix-Domain-Sockets` für lokale Clients. Eine Deaktivierung in `/etc/inetd.conf` betrifft nur die ersteren; lokale Zugriffe sind weiterhin möglich.

- `finger` ist als reiner Informationsdienst keine große Sicherheitslücke, gibt einem potenziellen Eindringling aber Auskunft darüber, welche Accounts bei Ihnen existieren, wann sie zuletzt benutzt worden sind, ob ungelesene Mail herumliegt, wie Ihre Computer heißen usw. Diese Daten sind nicht unbedingt privat, können für einen Black Hat aber sehr nützlich sein. Deshalb würde ich Ihnen nicht dazu raten, andere auf Ihren `finger`-Server zugreifen zu lassen.

```
#finger stream tcp nowait root /usr/sbin/tcpd in.fingerd
```

- `cfinger`, `sysstat` und `netstat` liefern Daten über das System und die darauf existierenden Benutzeraccounts – nützlich für einen Eindringling. Wie bereits gesagt, deaktivieren Sie in `/etc/inetd.conf` nur den Zugriff über das Netzwerk, während Sie die Dienste auf dem eigenen Rechner weiterhin verwenden können.

```
#cfinger stream tcp nowait root /usr/sbin/tcpd in.cfingerd
#sysstat stream tcp nowait guest /usr/sbin/tcpd /bin/ps -auwwx
#netstat stream tcp nowait guest /usr/sbin/tcpd /bin/netstat -f inet
```

Authentifizierung

`auth` ähnelt `finger` insofern, als es ebenfalls Benutzerinformationen liefert. Meist benutzen Mail- und Newsserver dieses Protokoll, um festzuhalten, wer eine bestimmte Mail oder einen News-Artikel sendet. Auch FTP-Server führen manchmal eine `identd`-Anfrage durch, bevor sie den Zugriff erlauben. Ob Sie `auth` bzw. `identd` aktivieren, bleibt Ihnen überlassen. Die Frage wird heftig diskutiert und es gibt sehr gute Argumente dafür und dagegen. Manche Leute bestehen schon aus Höflichkeit auf diesem Dienst. Andere deaktivieren ihn, weil sie die Namen ihrer Benutzeraccounts als vertraulich betrachten.

```
auth stream tcp nowait nobody /usr/sbin/in.identd in.identd -l -e -o
```

Hinweis

identd muss nicht unbedingt über inetd gestartet werden. Je nachdem, welche Version auf Ihrem Computer installiert ist, ist identd möglicherweise auch ein völlig selbstständiger Prozess, der unter der Kontrolle des Runlevel-Managers steht.

Lokale Services

Neben der vollautomatischen Verwaltung von Dämonen durch den Runlevel-Manger und `inetd` betreiben Sie möglicherweise noch weitere Dienste und Dämonen, die nicht Teil einer normalen Linux-Distribution sind. Diese müssen Sie explizit starten.

Normalerweise geschieht das über das Skript `/etc/rc.d/rc.local`. Dieses wird beim Bootvorgang ausgeführt und setzt lokale Einstellungen um. In der Voreinstellung schreibt es nur die Datei `/etc/issue` mit der Begrüßungsmeldung, die vor dem Login-Prompt steht. Sie können hier aber auch andere Aufgaben erledigen lassen, z.B. einen `sshd`-Server aufrufen oder die Systemzeit von einem anderen Computer übernehmen.

2.7 Zusammenfassung

Nach der Betrachtung der IP-Protokolle und den ausführlichen Überlegungen, welche Dienste auf einem Server aktiv sein sollten, wird es im nächsten Kapitel darum gehen, wie man eine Firewall erstellt. Das Firewall-Skript enthält spezifische Regeln für Ein- und Ausgabe, die mit Hilfe der Informationen aus den Datenpaketen die Konzepte aus diesem Kapitel in die Praxis umsetzen.

Kapitel 3

Gestaltung und Installation einer Firewall



- 3.1 ipchains: Firewall-Verwaltung unter Linux
- 3.2 Initialisierung der Firewall
- 3.3 ICMP-Nachrichten filtern
- 3.4 Dienste auf festen, unprivilegierten Ports schützen
- 3.5 Zwingend benötigte Dienste erlauben
- 3.6 Häufige TCP-Dienste
- 3.7 Häufige UDP-Dienste
- 3.8 Abgewiesene Pakete protokollieren
- 3.9 Zugriff für problematische Sites pauschal sperren
- 3.10 LAN-Zugang
- 3.11 Die Firewall installieren
- 3.12 Zusammenfassung

Kapitel 2 beschreibt die Ideen und Konzepte, auf denen eine Firewall aufbaut. Für jede Chain einer Firewall wird in der *Policy* eingestellt, was mit Paketen passiert, auf die keine Regel explizit zutrifft. Jede Regel gibt nicht nur genau an, ob sie für eingehende oder ausgehende Pakete gilt, sondern spezifiziert auch das Netzwerk-Interface, das Protokoll (z.B. TCP, UDP oder ICMP) und die Ports. Regeln, dass Pakete angenommen, abgelehnt oder verworfen werden sollen, existieren separat für die Chains *input* und *output* sowie auch für *forward*. Über die *forward*-Chain erfahren Sie mehr am Ende dieses Kapitels sowie im nächsten Kapitel. All die notwendigen Konzepte und Begriffe fasse ich in diesem Kapitel zusammen und demonstriere, wie Sie eine einfache Firewall für ein einzelnes System erstellen.

Die Firewall aus diesem Kapitel weist in der Voreinstellung alle Pakete zunächst einmal ab. Die gewünschten Dienste aktivieren wir einzeln als Ausnahmen von dieser Voreinstellung.

Nach dem Aufbau einer Firewall für ein einzelnes System schließt das Kapitel mit einer Demonstration, wie man sie zu einer Bastion-Firewall erweitert. Eine *Bastion-Firewall* hat mindestens zwei Netzwerk-Interfaces und trennt ein lokales Netz vom Internet. Eine Firewall für ein einzelnes System lässt sich mit einfachen Ergänzungen zu so einer Bastion-Firewall ergänzen. Sie schützt dann Ihr internes LAN durch Paketfilter und agiert als Proxy zwischen dem LAN und dem Internet.

Firewalls für ein einzelnes System sowie Bastion-Firewalls sind die unsichersten Architekturen für Firewall-Implementierungen. Wenn jemand in die Maschine mit der Firewall einbricht, sind alle angeschlossenen Maschinen verwundbar. Eine Standalone-Firewall arbeitet quasi nach dem Alles-oder-Nichts-Prinzip. Da sich dieses Buch aber an Benutzer auf privaten Systemen oder in kleinen Firmen richtet, gehen wir einfach einmal davon aus, dass die meisten Leser nur einen einzigen Computer ans Internet anschließen werden, oder ein kleines LAN über eine einzelne Firewall schützen.

»Unsicherste Architektur« im Vergleich zu anderen Systemen heißt nun nicht, dass eine Firewall mit diesem Design schrecklich unsicher wäre. Solche Firewalls sind aber weniger flexibel als kompliziertere Architekturen mit mehreren Systemen. In Kapitel 4 lernen Sie andere Konfigurationen kennen, die zusätzliche innere Sicherheit einführen und damit kompliziertere LANs und Server schützen können, als das mit einer Firewall auf einem Einzelsystem möglich ist.

3.1 **ipchains: Firewall-Verwaltung unter Linux**

Dieses Buch basiert auf RedHat Linux. Linux enthält eine Firewall namens IPFW (IP FireWall). Alle größeren Linux-Distributionen haben bereits auf *ipchains* umgestellt, eine Fassung von IPFW Version 4. Diese neue Version wird meistens einfach nur als *ipchains* bezeichnet, nach dem Namen des zugehörigen Administrations-Programms. In älteren Linux-Versionen war eine ältere IPFW-Implementierung enthalten, die man nach dem damaligen Programm *ipfwadm* nennt.

Die Beispiele in diesem Buch beziehen sich auf `ipchains`. Weil jedoch ältere Linux-Installationen weiterhin `ipfwadm` verwenden, finden Sie in Anhang B auch dafür passende Beispiele. Obwohl zwischen `ipchains` und `ipfwadm` Unterschiede in der Syntax bestehen, sind die beiden funktionell gleich. `ipchains` umfasst alle Befehle von `ipfwadm` plus einige zusätzliche Features aus der neuen IPFW-Implementierung. Die neuen Features werden in diesem Buch allerdings nicht benutzt. In den Beispielen finden Sie nur Optionen, die beide Versionen verstehen.

`ipchains` ist ein Programm zur Administration einer Firewall. Es legt Filterregeln für die einzelnen Ketten fest, aus denen sich die Firewall zusammensetzt. Einer der wichtigsten Aspekte ist dabei die Reihenfolge, in der die Regeln erscheinen.

Die Regeln für den Paketfilter werden in Tabellen im Kernel gespeichert, separat für jede Chain – `input`, `output` oder `forward` –, und zwar in der Reihenfolge, in der sie festgelegt wurden. Neue Regeln können am Anfang oder am Ende einer Chain eingefügt werden. In den Beispielen dieses Kapitels werden die Regeln immer nur angehängt (mit einer Ausnahme am Ende des Kapitels). Die Reihenfolge, in der Sie die Regeln definieren, ist die Reihenfolge, in der sie in den Kernel-Tabellen abgelegt werden und damit auch die Reihenfolge, in der die Pakete mit den Regeln verglichen werden.

Wenn ein Paket von außen an einem Netzwerk-Interface eintrifft, wird sein Header mit jeder Regel der `input`-Chain verglichen, bis eine passende Regel gefunden wird. Umgekehrt geschieht Ähnliches für abgehende Pakete, die das eigene System an ein Netzwerk-Interface schickt: Seine Header werden nacheinander mit den Regeln der zugehörigen `output`-Chain verglichen, bis eine Regel passt. In beiden Richtungen bricht der Vergleichsvorgang ab, sobald eine passende Regel gefunden wird. Die Regel legt fest, was mit dem Paket passieren soll: `ACCEPT`, `REJECT` oder `DENY`. Nur wenn überhaupt keine Regel passt, kommt die voreingestellte *Policy* der jeweiligen Chain zum Zuge. Langer Rede kurzer Sinn: *Die erste passende Regel gewinnt.*

In den Beispielen dieses Kapitels habe ich die Ports immer numerisch angegeben, nicht mit ihrem symbolischen Namen aus `/etc/services`. `ipchains` unterstützt zwar auch die symbolischen Namen. Für solche Beispiele wie hier eignen sich die Nummern aber besser, weil die symbolischen Namen nicht bei jeder Linux-Distribution identisch sind und sich manchmal sogar von einer Version zur nächsten ändern. In Ihren eigenen Regeln können Sie natürlich gerne symbolische Portnamen verwenden, aber denken Sie daran, dass Ihre Firewall beim nächsten Update dann vielleicht nicht mehr funktioniert.

`ipchains` ist ein C-Programm, das für jede Firewall-Regel einmal aufgerufen werden muss. Das erledigt normalerweise ein Shell-Skript, z.B. `/etc/rc.d/rc.firewall`. Soweit unterschiedliche Shells eine unterschiedliche Syntax haben, sind unsere Beispiele immer für die Shells Bourne (`sh`), Korn (`ksh`) oder Bourne Again (`bash`) geeignet.

Die Beispiele sind auch nicht irgendwie optimiert. Größtes Ziel war die Übersichtlichkeit und die Beibehaltung von konzeptueller und inhaltlicher Kompatibilität zwischen `ipchains` und `ipfwadm`. Diese beiden Programme benutzen für ähnliche Funktionalität unterschiedliche Kommandozeilenargumente und bieten Ihnen unterschiedliche Kürzel und Optimierungen. Hier werden nur die gemeinsamen Features eingesetzt.

3.1.1 Im Firewall-Skript eingesetzte ipchains-Optionen

Dieses Kapitel behandelt nicht alle `ipchains`-Optionen, sondern nur die in den Beispielen tatsächlich eingesetzten, also diejenigen, die mit `ipfwadm` kongruent sind. Tabelle 3.1 listet die verwendeten Kommandozeilenoptionen auf.

Hinweis

Vollständige Dokumentation zu ipchains liefern die Manual-Page zu ipchains(8) sowie das IPCHAINS-HOWTO.

```
ipchains
-A|-I [Chain]
[-i Interface]
[-p Protokoll]
[[-!] -y]
[-s Adresse [Port[:Port]]]
[-d Adresse [Port[:Port]]]
-j Policy
[-!]
```

Option	Beschreibung
-A [Chain]	Hängt eine Regel ans Ende einer Chain an. In den Beispielen werden nur die vordefinierten Chains <code>input</code> , <code>output</code> und <code>forward</code> verwendet. Wenn keine Chain angegeben ist, gilt die neue Regel für alle Chains.
-I [Chain]	Fügt eine Regel vor dem Anfang einer Chain ein. In den Beispielen werden nur die vordefinierten Chains <code>input</code> , <code>output</code> und <code>forward</code> verwendet. Wenn keine Chain angegeben ist, gilt die neue Regel für alle Chains.
-i Interface	Netzwerk-Interface, für das die Regel gilt. Wenn die Option <code>-i</code> nicht verwendet wird, gilt die Regel für alle Interfaces. Typische Namen für Interfaces sind <code>eth0</code> , <code>eth1</code> , <code>lo</code> und <code>ppp0</code> .
-p Protokoll	IP-Protokoll, für das die Regel gilt. Wenn die Option <code>-p</code> nicht verwendet wird, gilt die Regel für alle Protokolle. Mögliche Protokolle sind <code>tcp</code> , <code>udp</code> , <code>icmp</code> und <code>all</code> . Daneben sind alle Protokoll-Namen und -Nummern aus <code>/etc/protocols</code> erlaubt.

Tab. 3.1: Im Firewall-Skript eingesetzte ipchains-Optionen

Option	Beschreibung
<code>-y</code>	Das SYN-Flag einer TCP-Nachricht muss gesetzt, das ACK-Flag darf nicht gesetzt sein, d.h. das Paket muss das erste Paket des Handshakes für den Verbindungsaufbau sein. Wenn weder <code>-y</code> noch <code>! -y</code> angegeben sind, werden die TCP-Flags nicht überprüft.
<code>! -y</code>	Das ACK-Flag einer TCP-Nachricht muss gesetzt sein, d.h. das Paket muss eine Antwort im Handshake für den Verbindungsaufbau darstellen oder Teil einer bereits bestehenden Verbindung sein. Wenn weder <code>-y</code> noch <code>! -y</code> angegeben sind, werden die TCP-Flags nicht überprüft.
<code>-s Adresse [Port]</code>	Absender-Adresse des Pakets. Wenn keine Absenderadresse angegeben ist, sind alle Unicast-Absender erlaubt. Wenn zusätzlich ein Port oder Portbereich angegeben ist, gilt die Regel nur für diese Ports. Eine Regel ohne explizite Ports gilt für alle Ports. Ein Portbereich wird durch den untersten und obersten Port definiert, getrennt durch einen Doppelpunkt (z.B. 1024:65535). Ein Port kann nicht ohne Adresse festgelegt werden.
<code>-d Adresse [Port]</code>	Empfänger-Adresse des Pakets. Wenn keine Empfängeradresse angegeben ist, sind alle Unicast-Empfänger erlaubt. Wenn zusätzlich ein Port oder Portbereich angegeben ist, gilt die Regel nur für diese Ports. Eine Regel ohne explizite Ports gilt für alle Ports. Ein Portbereich wird durch den untersten und obersten Port definiert, getrennt durch einen Doppelpunkt (z.B. 1024:65535). Ein Port kann nicht ohne Adresse festgelegt werden.
<code>-j Policy</code>	Policy dieser Regel: ACCEPT, REJECT oder DENY. In der forward-Chain ist auch MASQ (Masquerading) erlaubt.
<code>-l</code>	Jedesmal, wenn diese Regel auf ein Paket zutrifft, wird eine entsprechende Meldung der Facility kern (Kernel) und der Priorität info (Informationsmeldung, die nicht auf ein Problem verweist) ins syslog geschrieben. Diese Meldungen landen normalerweise in der Datei <code>/var/log/messages</code> .

Tab. 3.1: Im Firewall-Skript eingesetzte ipchains-Optionen (Forts.)

3.1.2 IP-Adressen von Absender und Empfänger

Eine Firewall-Regel kann sowohl eine Absender- als auch eine Empfänger-Adresse enthalten. Sie trifft dann nur auf Pakete mit genau so einer Absender- bzw. Empfänger-IP zu. Als Adresse kann dabei eine einzelne IP-Adresse, ein voll qualifizierter Hostname, ein Domainname (Netzwerkname), ein Adressbereich, oder »alles« benutzt werden.

Eine IP-Adresse ist eine 32-Bit-Zahl, aufgeteilt in vier Bytes zu je acht Bit, also vier Zahlen von 0 bis 255. Diese vier Zahlen trennt man durch Punkte voneinander. (Als ob Sie das nicht längst wüssten!) In den Beispielen in diesem Buch wird für die Adresse des eigenen Rechners immer die Adresse 192.168.10.30 (aus dem privaten Klasse-C-Bereich) verwendet.

ipchains versteht CIDR-Adressen, bei denen an die IP-Adresse eine Bitmaske angehängt ist. Die Maske ist eine Zahl von 0 bis 32 und gibt die Zahl der zu maskierenden Bits an. Dabei werden die Bits von links gezählt (vom signifikantesten Bit – MSB – an). Die Bitmaske gibt an, wie viele der führenden Bits von Paketadresse und Regeladresse übereinstimmen müssen.

Eine Bitmaske von 32 (/32) bedeutet, dass alle Bits passen müssen. Die Adressen in Paket und Regel müssen also exakt übereinstimmen. Diese Maske ist die Voreinstellung, d.h. Sie können sie auch weglassen. 192.168.10.30 und 192.168.10.30/32 haben die gleiche Bedeutung.

Hinweis

IP-Adressen als symbolische Namen

Hosts und Netzwerke dürfen auch als symbolische Namen angegeben werden. Ein Hostname ist besonders bequem bei Regeln, die nur für einen einzelnen Host gelten, v.a. wenn sich die IP-Adresse öfter einmal ändert oder wenn der Hostname insofern für mehrere IP-Adressen steht, wie es z.B. bei einem Mail-Server häufig der Fall ist. Allgemein gilt aber: Geben Sie Adressen lieber als IP-Adressen an, weil DNS-Hostnamen gefälscht werden können.

Symbolische Namen lassen sich nicht auflösen, wenn die Firewall-Regeln (noch) kein DNS erlauben. Wenn Sie solche Hostnamen benutzen wollen, dann erst nach den Regeln für DNS.

Ein Beispiel für die Verwendung von Bitmasken: Angenommen, Sie wollen eine bestimmte Verbindung nur zwischen Ihrem Computer und den Rechnern Ihres Internet-Providers erlauben. Nehmen wir weiter an, Ihr ISP verwendet für seine Server Adressen im Bereich von 192.168.24.0 bis 192.168.27.255. In diesem Fall wären Adresse und Maske 192.168.24/22. Wie Bild 3.1 zeigt, sind die ersten 22 Bits für alle Adressen im angegebenen Bereich identisch.

Die Maske 0 (/0) bedeutet, dass überhaupt kein Bit der beiden Adressen übereinzustimmen braucht. Mit anderen Worten, die Angabe von /0 ist gleichbedeutend damit, gar keine Adresse anzugeben, weil ja alle Bits beliebig sein dürfen. Jede Unicast-Adresse passt. In ipchains gibt es für 0.0.0.0/0 ein Kürzel: any/0.

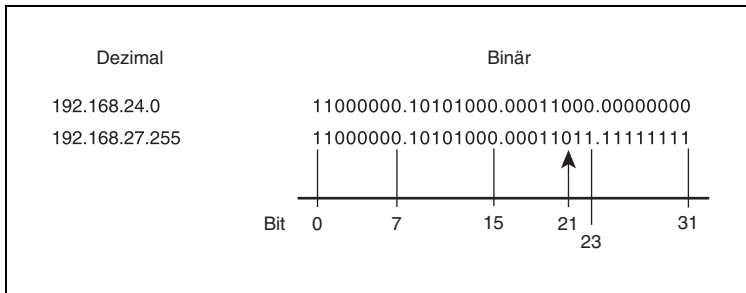


Bild 3.1: Die ersten 22 Bits im IP-Bereich 192.168.24.0/22

3.2 Initialisierung der Firewall

Eine Firewall besteht aus einer Reihe von Filterregeln, die durch entsprechende Aufrufe von `ipchains` erzeugt werden. Für jede einzelne Regel wird `ipchains` einmal aufgerufen. (Eine Firewall kann zehn Regeln umfassen oder ein paar hundert.)

Sie sollten `ipchains` über ein Skript aufrufen und nicht manuell. Das Skript sollte jedesmal komplett durchlaufen. Versuchen Sie nicht, einzelne `ipchains`-Regeln einzugeben, weil Ihre Firewall daraufhin Pakete in unpassender Weise akzeptieren oder verweigern könnte. Wenn die Chains initialisiert werden und die voreingestellte Policy gesetzt ist, sind alle Netzwerkdienste blockiert, bis die festgelegten Ausnahmeregeln in Kraft treten und einzelne Dienste wieder erlauben.

Das ist auch der Grund, warum Sie das Firewall-Skript auf der Konsole aufrufen sollten und nicht von einem anderen Computer bzw. von einem `xterm`. Während der Ausführung des Skripts ist der Zugang zum Netzwerk einschließlich des von X-Windows benutzten Loopback-Interfaces vorübergehend unterbrochen, solange bis der Zugang wieder explizit erlaubt wird.

Denken Sie daran, dass die Firewall ihre Regeln in der Reihenfolge abarbeitet, in der sie definiert sind. Die Regeln werden Stück für Stück an die Ketten angehängt. Die erste passende Regel gewinnt und entscheidet, was mit einem Paket passiert. Deshalb sollten Sie die Firewall-Regeln immer hierarchisch anordnen, von den spezifischsten hin zu den allgemeinsten.

Unter der »Initialisierung einer Firewall« versteht man eine ganze Menge. Darunter fallen die im Firewall-Skript definierten globalen Konstanten, das Löschen der alten Firewall-Regeln, die Definition der Policies für die einzelnen Ketten, das Einschalten des Loopback-Interfaces für den normalen Betrieb Ihres Computers, sowie die Regeln an sich, mit denen bestimmte Hosts oder Netzwerke gesperrt, fehlerhafte Adressen ignoriert und einzelne lokale Dienste vor fremdem Zugriff geschützt werden.

3.2.1 Symbolische Konstanten für die Beispiele

Ein Firewall-Skript lässt sich meines Erachtens am leichtesten lesen und warten, wenn man für immer wieder auftretende Namen und Adressen symbolische Konstanten benutzt. Die folgenden Konstanten werde ich entweder immer wieder verwenden, oder es handelt sich sogar um universelle Konstanten, die in den Netzwerk-Standards definiert wurden.

```
EXTERNAL_INTERFACE="eth0"           # ans Internet angeschlossenes In-
                                     # terface
LOOPBACK_INTERFACE="lo"             # oder wie auch immer es bei Ihnen
                                     # heißt
IPADDR="ei.ge.ne.IP"               # eigene IP-Adresse
ANYWHERE="any/0"                   # passt auf jede beliebige IP-
                                     # Adresse
MY_ISP="IP.Bereich.des.ISP"        # Adressbereich von ISP und NOC

LOOPBACK="127.0.0.0/8"              # Loopback-Adressbereich
CLASS_A="10.0.0.0/8"               # Klasse A: private Netze
CLASS_B="172.16.0.0/12"            # Klasse B: private Netze
CLASS_C="192.168.0.0/16"           # Klasse C: private Netze
CLASS_D_MULTICAST="224.0.0.0/4"    # Klasse D: Multicast-Adressen
CLASS_E_RESERVED_NET="240.0.0.0/5" # Klasse E: reservierte Adressen
BROADCAST_SRC="0.0.0.0"            # Broadcast-Absender
BROADCAST_DEST="255.255.255.255"   # Broadcast-Empfänger
PRIVPORTS="0:1023"                 # privilegierte Ports
UNPRIVPORTS="1024:65535"           # unprivilegierte Ports
```

Hier nicht aufgeführte Konstanten definiere ich im Kontext der jeweils zugehörigen Regeln.

3.2.2 Löschen alter Firewall-Regeln

Wenn Sie neue Regeln für Ihre Firewall festlegen, löschen Sie als Erstes immer die eventuell noch gültigen alten Regeln aus den Chains heraus. Ansonsten würde der Kernel die neuen Regeln einfach ans Ende der alten Chains anhängen. Vermutlich würde dann jedesmal eine der alten Regeln auf ein Paket zutreffen, und die neuen Regeln würden nie benutzt.

Das Löschen einer Kette wird als *Flushing* bezeichnet und über die `ipchains`-Option `-F` durchgeführt. Wenn Sie nicht explizit angeben, *welche* Chain Sie löschen wollen, dann sind alle drei eingebauten Ketten (`input`, `output` und `forward`) betroffen.

```
# Vorbestehende Regeln löschen
ipchains -F
```

Die Chains sind jetzt leer, Sie können Sie ganz von Anfang an definieren. Die alte Policy ist zwar noch gesetzt, aber wir werden als Nächstes eine neue Policy festlegen.

3.2.3 Festlegung der voreingestellten Policy

Das *Flushing* oder Löschen der Regeln hat keine Auswirkung auf die Policy der einzelnen Chains, d.h. auf die Voreinstellung, was mit einem Paket geschehen soll, wenn keine Regel zutrifft. Die Policy muss immer separat gesetzt werden.

Normalerweise soll die Firewall alle ankommenden Pakete verwerfen (DENY) und alle abgehenden Pakete ablehnen (REJECT). D.h. solange keine expliziten Ausnahmeregeln definiert sind, werden ankommende Pakete stillschweigend ignoriert, ohne Rückmeldung an den Absender, wohingegen abgehende Pakete mit einer ICMP-Fehlermeldung blockiert werden. Der Unterschied für den Benutzer liegt darin, dass z.B. der Browser eines Anwenders im Internet, der auf Ihren (nicht existierenden) Web-Server zugreifen will, einfach hängen bleibt, bis sein Computer einen TCP-Timeout liefert. Er weiß nicht, ob Sie nun tatsächlich einen Web-Server betreiben oder nicht. Wenn umgekehrt Sie versuchen, auf einen fremden Web-Server zuzugreifen, ohne dass Sie es sich in der Firewall erlaubt haben, dann erhalten Sie sofort eine Fehlermeldung, dass dieser Zugriff nicht erlaubt ist.

```
# Voreingestellte Policies setzen
ipchains -P input DENY
ipchains -P output REJECT
ipchains -P forward REJECT
```

Nach diesen Zeilen ist jeder Netzwerk-Traffic gesperrt.

3.2.4 Einschalten des Loopback-Interfaces

Sie sollten auf dem Loopback-Interface alles zulassen, ohne irgendwelche Restriktionen. Dadurch können Sie jeden gewünschten – oder vom System benötigten – lokalen Dienst nutzen, ohne sich um alle Firewall-Regeln kümmern zu müssen.

Traffic von und zu Loopback wird ganz am Anfang des Firewall-Skripts zugelassen. Dieses Interface ist von außen nicht zugänglich. Lokale Services, z.B. X-Windows, sind blockiert, solange sie Loopback nicht benutzen können.

Um alles zuzulassen, braucht man keine komplizierten Regeln. Sie heben einfach die Policy für das Loopback-Interface auf, indem Sie alles auf diesem Interface akzeptieren (ACCEPT):

```
# Keine Einschränkungen für Loopback
ipchains -A input -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT
```

Hinweis***Policies und »Die erste Regel gewinnt«***

Die Policies scheinen Ausnahmen von der Faustregel »Die erste Regel gewinnt« zu sein, denn ihre Wirkung hängt nicht davon ab, ob sie vor oder nach den anderen Regeln gesetzt werden. Policies sind eigentlich überhaupt keine Regeln. Die Policy einer Chain kommt dann zum Tragen, wenn ein Paket mit jeder Regel verglichen worden ist und auf keine gepasst hat.

Im Skript werden die Policies zuallererst definiert, damit klar ist, was mit einem Paket passiert, sofern keine der danach festgelegten Ausnahmen zutrifft.

Wenn Sie die Policies erst am Ende definieren und im Skript einen Syntax-Fehler begehen, bricht das Skript vorher ab, und die Policies werden nie gesetzt. Damit ist nach dem Bootvorgang die Policy ACCEPT gültig: Wenn ein Paket zu keiner Regel passt, gilt ACCEPT, und das Paket darf passieren. (Wenn es auf eine Regel passt, hilft das auch nichts, weil unsere Regeln ja als Ausnahmen von der Voreinstellung DENY definiert sind und damit Pakete i.d.R. akzeptieren.) D.h. in so einer Situation würde die Firewall nichts ausrichten. Deshalb: Lassen Sie die Policy-Einstellung am Anfang.

Jetzt funktionieren das syslog, X-Windows und andere lokale Services wieder, die Unix-Domain-Sockets benutzen.

3.2.5 Gefälschte Absender und andere fehlerhafte Adressen

Die Regeln aus diesem Abschnitt sind für die input-chain gedacht. Sie filtern Absender- und Empfänger-IPs heraus, die normalerweise nicht aus dem Internet kommen dürfen.

Linux enthält im Kernel schon gewisse Schutzmechanismen gegen gefälschte Pakete, die wir nur noch einschalten müssen. Außerdem sollten SYN-Cookies aktiviert werden:

```
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Schutz vor gefälschten IP-Adressen:
# Source Address Verification einschalten
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 >$f
done
```

Eigene IP-Adresse

Einer der wenigen Fälle einer gefälschten Absenderadresse, die Sie auf der Ebene eines Paketfilters entdecken können, ist eine Fälschung Ihrer eigenen IP-Adresse. Die folgende Regel verwirft ankommende Pakete, die angeblich von Ihnen selbst sind:


```
# Ankommende Pakete ablehnen, die vorgeblich von der
# IP-Adresse unseres externen Interfaces kommen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $IPADDR -j DENY -1
```

Hinweis***Log-Optionen der Firewall***

Die Option -l aktiviert das Logging von Paketen, für die diese Regel zutrifft. Wenn ein passendes Paket ankommt, wird dieses Ereignis ans syslog geschickt und typischerweise in /var/log/messages festgehalten. Seit RedHat 6.0 steht das Logging im Kernel automatisch zur Verfügung; bei früheren Versionen mussten Sie den Kernel entsprechend neu übersetzen.

Abgehende Pakete an Sie selbst brauchen Sie nicht blockieren. Sie kommen sicherlich nicht zurück, behaupten, von Ihnen abgeschickt worden zu sein, und werden als gefälscht abgewiesen. Wenn Sie ein Paket an Ihr eigenes externes Interface schicken, dann kommt es nicht wirklich über dieses externe Interface an, sondern über Loopback. Auch in diesem Fall erreicht das Paket mit Ihrer IP als Absender also niemals die input-Chain des externen Interfaces.

Private IP-Adressen (Klassen A bis C)

Wie in Kapitel 2 erklärt, gibt es in jeder der Klassen A, B und C einen reservierten IP-Bereich für private Netze. Diese IPs dürfen im Internet nicht benutzt werden. Router sollen Pakete mit solchen IPs im Absender nicht weitergeben; hat ein Paket eine IP in diesem Bereich als Empfänger, kann der Router gar nicht damit umgehen. Trotz dieser Regel lassen einige Router Pakete mit einer Absenderadresse in solch einem Bereich passieren.

Ganz abgesehen davon werden Sie Pakete mit diesen Adressen auch unabhängig vom Router sehen, wenn jemand sie auf Ihrer Seite des Routers ins Netz schickt. Falls Ihre Proxy- oder Masquerading-Konfiguration Fehler enthält, können auch Computer in Ihrem eigenen LAN Pakete mit einer privaten Absender-IP verschicken.

Die folgenden Regeln unterbinden ankommende und abgehende Pakete, bei denen Absender- oder Empfänger-Adresse in einem der drei privaten Bereiche liegt. Keines dieser Pakete ist außerhalb eines privaten Netzes erlaubt.

```
# Paket abweisen, wenn Absender oder Empfänger im privaten Klasse-A-Bereich
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY -l
```

```
# Paket abweisen, wenn Absender oder Empfänger im privaten Klasse-B-Bereich
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_B -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_B -j DENY -1
```

```
# Paket abweisen, wenn Absender oder Empfänger im privaten Klasse-C-Bereich
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_C -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_C -j DENY -1
```

Wenn Sie das Routing korrekt eingerichtet haben, wird Ihr externes Netzwerk-Interface nur Pakete an seine eigene IP-Adresse annehmen. Wenn Sie aber automatisch routen und private IP-Adressen in Ihrem LAN verwenden *und* ein anderer Kunde Ihres Providers, der aber im gleichen Subnetz hängt, Pakete mit genau diesen privaten IP-Adressen nach außen routet, dann könnte Ihre Firewall diese Pakete an Ihr LAN weitergeben.

Loopback-Adressen

Die nächsten Regeln verbieten Pakete mit einer Absender-Adresse im Loopback-Bereich:

```
# Pakete von Loopback verwerfen
ipchains -A input -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY -1
```

Loopback-Adressen beziehen sich auf ein Interface, das nur aus Ihrer Betriebssystem-Software besteht. Jedes ankommende Paket, das von solch einer Adresse stammt, ist mit Sicherheit gefälscht.

Hinweis

Beachten Sie, dass ich gefälschte Adressen von lokalen Benutzern im `syslog` aufzeichne.

Router *sollen* Pakete von Loopback nicht weiterleiten und *können* Pakete an Loopback nicht weiterleiten – ähnlich, wie bei privaten IP-Adressen.

Illegale Broadcast-Adressen

Als Nächstes blockieren wir Pakete mit einer illegalen Broadcast-Adresse im Absender- oder Empfänger-Feld. Da die Firewall per Voreinstellung ohnehin alle Pakete verwirft, kommen auch legitime Broadcast-Adressen nicht durch unsere Firewall und müssen – falls gewünscht – explizit erlaubt werden.

```
# Fehlerhafte Broadcast-Pakete verwerfen und aufzeichnen
ipchains -A input -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC -j DENY -1
```

Die erste Regel gilt für Pakete vom Absender 255.255.255.255, der Empfänger-Adresse für Broadcast-Pakete. Ein Paket wird niemals von der IP 255.255.255.255 kommen.

Die zweite Regel bezieht sich auf Pakete an den Empfänger 0.0.0.0, der Absender-Adresse für Broadcast-Pakete. Solche Pakete entstehen nicht durch Fehler, sondern werden absichtlich losgeschickt. Jemand will herausfinden, ob auf einem Computer ein Unix-Betriebssystem mit BSD-Netzwerkcode läuft. Da die meisten Unices BSD-Code für das Netzwerk verwenden, sucht er also effektiv nach Unix-Maschinen.

Hinweis***Klarstellung: Die Bedeutung der IP-Adresse 0.0.0.0***

Die Adresse 0.0.0.0 ist für die Verwendung als Broadcast-Absender reserviert. Die IPFW-Schreibweise für »beliebige Adresse«, nämlich any/0 oder 0.0.0.0/0 oder 0.0.0.0/0.0.0.0 beinhaltet nicht die Broadcast-Absenderadresse. Diese letztere hat nämlich noch ein zusätzliches Bit im IP-Header gesetzt, wodurch das betreffende Paket als Broadcast-Paket »für alle Interfaces auf dem Netz« markiert wird. Dieses Bit stellt klar, dass es sich nicht um ein normales Unicast-Paket an irgendeinen konkreten Host handelt. Broadcast-Pakete werden ganz anders behandelt als normale Pakete; es existiert auch keine legitime IP-Adresse 0.0.0.0.

Multicast-Adressen (Klasse D)

Multicast-Adressen sind nur als Empfänger-IPs erlaubt. Die folgende Regel verwirft Pakete, in denen sie als Absender auftauchen, und zeichnet die Ereignisse wieder auf:

```
# Pakete mit einer Klasse-D-Multicast-Adresse  
# im Absender-Feld verwerfen  
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j DENY -l  
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j REJECT -l
```

Legitime Multicast-Pakete benutzen als Protokoll immer UDP. Das bedeutet, dass ein Multicast-Paket – wie jedes andere UDP-Paket auch – immer von einer IP an eine andere IP geschickt wird. Das besondere an Multicast ist nur die Empfängeradresse in einem ganz bestimmten Adressbereich.

Die nächste Regel unterbindet abgehende Multicast-Pakete von Ihrem Computer:

```
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_D_MULTICAST -j REJECT -l
```

Multicast wird bei der Kernel-Übersetzung ein- oder ausgeschaltet. Zusätzlich muss es für ein konkretes Netzwerk-Interface eingeschaltet werden. Bei RedHat 6.0 ist es z.B. normalerweise aktiviert, bei früheren Versionen nicht. Wenn Sie an

einem Konferenzsystem oder anderen Audio- und Video-Übertragungen teilnehmen, benötigen Sie solche Pakete eventuell.

Wenn Sie sich nicht selbst aktiv als Empfänger angemeldet haben, erhalten Sie auch keine Pakete an Empfängeradressen im Multicast-Bereich. Multicast-Pakete werden an viele, aber klar definierte Empfänger verschickt. Allerdings habe ich auch schon Multicast-Pakete von anderen Rechnern auf dem gleichen Subnetz meines Internet-Providers gesehen. Wenn Sie Multicast nicht verwenden, können Sie solche Pakete pauschal sperren, wie in der nächsten Regel:

```
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_D_MULTICAST -j REJECT -1
```

Die Anmeldung für und das Routing von Multicast ist eine komplizierte Angelegenheit mit einem eigenen Protokoll, dem IGMP (das *Internet Group Management Protocol*, Protokoll-Nummer 2). Zu diesem Thema gibt es eine hervorragende technische Dokumentation von Vicki Johnson und Marjory Johnson: »How IP Multicast Works«, erhältlich von <http://www.ipmulticast.com/community/whitepapers/howipmcworks.html>.

Die Klasse D reicht von 224.0.0.0 bis 239.255.255.255. Die Konstante `CLASS_D_MULTICAST` (224.0.0.0/4) ist so definiert, dass sie nur die ersten vier Bits einer Adresse berücksichtigt. Wie Bild 3.2 zeigt, sind die ersten vier Bits (1110) der Zahlen 224 (11100000) bis 239 (11101111) identisch.

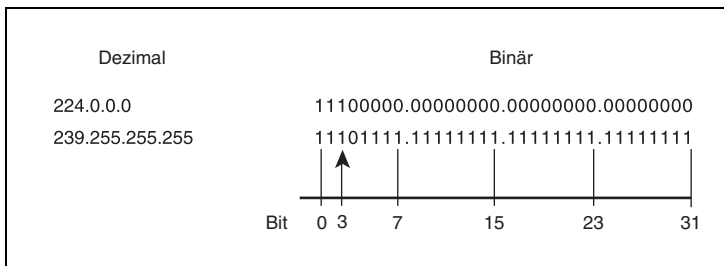


Bild 3.2: Die ersten vier Bits der Multicast-Adressen (Klasse D) sind identisch

Reservierte Adressen (Klasse E)

Die nächste Regel verwirft alle Pakete, die angeblich von einer Klasse-E-Adresse kommen, und zeichnet sie auf:

```
# IP-Adressen der reservierten Klasse E verwerfen
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_E_RESERVED_NET -j DENY -1
```

Die Klasse E reicht von 240.0.0.0 bis 247.255.255.255. Die Konstante `CLASS_E_MULTICAST` (240.0.0.0/5) ist so definiert, dass sie nur die ersten fünf Bits einer Adresse berücksichtigt. Wie Bild 3.3 zeigt, sind die ersten fünf Bits (11110) der Zahlen 240 (11110000) bis 247 (11110111) identisch.

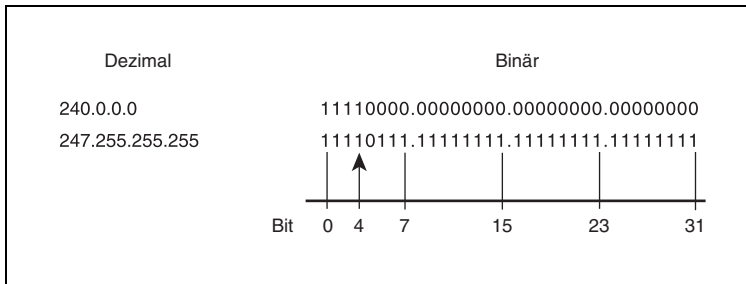


Bild 3.3: Die ersten fünf Bits der reservierten Klasse-E-Adressen sind identisch

Reservierte Adressen (IANA)

Die IANA verwaltet die Zuweisung und Registrierung von IP-Adressen. Genauere Informationen hierzu erhalten Sie unter <http://www.isi.edu/in-notes/iana/assignments/ipv4-address-space>. Einige Adressblöcke hat die IANA als reserviert deklariert. Diese sollten nicht öffentlich im Internet auftauchen. Die letzten Regeln aus diesem Abschnitt werfen Pakete von solchen Adressen:

```
# Von der IANA reservierte Adressen werfen
# 0.*.*.*, 1.*.*.*, 2.*.*.*, 5.*.*.*, 7.*.*.*, 23.*.*.*, 27.*.*.*
# 31.*.*.*, 37.*.*.*, 39.*.*.*, 41.*.*.*, 42.*.*.*, 58-60.*.*.*

ipchains -A input -i $EXTERNAL_INTERFACE -s 1.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 2.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 5.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 7.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 23.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 27.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 31.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 37.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 39.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 41.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 42.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 58.0.0.0/7 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 60.0.0.0/8 -j DENY -1

# 65: 01000001 - /3 beinhaltet 64 - deshalb 65-79 einzeln angeben
ipchains -A input -i $EXTERNAL_INTERFACE -s 65.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 66.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 67.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 68.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 69.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 70.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 71.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 72.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 73.0.0.0/8 -j DENY -1
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 74.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 75.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 76.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 77.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 78.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 79.0.0.0/8 -j DENY -1

# 80: 01010000 - /4 gilt für 80-95
ipchains -A input -i $EXTERNAL_INTERFACE -s 80.0.0.0/4 -j DENY -1

# 96: 01100000 - /4 gilt für 96-111
ipchains -A input -i $EXTERNAL_INTERFACE -s 96.0.0.0/4 -j DENY -1

# 126: 01111110 - /3 beinhaltet 127 - deshalb 112-126 einzeln angeben
ipchains -A input -i $EXTERNAL_INTERFACE -s 112.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 113.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 114.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 115.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 116.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 117.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 118.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 119.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 120.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 121.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 122.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 123.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 124.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 125.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 126.0.0.0/8 -j DENY -1

# 217: 11011001 - /5 beinhaltet 216 - deshalb 217-219 einzeln angeben
ipchains -A input -i $EXTERNAL_INTERFACE -s 217.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 218.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 219.0.0.0/8 -j DENY -1

# 223: 11011111 - /6 gilt für 220-223
ipchains -A input -i $EXTERNAL_INTERFACE -s 220.0.0.0/6 -j DENY -1
```

Hinweis

Indem man mit den Masken ein wenig jongliert, kann man hier auch mit weniger Regeln auskommen. Der besseren Übersicht halber habe ich aber fast alles explizit angegeben.

3.3 ICMP-Nachrichten filtern

ICMP-Kontrollnachrichten entstehen als Reaktion auf bestimmte Fehler. Außerdem werden sie von einigen Tools zur Netzwerkanalyse erzeugt, z.B. von `ping` und `tracert`. Tabelle 3.2 enthält die für eine kleine Site wichtigsten ICMP-Nachrichtentypen.

Code	Symbolischer Name	Beschreibung
0	<code>echo-reply</code>	Antwort auf ein <code>ping</code> .
3	<code>destination-unreachable</code>	Eine Fehlermeldung: Ein Router konnte das Paket auf dem Weg zum Empfänger nicht an den nächsten Computer weitergeben. Dieser Pakettyp wird von <code>tracert</code> benutzt.
4	<code>source-quench</code>	Flusskontrolle zwischen zwei Routern oder zwischen einem Router und einem Host.
5	<code>redirect</code>	Diese Routing-Nachricht wird erzeugt, wenn ein Router denkt, es existiere ein kürzerer Weg für ein bestimmtes Paket.
8	<code>echo-request</code>	Ein abgehendes <code>ping</code> .
11	<code>time-exceeded</code>	Diese Routing-Nachricht bedeutet, dass ein Paket zu oft von einem Router zum nächsten weitergegeben wurde, dass also die <i>Time-To-Live</i> (TTL) überschritten wurde.
12	<code>parameter-problem</code>	Der Header des IP-Paketes enthält ungültige Werte.

Tab. 3.2: Häufige ICMP-Nachrichtentypen

3.3.1 Fehler- und Kontrollnachrichten

Vier Typen von ICMP-Nachrichten müssen die Firewall passieren: `source-quench`, `parameter-problem`, eingehende `destination-unreachable`, sowie abgehende `destination-unreachable` vom Subtyp `fragmentation-needed`. Vier weitere Typen sind optional: `echo-request`, `echo-reply`, andere Sorten abgehender `destination-unreachables` sowie `time-exceeded`. Die restlichen Nachrichtentypen ignoriert man, sie werden dann von der voreingestellten Policy herausgefiltert.

Von den zu ignorierenden Typen enthält Tabelle 3.2 nur eine – `redirect` –, und zwar wegen ihrer Bedeutung in Denial-of-Service-Angriffen, z.B. als Redirect-Bombe (siehe ICMP-Redirect-Bomben). Wie auch `redirect` sind die übrigen ICMP-Typen spezielle Kontroll- und Statusmeldungen, die zwischen Routern ausgetauscht werden.

Hinweis**ICMP-Unterschiede zwischen *ipchains* und *ipfwadm***

ipchains unterstützt die Verwendung sowohl des numerischen Nachrichtentyps als auch des symbolischen Namens. Frühere Versionen von *ipfwadm* verstanden nur die numerischen Typen.

ipchains erlaubt darüber hinaus den Einsatz von Subtypen oder -codes. Das ist besonders interessant für eine feinere Kontrolle der destination-unreachable-Nachrichten. Z.B. könnten Sie abgehende port-unreachables explizit unterdrücken, um traceroutes zu verhindern, oder Sie könnten unter den abgehenden Paketen selektiv nur fragmentation-needed erlauben. *ipfwadm* hingegen kommt mit Subtypen nicht zurecht.

Eine Aufzählung aller erlaubten symbolischen Namen für ICMP-Nachrichten produziert der Befehl *ipchains -h icmp*. Die offiziellen Zuordnungen aus den RFCs sind in <http://www.isi.edu/in-notes/iana/assignments/icmp-parameters> zusammengestellt.

Die folgenden Abschnitte beschreiben die Nachrichtentypen, die für einen Host-computer – im Unterschied zu einem Router – von Bedeutung sind.

Kontrollnachricht source-quench (Typ 4)

ICMP-Nachrichten vom Typ 4 (source-quench) werden erzeugt, wenn ein Router Daten schneller produziert, als der nächste Router sie verarbeiten kann. Damit ist source-quench eine primitive Form der Flusskontrolle auf der Ebene der IP-Netzwerkschicht, typischerweise eingesetzt zwischen zwei benachbarten Geräten.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \  
-s $ANYWHERE 4 -d $IPADDR -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \  
-s $IPADDR 4 -d $ANYWHERE -j ACCEPT
```

Wenn ein Router vom Empfänger eine source-quench-Nachricht erhält, reagiert er darauf, indem er Pakete langsamer sendet. Die Geschwindigkeit erhöht er dann ganz allmählich wieder, bis erneut ein source-quench eintrifft.

Statusmeldung parameter-problem (Typ 12)

Der Typ 12 (parameter-problem) wird als Antwort auf ein Paket mit ungültigen oder unerwarteten Daten im Header erzeugt, oder wenn die Prüfsumme im Header nicht mit der Prüfsumme übereinstimmt, die der Empfänger ausgerechnet hat.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \  
-s $ANYWHERE 12 -d $IPADDR -j ACCEPT
```



```
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 12 -d $ANYWHERE -j ACCEPT
```

Fehlermeldung destination-unreachable (Typ 3)

Der ICMP-Typ 3 (destination-unreachable) steht für eine allgemeine Fehlermeldung. Der Header solch einer Nachricht enthält ein Feld mit einem Fehlercode, der die genaue Art des Fehlers angibt.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-s $ANYWHERE 3 -d $IPADDR -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 3 -d $ANYWHERE -j ACCEPT
```

Im Idealfall würden Sie abgehende Typ-3-Meldungen gerne komplett unterdrücken. Solche Pakete erzeugt Ihr System nämlich, wenn ein Eindringling untersucht, welche Ports auf welchen Ihrer Computer geöffnet sind. Ein Denial-of-Service-Angriff ist einfach dadurch möglich, dass jemand die ungenutzten UDP-Ports ihrer Computer mit allen möglichen Paketen überschüttet und Ihr System so dazu zwingt, eine riesige Menge an Fehlermeldungen zu generieren und zurückzusenden. Noch schlimmer: Seine IP-Adresse kann der Angreifer in diesem Szenario fälschen, wodurch Ihr Computer die Fehlermeldungen an völlig unschuldige Sites schickt.

Leider dürfen Sie destination-unreachables nicht ganz sperren. Der Subtyp fragmentation-needed wird nämlich für die Aushandlung der Größe von Paket-Fragmenten genutzt. Wenn Sie das blockieren, kann sich das sehr negativ auf die Performance Ihrer Netzwerkanbindung auswirken.

Wenn Sie traceroute von anderen Systemen aus zulassen wollen, müssen Sie für die output-Chain zusätzlich den Subtyp port-unreachable einschalten.

Statusmeldung time-exceeded (Typ 11)

ICMP-Nachrichten vom Typ 11 (time-exceeded) werden erzeugt, wenn ein Timeout auftritt, genauer gesagt, wenn ein Paket öfter als erlaubt von einem Router zum nächsten weitergegeben wurde. Auf den heutigen Netzwerken tritt time-exceeded meistens als Antwort auf ein traceroute auf.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-s $ANYWHERE 11 -d $IPADDR -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 11 -d $MY_ISP -j ACCEPT
```

Wenn Sie eingehende traceroutes erlauben wollen, müssen Sie abgehende time-exceeded-Nachrichten zulassen. Im angegebenen Beispiel dürfen nur die Computer Ihres Internet-Providers ein traceroute durchführen. Wenn Sie selber traceroute benutzen, müssen Sie zusätzlich auch ankommende time-exceededs anneh-

men. Weil Ihre Maschine nicht als Router arbeitet, benötigen Sie diesen Nachrichtentyp nicht für andere Zwecke.

Hinweis***Nur für Anwender von ipchains: Verwendung der Subtypen***

Wenn Sie ipchains benutzen, können Sie die allgemeine Regel für abgehende Pakete durch spezifischere Regeln ersetzen, die Typ-3-Nachrichten im Allgemeinen nur an Geräte Ihres Providers erlauben, fragmentation-needed aber an jede Empfänger-IP:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-s $ANYWHERE 3 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 3 -d $MY_ISP -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR fragmentation-needed -d $ANYWHERE -j ACCEPT
```

3.3.2 Kontrollnachrichten für ping: echo-request (Typ 8) und echo-reply (Typ 0)

ping verwendet zwei verschiedene ICMP-Typen. Die ping-Anfrage hat den Typ 8; die Antwort den Typ 0. ping ist ein einfaches Analyse-Utility, das noch aus der Zeit des DARPA-NET kommt. Der Name kommt vom »Ping« der Sonarsysteme. (DARPA steht immerhin für »Defense Advanced Research Projects Agency«.) Ähnlich wie beim Sonar wird ein echo-request an alle Hosts auf einem Netzwerk ein echo-reply von allen antwortenden Maschinen provozieren.

Abgehende pings

Mit den folgenden beiden Regeln können Sie jeden Host im Internet anpingen:

```
# abgehende Pings überallhin erlauben
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 8 -d $ANYWHERE -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-s $ANYWHERE 0 -d $IPADDR -j ACCEPT
```

Ankommende pings

Hier erlauben wir nur ausgewählten Rechnern, uns zu pingen:

```
# ankommende Pings vom eigenen Provider
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-s $MY_ISP 8 -d $IPADDR -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 0 -d $MY_ISP -j ACCEPT
```

Im diesem Beispiel dürfen die in `$IPADDR` definierten Maschinen Sie anpingen, falls NOC oder Support Ihres Providers Ihren Computer überprüfen wollen. Alle anderen echo-requests werden verworfen – ping kann immerhin für einige Denial-of-Service-Angriffe missbraucht werden.

smurf-Angriffe unterbinden

Bei einer smurf-Attacke schickt der Angreifer echo-request-Pakete an die Broadcast-Adresse eines zwischengeschalteten Netzwerkes, wobei die Absender-Adresse des echo-requests gefälscht ist und der Adresse des Opfers entspricht. Jeder Computer, der eines dieser Pakete sieht, wird als Antwort ein echo-reply an das Opfer senden. Indem der Angreifer sehr viele echo-requests an Broadcast-Adressen verteilt, wird das Opfer förmlich mit Antworten bombardiert, und es bleibt keine Bandbreite für seine reguläre Arbeit übrig.

Die folgenden Regeln zeichnen smurf-Angriffe auf. Weil ICMP-Pakete an Broadcast-Adressen nicht explizit erlaubt sind, würde die Firewall diese Pakete sowieso nicht durchlassen. Beachten Sie, dass hier alle ICMP-Typen blockiert werden, nicht nur echo-request. smurf benutzt ursprünglich zwar ping-Pakete, aber theoretisch lassen sich alle ICMP-Typen so missbrauchen. Beim Design einer Firewall kann man nicht vorsichtig genug sein.

```
# smurf-Angriff
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-d $BROADCAST_DEST -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-d $BROADCAST_DEST -j REJECT -1
```

```
# smurf-Angriff an Netmask
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp\
-d $NETMASK -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp\
-d $NETMASK -j REJECT -1
```

```
#smurf-Angriff an Netzadresse
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp\
-d $NETWORK -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp\
-d $NETWORK -j REJECT -1
```

Hinweis***smurf-Angriffe***

Senden Sie keine Broadcast-Pakete ins Internet! Der beschriebene smurf-Denial-of-Service wird erst möglich durch einen ping-Broadcast. Das CERT-Advisory CA-98.01.smurf von www.cert.org enthält nähere Informationen zu smurf-Angriffen.

3.4 Dienste auf festen, unprivilegierten Ports schützen

Besonders LAN-Dienste laufen häufig auf unprivilegierten Ports. Bei TCP-basierten Diensten lassen sich neue Verbindungsanfragen von bestehenden Verbindungen durch die SYN- und ACK-Bits unterscheiden. Verbindungsanfragen von außen sollten Sie aus Sicherheitsgründen verbieten. Die Sperre abgehender Verbindungen schützt Sie und andere vor Fehlern auf Ihrer Seite und hilft, etwaige interne Sicherheitsprobleme frühzeitig zu erkennen.

Hinweis***Offizielle Portnummern***

Portnummern werden bei der IANA registriert. Früher waren sie in der RFC 1700 (»Assigned Numbers«) verzeichnet; heute finden Sie die offizielle Liste unter <http://www.isi.edu/in-notes/iana/assignments/port-numbers>.

Vor welchen Fehlern sollten Sie sich schützen? Der schlimmste Fehler ist es, absichtlich oder unabsichtlich einen gefährlichen Dienst öffentlich anzubieten. In Kapitel 2 wird dieser Punkt ausführlich diskutiert. Ein häufiger Fehler besteht ferner darin, Pakete von rein lokalen Diensten versehentlich ins Internet zu routen, wo sie andere Leute stören. Gleichfalls sollten Sie fragwürdige Zugriffe unterbinden, die von einem Ihrer Systeme ausgehen, z.B. Portscans. Dabei spielt es keine Rolle, ob jemand diesen Traffic absichtlich generiert oder ob er versehentlich oder als Folge eines Einbruchs entsteht. Wenn Ihre Firewall alles sperrt, was nicht explizit erlaubt ist, sind Sie vor den meisten Fehlern dieser Art geschützt.

Mit einer Firewall-Policy von DENY können Sie hinter der Firewall viele private Dienste ohne große Sicherheitsrisiken betreiben. Bevor Fremde auf diese Dienste zugreifen können, müssen Sie das in der Firewall ausdrücklich erlauben. In der Praxis gilt das hier Gesagte allerdings nur eingeschränkt: TCP-Dienste auf privilegierten Ports sind zwar nur für die geschicktesten Hacker zugänglich, aber UDP-Dienste sind von Natur aus viel unsicherer, und einige Dienste laufen auch auf unprivilegierten Ports. RPC-Dienste – die meistens ebenfalls über UDP arbeiten – sind noch riskanter. RPC-Dienste werden irgendeinem Port zugeordnet, meist aus dem unprivilegierten Bereich. Der portmap-Dämon sorgt für die Zuordnung zwischen RPC-Servicenummer und der tatsächlich gewählten Portnummer. Ein Portscan kann die benutzten Ports auch ohne Zugriff auf portmap enthüllen.

Hinweis**Portscans**

Ein Portscan an sich ist nicht schädlich. Er ist das Produkt von Software zur Analyse von Netzwerken. Das Problem an Portscans besteht darin, dass sie heutzutage nur selten mit ehrenhaften Hintergedanken gestartet werden. Die betreffenden Leute analysieren ja schließlich nicht das eigene Netzwerk, sondern Ihres. Oft wollen sie damit eine Sicherheitslücke finden, die sie anschließend ausnutzen. Sicherlich, manche Leute sind einfach nur neugierig. Aber auch sie machen sich dadurch verdächtig.

3.4.1 Häufige TCP-Dienste auf unprivilegierten Ports

Besonders LAN-Dienste sind manchmal ganz offiziell einem unprivilegierten Port zugeordnet. Darüber hinaus sind manche Protokolle, z.B. FTP und IRC, sehr kompliziert und für einen Schutz durch Paketfilter schlecht geeignet. Die im Folgenden vorgestellten Firewall-Regeln hindern lokale und fremde Client-Software am Aufbau einer Verbindung zu einem dieser Ports.

FTP ist eines der Beispiele, warum eine reine Paketfilter-Firewall manchmal nicht ausreicht. Später wird das FTP-Protokoll noch genauer beschrieben. Im Moment sei nur soviel gesagt: Für eine FTP-Übertragung wird eine Verbindung zwischen zwei unprivilegierten Ports aufgebaut. Um das zu ermöglichen, muss die Firewall also den Aufbau einer Verbindung von einem beliebigen fremden unprivilegierten Port zu einem beliebigen unprivilegierten Port auf dem eigenen Computer erlauben. Der Nebeneffekt ist, dass damit plötzlich auch Verbindungen zu anderen lokalen Diensten erlaubt sind, sofern diese Dienste unprivilegierte Ports benutzen. Falls Sie solche lokalen Dienste verwenden, müssen Sie also Schutzregeln für diese Dienste aufstellen, bevor eine Regel für FTP in Kraft tritt.

Verbindungen zu Open Windows sperren (TCP-Port 2000)

Abgehende Verbindungen zu einem fremden Open-Windows-Display-Manager sollten Sie sperren. Durch die Option `-y` (SYN) gilt diese Regel nur für den Aufbau einer Verbindung zu einem fremden Computer. Bestehende Verbindungen zu einem fremden Client-Programm, das zufällig den unprivilegierten Port 2000 verwendet, sind nicht betroffen – dabei handelt es sich um eine Verbindung, die von dem fremden Programm zu Ihrer Maschine aufgebaut wurde.

```
OPENWINDOWS_PORT="2000"      # (TCP) OpenWindows
```

```
# Open-Windows-Verbindungsaufbau
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
-s $IPADDR \
-d $ANYWHERE $OPENWINDOWS_PORT -j REJECT
```

Ankommende Verbindungen zu Port 2000 müssen Sie nicht sperren. Linux enthält keinen Open-Windows-Display-Manager.

Verbindungen zu X-Windows sperren (TCP-Ports 6000 bis 6063)

Verbindungen zu X-Servern auf anderen Computern sollten Sie nur über `ssh` herstellen, das X-Windows vollautomatisch unterstützt. Die Option `-y` steht für das SYN-Bit und bewirkt, dass nur der Aufbau einer Verbindung zu einem fremden Rechner unterdrückt wird. Andere Verbindungen, bei denen ein fremder Rechner diesen Port zufällig als Client-Port verwendet, sind nicht betroffen.

Die Ports für X-Windows fangen mit dem ersten Server bei 6000 an. Weitere Server erhalten die folgenden Ports. Auf einem kleinen Computer läuft vermutlich immer nur ein X-Server gleichzeitig, d.h. nur Port 6000 ist belegt. Der höchste Port in diesem Bereich ist 6063; auf einem Computer können also theoretisch bis zu 64 separate X-Server gleichzeitig aktiv sein.

```
XWINDOW_PORTS="6000:6063" # (TCP) X-Windows
```

Die folgende Regel verhindert abgehende Verbindungen zu fremden X-Servern:

```
# X-Windows: Aufbau einer Verbindung zu einem fremden Server
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
-s $IPADDR \
-d $ANYWHERE $XWINDOW_PORTS -j REJECT
```

Als Nächstes werden alle Versuche gesperrt und aufgezeichnet, auf Ihren X-Windows-Server zuzugreifen. Lokale Verbindungen sind davon nicht betroffen, weil sie ja über das Loopback-Interface laufen.

```
# X-Windows: Aufbau einer Verbindung von außen zu einem unserer Server
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \
-d $IPADDR $XWINDOW_PORTS -j DENY -l
```

Verbindungen zu SOCKS-Servern sperren (TCP-Port 1080)

SOCKS ist ein lokaler Proxy, der von www.socks.nec.com kostenlos erhältlich ist. Entsprechend programmierte Software baut Verbindungen über diesen SOCKS-Server auf, anstatt einen Server direkt zu kontaktieren. Der SOCKS-Server stellt an ihrer Stelle die Verbindung zum Server im Internet her, wobei er sich gegenüber diesem Server wie ein normaler Client verhält.

Relativ häufig versuchen Leute, fremde SOCKS-Server zu missbrauchen. Die folgenden Regeln erlauben den Port 1080 als Client-Port für lokale oder fremde Programme, sperren ihn aber, wenn jemand ihn als Server-Port benutzen will.

```
SOCKS_PORT="1080" # (TCP) SOCKS
```

Die erste Regel verhindert abgehende Verbindungen von Ihrem Computer zu einem fremden SOCKS-Server:

```
# SOCKS: abgehende Verbindung
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
```

```
-s $IPADDR \  
-d $ANYWHERE $SOCKS_PORT -j REJECT -1
```

Die zweite Regel sperrt eingehende Verbindungen zu Ihrem SOCKS-Server (wenn Sie denn einen haben):

```
# SOCKS: ankommende Verbindung  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $IPADDR SOCKS_PORT -j DENY -1
```

3.4.2 Lokale UDP-Dienste auf unprivilegierten Ports

Firewall-Regeln lassen sich für das TCP-Protokoll viel detaillierter festlegen als für UDP – nur bei TCP wird für den Aufbau einer Verbindung ein bestimmter Handshake benutzt, den man herausfiltern kann. Im Gegensatz dazu kennt UDP keine Verbindungen. Daher sollten Sie den Zugang zu UDP-Diensten am besten pauschal sperren. Explizite Ausnahmen ermöglichen den Zugriff auf DNS und andere UDP-basierte Dienste, die Sie eventuell verwenden. Zum Glück sind UDP-Dienste häufig so gestaltet, dass sie nur zwischen einem Client und einem ganz bestimmten Server eingesetzt werden. In diesem Fall erlaubt man UDP durch die Firewall nur zu diesem einzelnen Server.

NFS ist der wichtigste UDP-Service aus diesem Bereich. NFS benutzt den unprivilegierten Port 2049, und zwar vorwiegend über UDP. NFS lässt sich zwar für TCP konfigurieren, aber das wird man in der Regel nicht tun.

NFS-Verbindungen sperren (UDP- und TCP-Port 2049)

Die erste Regel sperrt eingehende Pakete zum UDP-Port 2049. Wenn Sie NFS überhaupt nicht benötigen, ist sie überflüssig. Auf einer Firewall sollte NFS nicht aktiv sein, aber falls doch, sperren wir den Zugang:

```
NFS_PORT="2049" # (TCP/UDP) NFS  
  
# NFS: UDP-Pakete  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-d $IPADDR $NFS_PORT -j DENY -1
```

Die nächsten beiden Regeln kümmern sich um den selten eingesetzten TCP-Modus von NFS. Wie in den vorhergehenden Abschnitten zu TCP werden sowohl ankommende als auch abgehende Anfragen unterdrückt.

```
# NFS: TCP-Verbindungen  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $IPADDR $NFS_PORT -j DENY -1  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $ANYWHERE $NFS_PORT -j DENY -1
```

Hinweis**Die Protokoll-Tabellen für TCP und UDP**

Im Rest dieses Kapitels stelle ich Regeln vor, die den Zugriff auf ausgewählte Dienste erlauben. Die Client-Server-Kommunikation – sowohl über TCP als auch über UDP – ist immer Kommunikation in zwei Richtungen. Dabei wird ein Protokoll eingesetzt, das für den betreffenden Dienst spezifisch ist. Damit treten die Firewall-Regeln immer paarweise auf: eine für ankommende, eine für abgehende Pakete. Der Client formuliert eine Anfrage, der Server schickt eine Antwort.

Die Regeln für jeden Dienst sind immer einmal als Client-Regeln und einmal als Server-Regeln angegeben. Erstere beschreiben die Kommunikation zwischen einem lokalen Client und einem Server auf einem fremden System; zweitere benötigt man für die Interaktion zwischen fremden Clients und Servern auf dem eigenen Computer.

In der Transportschicht sind die Anwendungsdaten in TCP- oder UDP-Pakete verpackt. Weil jeder Dienst in der Anwendungsschicht über sein eigenes Protokoll verfügt, hängt es letztlich vom jeweiligen Dienst ab, wie die Interaktion zwischen Client und Server im Einzelnen abläuft.

Die Firewall-Regeln beschreiben den erlaubten Datenaustausch zwischen den beiden Endpunkten der Verbindung. Damit sorgen sie unter anderem auch dafür, dass die Protokolle zumindest auf der Paketebene eingehalten werden. Allerdings sind Firewall-Regeln (in der von `ipchains` benötigten Syntax) für uns Menschen nicht besonders leserlich. In den folgenden Abschnitten werde ich das jeweils behandelte Protokoll daher immer zuerst tabellarisch darstellen, bevor ich die `ipchains`-Regeln anführe.

Jede Zeile dieser Tabellen beschreibt einen Pakettyp. Für jeden dieser Pakettypen benötigt man eine eigene Firewall-Regel.

Die Spalten der Tabellen lauten wie folgt:

Beschreibung

enthält ein paar Stichworte zu Funktion und Herkunft des Pakets.

Protokoll

ist das Transportprotokoll: TCP, UDP oder ICMP.

Fremde IP

gibt an, welche Adresse bzw. welcher Adressbereich als fremde IP-Adresse stehen darf. Je nachdem, ob das Paket von unserem System abgeschickt wird oder nicht, kann sich das auf die Empfänger- oder auf die Absender-Adresse beziehen.

Fremder Port

beschreibt die Ports, die auf der Internet-Seite zulässig sind.

Chain

gibt an, in welcher Firewall-Chain – input oder output – dieses Paket erwartet wird. Ein Paket in der input-Chain kommt aus dem Internet, ein Paket in output kommt von Ihrem System und wird ins Internet geschickt.

Lokale IP

gibt an, welche Adresse bzw. welcher Adressbereich als lokale IP-Adresse stehen darf. Je nachdem, ob das Paket von unserem System abgeschickt wird oder nicht, kann sich das auf die Absender- oder auf die Empfänger-Adresse beziehen.

Lokaler Port

beschreibt die Ports, die auf unserer Seite zulässig sind.

TCP-Flags

sind, wie der Name schon sagt, nur für TCP-Pakete definiert. Für die Firewall-Regeln sind nur die Flags SYN und ACK von Bedeutung.

Durch die Angabe der Chain werden Pakete in ankommende und abgehende unterschieden. Adressen und Ports sind als »fremd« oder »lokal« klassifiziert, und zwar immer in Bezug auf das Netzwerkinterface Ihrer Maschine. Bei ankommenden Paketen beziehen sich also »fremde IP« und »fremder Port« auf die Absender-Felder im Paket-Header, während mit »lokaler IP« und »lokalem Port« die Empfänger-Felder gemeint sind. Bei abgehenden Paketen ist es umgekehrt: »fremde IP« und »fremder Port« meinen die Empfänger-Felder, »lokale IP« und »lokaler Port« die Absender-Felder.

In den wenigen Fällen, in denen ICMP-Pakete beteiligt sind, denken Sie daran, dass ICMP-Pakete im Gegensatz zu TCP- oder UDP-Paketen keinen Absender- und Empfänger-Port kennen. Dafür enthält der Header einen so genannten Nachrichtentyp. ICMP-Pakete werden ja nicht an Programme geschickt, die bestimmte Ports benutzen, sondern es handelt sich um Botschaften von einem Computer an einen anderen. In den Tabellen dieses Buches gebe ich den Nachrichtentyp in der Spalte mit dem Absender-Port an. Bei ankommenden ICMP-Paketen ist das die Spalte »fremder Port«, bei abgehenden ICMP-Paketen die Spalte »lokaler Port«.

3.5 Zwingend benötigte Dienste erlauben

Nur zwei Dienste sind wirklich unbedingt notwendig: der *Domain Name Service* DNS sowie der *ident*-Service zur Identifikation von Benutzern. DNS übersetzt symbolische Hostnamen und die entsprechenden IP-Adressen ineinander. Wenn DNS nicht verfügbar ist, können Sie auf fremde Hosts nicht sinnvoll zugreifen. *identd* liefert den Benutzernamen bzw. die numerische Benutzerkennung, die zu einer gegebenen Verbindung gehört. Diese Information wird häufig von Mailservern abgefragt. Sie müssen den *identd*-Service nicht unbedingt anbieten. Wollen Sie jedoch zeitaufwändige Timeouts vermeiden, sollten Sie eingehende Anfragen zumindest mit einer entsprechenden Fehlermeldung an den fremden Server ablehnen.

3.5.1 DNS (UDP- und TCP-Port 53)

Das Kommunikationsprotokoll von DNS verwendet sowohl UDP als auch TCP. Es existieren mehrere Arten von Verbindungen, darunter normale Verbindungen zwischen Client und Server, Informationsaustausch zwischen einem Forwarding-Server und einem regulären DNS-Server, sowie Verbindungen zwischen einem primären und einem sekundären Nameserver.

Anfragen erfolgen normalerweise über UDP, und zwar sowohl zwischen Client und Server als auch zwischen zwei Servern. Die UDP-Übertragung schlägt fehl, wenn die abgefragte Information zu groß ist, als dass sie in ein einzelnes UDP-DNS-Paket passen würde. In diesem Fall setzt der Server ein bestimmtes Flag im DNS-Header, das bedeutet, dass die Daten am Ende abgeschnitten worden sind. Wenn so etwas passiert, kann die Anfrage über TCP wiederholt werden. Bild 3.4 zeigt, wie sich in so einer Situation UDP und TCP zueinander verhalten. In der Praxis wird TCP fast nie benötigt. TCP wird für die Zone-Transfers zwischen primären und sekundären Nameservern eingesetzt.

Bei einem *Zone-Transfer* wird die vollständige Information über ein Netzwerk oder einen Teil davon (eine *Zone*) übertragen, für das oder den der Server autoritativ ist. *Autoritativ* bedeutet, dass der betreffende Server die offizielle Information für diese Zone aus erster Hand kennt und nicht nur eine beliebige Kopie davon speichert. Einen autoritativen Nameserver nennt man auch *primären* Nameserver. Die »sekundären Nameserver« oder »Backup-Nameserver« halten ihre DNS-Caches auf dem Laufenden, indem sie von Zeit zu Zeit einen Zone-Transfer vom primären Nameserver durchführen.

Beispielsweise könnte einer der Nameserver Ihres Internet-Providers der primäre, autoritative Nameserver für den Adressbereich des Providers sein. Die meisten Provider betreiben aber mehrere Nameserver, sowohl um die Last zwischen ihnen aufzuteilen, als auch um eine gewisse Ausfallsicherheit zu schaffen. Die anderen Nameserver sind dann sekundäre Nameserver und holen sich regelmäßig die aktuellsten Daten vom primären Server.

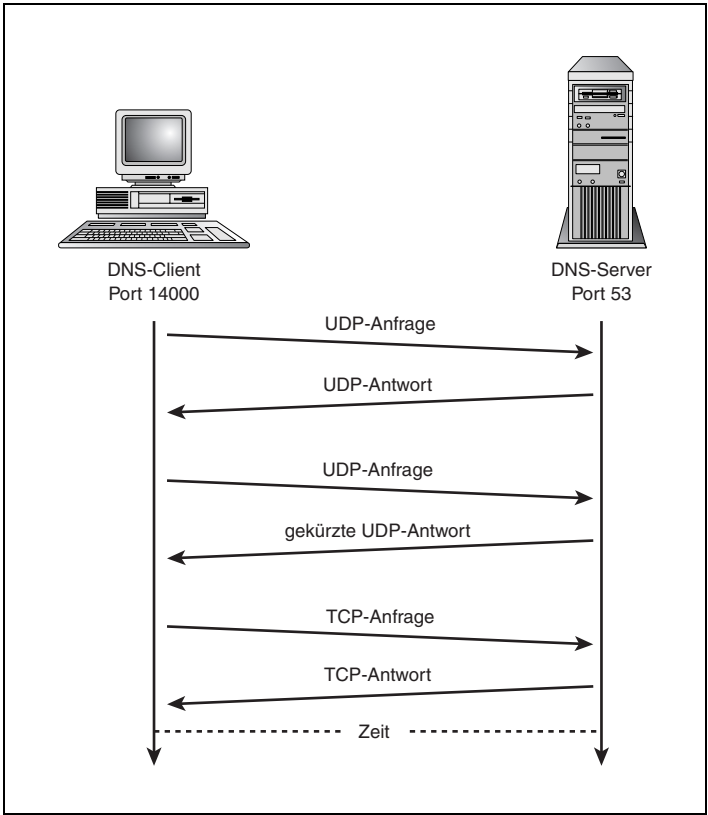


Bild 3.4: DNS-Anfragen

Zone-Transfers reichen ein wenig über das Thema dieses Buches hinaus. Ein kleines System ist wohl kaum der autoritative Nameserver für einen öffentlichen Adressbereich, und ebensowenig ein öffentlicher Backup-Server.

Tabelle 3.3 beschreibt das DNS-Protokoll, wie die Firewall-Regeln es berücksichtigen.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines lokalen Clients	UDP	NAMESERVER	53	output	IPADDR	1024:65535	-
Antwort des fremden Servers	UDP	NAMESERVER	53	input	IPADDR	1024:65535	-

Tab. 3.3: Das DNS-Protokoll

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines lokalen Clients	TCP	NAMESERVER	53	output	IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	NAMESERVER	53	input	IPADDR	1024:65535	ACK
Anfrage des lokalen Servers	UDP	NAMESERVER	53	output	IPADDR	53 oder andere	-
Antwort des fremden Servers	UDP	NAMESERVER	53	input	IPADDR	53 oder andere	-
Bitte um Zone-Transfer	TCP	Primärer Nameserver	53	output	IPADDR	1024:65535	Egal
Antwort auf Bitte um Zone-Transfer	TCP	Primärer Nameserver	53	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	UDP	DNS-Client	1024:65535	input	IPADDR	53	-
Antwort des lokalen Servers	UDP	DNS-Client	1024:65535	output	IPADDR	53	-
Anfrage eines fremden Clients	TCP	DNS-Client	1024:65535	input	IPADDR	53	Egal
Antwort des lokalen Servers	TCP	DNS-Client	1024:65535	output	IPADDR	53	ACK
Anfrage eines fremden Servers	UDP	DNS-Server	53 oder andere	input	IPADDR	53	-
Antwort des lokalen Servers	UDP	DNS-Server	53 oder andere	output	IPADDR	53	-
Jemand bittet um Zone-Transfer	TCP	Sekundärer Nameserver	1024:65535	input	IPADDR	53	Egal
Antwort auf Bitte um Zone-Transfer	TCP	Sekundärer Nameserver	1024:65535	output	IPADDR	53	ACK

Tab. 3.3: Das DNS-Protokoll (Forts.)

DNS-Anfragen als Client

Der DNS-Resolver ist kein separates Client-Programm, sondern ein Teil der Netzwerk-Bibliothek, die in Netzwerkprogramme eingebunden wird. Wenn ein Hostname nachgeschlagen wird, erfragt der Resolver ihn von einem `named`-Server. Viele kleine Systeme sind ausschließlich als DNS-Client konfiguriert, wobei der Server auf einer anderen Maschine läuft. Bei Privatpersonen ist der Nameserver meistens eine Maschine des Internet-Providers.

DNS schickt die Anfrage als UDP-Datagramm:

```
NAMESERVER="IP.Ihres.Name.servers"           # (TCP/UDP) DNS
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR $UNPRIVPORTS \
-d $NAMESERVER 53 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $NAMESERVER 53 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Falls die zurückgegebenen Daten zu groß sind und nicht mehr in ein einzelnes UDP-Paket passen, wiederholt DNS die Anfrage über eine TCP-Verbindung.

Die folgenden beiden Regeln sind für diesen seltenen Fall gedacht. Im normalen Betrieb werden Sie kaum einmal zum Zug kommen. Wahrscheinlich können Sie Ihr System über Monate problemlos laufen lassen, ohne TCP-Anfragen zuzulassen. Von Zeit zu Zeit – vielleicht ein- oder zweimal pro Jahr – würde DNS dann aber hängen bleiben, zumal wenn ein DNS-Server nicht ganz einwandfrei konfiguriert ist. Wichtiger sind diese Regeln für einen sekundären Nameserver, der vom primären Server einen Zone-Transfer anfordert.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d IP.Primärer.DNS.Server 53 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s IP.Primärer.DNS.Server 53 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

DNS-Anfragen von Server zu Server

So genannte *Peer-to-Peer*-Transaktionen finden zwischen zwei Servern statt. Beim DNS wird ein lokaler Server, wenn er die angeforderten Informationen nicht kennt, einen anderen Server kontaktieren und die Anfrage an ihn weiterleiten: man spricht vom *Forwarding*.

Die Einrichtung eines lokalen Forwarding-Nameservers kann Ihr Netzwerk deutlich beschleunigen. Bild 3.5 zeigt einen lokalen `named`, der für Caching und Forwarding konfiguriert ist. Dieser arbeitet gleichzeitig als lokaler DNS-Server und – gegenüber anderen DNS-Servern – als Client. DNS-Anfragen von einem Client und von einem Server unterscheiden sich in den Absender-Ports. Ein Client schickt die Anfrage von einem unprivilegierten Port, `named` hingegen vom DNS-Port 53. Ein zweiter Unterschied liegt darin, dass Anfragen von einem Server immer über UDP durchgeführt werden.

Hinweis

Das beschriebene Verhalten, dass `named` seine Anfragen von Port 53 abschickt, ist die Voreinstellung. Über die Direktive `query-source` in `/etc/named.conf` können Sie den Absender-Port frei einstellen.

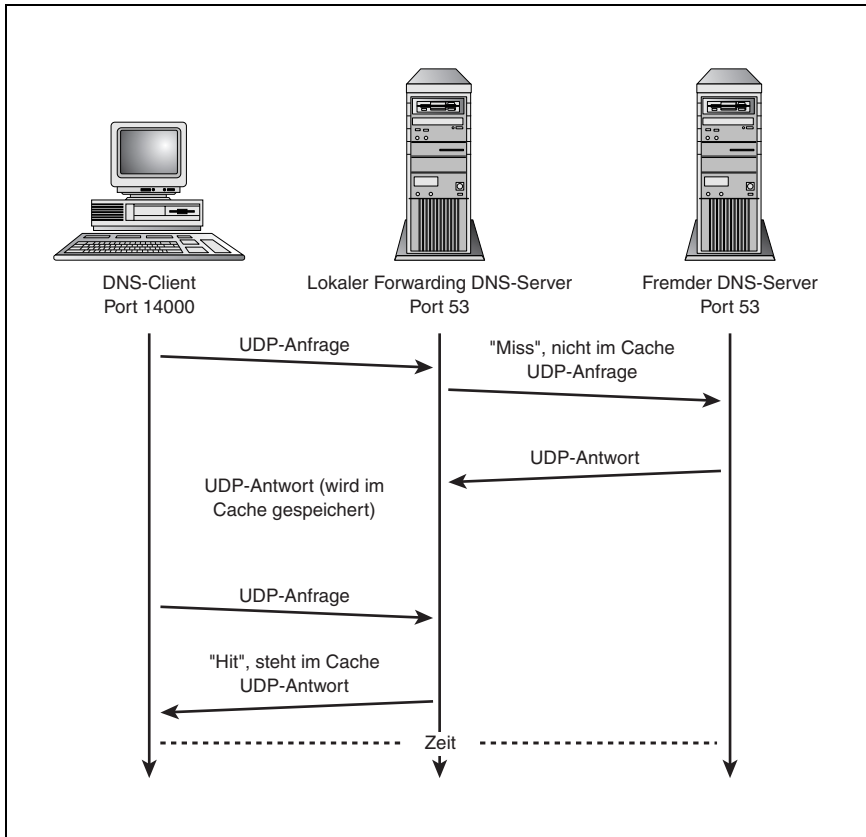


Bild 3.5: DNS-Anfrage über einen Forwarding-Nameserver

Lokale Anfragen werden in dieser Konstellation immer an den lokalen DNS-Server gesandt. Beim ersten Mal kennt `named` die entsprechenden Daten noch nicht und leitet die Anfrage daher an einen anderen Nameserver weiter. Die erhaltenen Informationen gibt es an den Client zurück und speichert sie im Cache. Wenn die gleiche Anfrage später noch einmal durchgeführt wird, kann `named` sie aus seinem lokalen Cache abrufen und muss die Leitung ins Netz diesmal nicht belasten.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 53 \  
-d $NAMESERVER 53 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $NAMESERVER 53 \  
-d $IPADDR 53 -j ACCEPT
```

Wie Sie Dritte auf Ihren DNS-Server zugreifen lassen

Für einen typischen Privatanwender gibt es eigentlich keinen guten Grund, warum beliebige Fremde auf Ihren DNS-Server zugreifen sollten. Wenn Sie kein Internet-Provider sind, werden die Clients Ihres DNS-Servers Maschinen Ihres eigenen LANs sein, Ihres Subnetzes, Routers, Adressbereichs oder wie auch immer Sie das nennen.

Unter diesen Voraussetzungen sollten Sie die Liste der erlaubten Clients also auf eine kleine Gruppe beschränken und Verbindungen nicht von überall zulassen:

```
# Client-Zugriff auf eigenen DNS  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s IPs.meiner.DNS.Clients $UNPRIVPORTS \  
-d $IPADDR 53 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 53 \  
-d IPs.meiner.DNS.Clients $UNPRIVPORTS -j ACCEPT  
  
# Peer-to-Peer-Server-Zugriff auf eigenen DNS  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s IPs.meiner.DNS.Clients 53 \  
-d $IPADDR 53 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 53 \  
-d IPs.meiner.DNS.Clients 53 -j ACCEPT
```

Die folgenden zwei Regeln gelten für Client-Anfragen, wenn die Antwortdaten zu groß für ein einzelnes UDP-Paket sind. Sie gelten ebenso für sekundäre Name-server, die einen Zone-Transfer vom primären Nameserver anfordern. TCP wird, wie bereits gesagt, fast ausschließlich für Zone-Transfers benutzt, die einige potenzielle Sicherheitsrisiken beinhalten. Daher sollten Sie besonderen Wert darauf legen, die Liste der erlaubten Clients einzuschränken.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s IPs.meiner.sekundären.Server $UNPRIVPORTS \  
-d $IPADDR 53 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 53 \  
-d IPs.meiner.sekundären.Server $UNPRIVPORTS -j ACCEPT
```

Warnung

DNS-Zone-Transfers über TCP

Kleine Sites sollten so mächtige Netzwerkdienste wie Zone-Transfers nicht anbieten. Manchmal gibt es sicherlich Ausnahmen. Wenn Sie so eine Ausnahme sind, oder wenn Sie DNS über TCP trotz aller Warnungen »einfach so« anbieten wollen, begrenzen Sie zumindest die Liste der erlaubten sekundären Server auf das Nötigste. Sie sollten Ihre privaten DNS-Informationen nur mit vertrauenswürdigen Sites teilen.

3.5.2 Der auth-Service (TCP-Port 113)

Der auth- oder identd-Service identifiziert Benutzer auf dem eigenen Computer gegenüber fremden Systemen. Er wird am häufigsten beim Versand von E-Mail oder beim Posten eines Usenet-Artikels benötigt. Auch einige FTP-Sites verlangen eine erfolgreiche auth-Anfrage. Dabei stellt der Server zu Protokollzwecken eine Anfrage an Ihren Rechner, welcher Benutzer denn diese Mail- oder News-Verbindung aufgebaut hat. Tabelle 3.4 stellt das Client-Server-Protokoll für den auth-Dienst dar.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage von lokalem Client	TCP	ANYWHERE	113	output	IPADDR	1024:65535	Egal
Antwort von fremdem Server	TCP	ANYWHERE	113	input	IPADDR	1024:65535	ACK
Anfrage von fremdem Client	TCP	ANYWHERE	1024:65535	input	IPADDR	113	Egal
Antwort von lokalem Server	TCP	ANYWHERE	1024:65535	output	IPADDR	113	ACK

Tab. 3.4: Das identd-Protokoll

Abgehende auth-Anfragen als Client

Ihre Maschine agiert als auth-Client, wenn Sie einen Mail- oder FTP-Server betreiben. Es gibt keinen Grund, warum Sie Ihrem System verbieten sollten, auth-Anfragen zu stellen.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 113 -j ACCEPT
```



```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 113 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Ankommende auth-Anfragen an einen Server auf Ihrem Computer

Ob Sie einen auth-Dienst anbieten sollten, wird immer noch und immer wieder diskutiert. Es gibt keinen wirklich durchschlagenden Grund dafür oder dagegen. Ein paar wenige FTP-Sites verlangen auth, dafür gibt auth Benutzerinformationen an andere weiter. Ob Sie diesen Service nun aktivieren oder nicht, Sie werden regelmäßig entsprechende Anfragen erhalten.

Wenn Sie in `/etc/inetd.conf` oder unterhalb von `/etc/rc.d` den `identd`-Server aktiviert haben, erlauben Sie mit den folgenden beiden Regeln ankommende `identd`-Anfragen:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR 113 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 113 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Falls Sie sich gegen diesen Dienst entscheiden, sollten Sie eingehende Anfragen nicht kommentarlos verwerfen. Damit würden Sie sich immer wieder lange Wartezeiten einhandeln, wenn beim Absenden von E-Mails oder News-Artikeln eine auth-Anfrage gestartet wird. Ihr Client erhält dann keine Bestätigung über das erfolgreiche Absenden der Nachricht, bis für die `identd`-Verbindung ein Timeout eintritt. Stattdessen sollten Sie die Anfrage mit einer Fehlermeldung zurückweisen (REJECT). In unseren Beispielen ist das der einzige Fall, in dem ein ankommendes Paket nicht mit DENY, sondern mit REJECT zurückgewiesen wird.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE \  
-d $IPADDR 113 -j REJECT
```

3.6 Häufige TCP-Dienste

Vermutlich wird niemand alle Dienste aus diesem Abschnitt aktivieren, aber jeder wird ein paar davon benötigen. Sie gehören zu den am häufigsten benutzten Diensten im Internet. Verstehen Sie das Folgende also bitte als Nachschlagewerk. Sie finden hier Regeln für:

- E-Mail
- Usenet
- telnet

- ssh
- ftp
- WWW
- finger
- whois
- gopher
- *Wide Area Information Service (WAIS)*

Es gibt natürlich noch viele weitere Dienste, auf die wir hier nicht eingehen können. Einige davon werden nur auf sehr spezialisierten Servern eingesetzt, andere von großen Firmen und Organisationen, einige nur auf lokalen Netzen.

3.6.1 E-Mail (TCP: SMTP – Port 25; POP – Port 110; IMAP – Port 143)

E-Mail braucht fast jeder. Freilich gibt es in Abhängigkeit von Ihrem Internet-Provider, Ihrer Internet-Anbindung und Ihren persönlichen Vorlieben zahlreiche Konfigurationsmöglichkeiten. Über das Internet wird E-Mail mit dem SMTP-Protokoll transportiert, dem der TCP-Port 25 zugeordnet ist. Lokal setzt man, je nach dem Provider und den Bedingungen vor Ort, eines von drei Protokollen ein: SMTP, POP oder IMAP.

SMTP ist ein allgemeines Protokoll zur Übertragung von E-Mail. Die Mail wird an den Empfänger-Computer ausgeliefert. Dieser entscheidet, ob die Mail zustellbar ist (d.h. an einen gültigen Benutzeraccount auf der Maschine adressiert ist), und schreibt sie gegebenenfalls in die Mailbox des Benutzers.

Mit den Protokollen POP und IMAP holt man Mail ab. POP arbeitet über den TCP-Port 110, IMAP über 143. Typischerweise geschieht über eines dieser Protokolle die Übertragung vom Internet-Provider zum Kunden. Beide Dienste benutzen Authentifizierung, d.h. der Kunde gibt beim Abholen von Mail seinen Benutzernamen und ein gültiges Passwort ein. Der Unterschied zwischen SMTP einerseits und POP bzw. IMAP andererseits liegt darin, dass SMTP ankommende Mails annimmt und sie in die Mailbox des Benutzers auf dem Computer seines Providers schreibt, während POP und IMAP die Mails aus dieser Provider-Mailbox abholen und an das Mail-Programm auf dem privaten PC übergeben.

Tabelle 3.5 zeigt, wie sich die genannten Protokolle auf Paketebene präsentieren.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Abgehende Mail senden	TCP	ANYWHERE	25	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	25	input	IPADDR	1024:65535	ACK
Ankommende Mail empfangen	TCP	ANYWHERE	1024:65535	input	IPADDR	25	Egal
Antwort des loka- len Servers	TCP	ANYWHERE	1024:65535	output	IPADDR	25	ACK
Anfrage eines loka- len Clients	TCP	POP_SERVER	110	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	POP_SERVER	110	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	POP-Client	1024:65535	input	IPADDR	110	Egal
Antwort des loka- len Servers	TCP	POP-Client	1024:65535	output	IPADDR	110	ACK
Anfrage eines loka- len Clients	TCP	IMAP_SERVER	143	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	IMAP_SERVER	143	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	IMAP-Client	1024:65535	input	IPADDR	143	Egal
Antwort des loka- len Servers	TCP	IMAP-Client	1024:65535	output	IPADDR	143	ACK

Tab. 3.5: Die E-Mail-Protokolle SMTP, POP und IMAP

Mailversand über SMTP (TCP-Port 25)

Der Versand von Mail erfolgt über SMTP. Aber an welchen SMTP-Server geben Sie Ihre Mails weiter? Die meisten Provider bieten ihren Kunden einen SMTP-Dienst an. Der Mailserver des Providers agiert dabei als Mail-Gateway. Er nimmt E-Mails vom Computer des Kunden an, sucht den Empfänger und übermittelt die Mail. Unter Unix können Sie aber auch Ihren eigenen Mailserver betreiben. Ihr eigener Computer wird dann die Mail an den Empfänger ausliefern.

Abgehende Mails über ein externes SMTP-Gateway verschicken

Wenn Sie Mail über ein externes SMTP-Gateway verschicken, leitet Ihre Client-Software alle abgehende Mail an den Mailserver Ihres Internet-Providers weiter. Ihr Provider fungiert dann als Gateway zum Rest der Welt. Ihr eigener Computer braucht sich nicht darum zu kümmern, wie er Empfängercomputer ausfindig macht und mit ihnen spricht. Der ISP dient als *Relay*.

Die folgenden zwei Regeln erlauben den dafür nötigen Netzwerkverkehr:

```
SMTP_GATEWAY="Server.des.Internet.Providers" # externer Mail-Server/  
Relay
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $SMTP_GATEWAY 25 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $SMTP_GATEWAY 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Mail an beliebige externe Mailserver verschicken

Die Alternative: Sie verzichten auf den Mailserver Ihres Providers und betreiben einen eigenen. Ihr lokaler Server sammelt alle abgehenden E-Mails, schlägt die Empfänger-Hosts im DNS nach und sendet die Mails direkt dorthin. Ihr Client-Programm schickt die Mail an Ihren eigenen SMTP-Server, nicht an den Ihres Providers.

Die folgenden beiden Regeln erlauben Mailversand direkt an die Empfängercomputer:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 25 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Hinweis

Proxies als Client und Server

Der meistgenutzte SMTP-Server ist sendmail, ein Proxy. Dem Client-Programm gegenüber, das die Mail abschickt, verhält es sich wie ein Mail-Server. Dem fremden Server gegenüber benimmt es sich wie ein Client. In solchen Kontexten können die Begriffe »Client« und »Server« etwas verwirrend sein. Je nachdem, mit wem es gerade spricht, kann sendmail beide Funktionen erfüllen.

Mailempfang

Wie Sie Ihre Mail empfangen, hängt von Ihrer konkreten Situation ab. Wenn Sie einen vollwertigen eigenen Mailserver betreiben, kann ankommende Mail direkt an Ihren Computer geleitet werden. Wenn Sie Ihre Mail hingegen von einer Mailbox bei Ihrem Internet-Provider abholen, werden Sie eventuell einen POP- oder

IMAP-Client einsetzen, je nachdem, wie die Mailbox beim ISP konfiguriert ist und welche Services Ihr ISP anbietet.

Mail als SMTP-Server empfangen (TCP-Port 25)

Wenn alle Maschinen in der Welt Mail direkt an Ihren eigenen Computer senden dürfen, aktivieren Sie `sendmail` und benutzen die folgenden Regeln:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 25 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 25 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Falls Sie diesen lokalen Mailserver lieber privat halten wollen und stattdessen eine Mailadresse in der Arbeit oder bei Ihrem Internet-Provider als öffentliche Adresse benutzen, können Sie Mail an Ihren lokalen Server weiterleiten lassen. In diesem Fall verändern Sie die zwei Regeln so, dass Verbindungen nicht mehr von überall angenommen werden, sondern nur noch von den Computern, die Mail an Sie weiterleiten.

Mail über einen POP-Client abholen (TCP-Port 110)

Sehr viele Leute holen ihre Mail über das *Post Office Protocol* POP von ihrem Internet-Provider oder von der Arbeit ab. Dazu müssen Sie abgehende Verbindungen zu einem fremden Server erlauben.

Als Server-Adresse sollten Sie einen einzelnen Hostnamen oder eine IP-Adresse benutzen, nicht einfach »ANYWHERE«.

```
POP_SERVER="IP.meines.POP.Servers"      # externer POP-Server
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $POP_SERVER 110 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $POP_SERVER 110 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Mail über einen IMAP-Client abholen (TCP-Port 143)

IMAP ist eine Alternative zu POP. Wenn Sie Ihre Mail von Ihrem Internet-Provider über IMAP abholen wollen, erlauben Sie abgehende Verbindungen.

Auch hier ist die Server-Adresse wieder ein einzelner Hostname oder eine IP-Adresse, nicht »ANYWHERE«.

```
IMAP_SERVER="IP.meines.IMAP.Servers"      # externer IMAP-Server
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $IMAP_SERVER 143 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IMAP_SERVER 143 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Beispiele

Dieser Abschnitt beschreibt vier typische Kombinationen, wie E-Mail häufig versandt und empfangen wird:

- Mailversand als SMTP-Client; Mailempfang als POP-Client
- Mailversand als SMTP-Client; Mailempfang als IMAP-Client
- Mailversand als SMTP-Client; Mailempfang als SMTP-Server
- Mailversand als SMTP-Server; Mailempfang als SMTP-Server

In den ersten zwei Beispielen verlassen Sie sich gänzlich auf die E-Mail-Dienste Ihres Internet-Providers. Im dritten Fall senden Sie abgehende Mails zwar über den Provider, empfangen eingehende Mails aber direkt über einen eigenen SMTP-Server. Im vierten Fall betreiben Sie sowohl für Versand als auch für Empfang von E-Mail einen völlig eigenständigen Server.

Mailversand als SMTP-Client; Mailempfang als POP-Client

Wenn Sie Mail als SMTP-Client versenden und als POP-Client abholen, verlassen Sie sich in Bezug auf Ihre E-Mail vollständig auf einen anderen Computer. Dieser betreibt einen SMTP-Server, der Ihre abgehenden Mails weiterleitet, und einen POP-Server, von dem Sie ankommende Mails abholen.

```
SMTP_GATEWAY="IP.meines.SMTP.Servers"      # externer Mailserver/Relay
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $SMTP_GATEWAY 25 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $SMTP_GATEWAY 25 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
POP_SERVER="IP.meines.POP.Servers"         # externer POP-Server
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $POP_SERVER 110 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $POP_SERVER 110 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Mailversand als SMTP-Client; Mailempfang als IMAP-Client

Wenn Sie Mail als SMTP-Client versenden und als IMAP-Client abholen, verlassen Sie sich in Bezug auf Ihre E-Mail vollständig auf einen anderen Computer. Dieser betreibt einen SMTP-Server, der Ihre abgehenden Mails weiterleitet, und einen IMAP-Server, von dem Sie ankommende Mails abholen.

```
SMTP_GATEWAY="IP.meines.SMTP.Servers"      # externer SMTP-Server/Relay
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $SMTP_GATEWAY 25 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $SMTP_GATEWAY 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
IMAP_SERVER="IP.meines.IMAP.Servers"      # externer IMAP-Server
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $IMAP_SERVER 143 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IMAP_SERVER 143 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Mailversand als SMTP-Client; Mailempfang als SMTP-Server

Wenn Sie Mail als SMTP-Client versenden und als SMTP-Server empfangen, verlassen Sie sich in Bezug auf abgehende E-Mails auf einen anderen Computer, der die Mails weiterleitet. Lokal läuft `sendmail`, bei dem fremde Hosts direkt E-Mail abliefern können. Der Mailversand erfolgt über Ihren Internet-Provider, den Mailempfang hingegen kann der lokale `sendmail`-Dämon selbst bewerkstelligen.

```
SMTP_GATEWAY="IP.meines.SMTP.Servers"      # externer Mailserver/Relay
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $SMTP_GATEWAY 25 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $SMTP_GATEWAY 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR 25 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 25 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Mailversand als SMTP-Server; Mailempfang als SMTP-Server

Wenn Sie Mail als SMTP-Server sowohl versenden als auch empfangen, sorgen Sie selbst für alle Ihre E-Mail-Services. Ihr lokaler `sendmail`-Dämon verschickt abgehende Mails direkt an die jeweiligen Empfänger und kümmert sich um Empfang und Zustellung eingehender Mails.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 25 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR 25 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 25 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Einen Mailserver für andere betreiben

Ein kleines System wird normalerweise keinen öffentlich zugänglichen POP- oder IMAP-Server betreiben. Eine Ausnahme wäre beispielsweise gegeben, wenn Sie ein paar Freunden Mail-Dienste anbieten wollen, z.B. weil der Mail-Service ihres Internet-Providers vorübergehend nicht verfügbar ist. Dabei ist es sehr wichtig, dass Sie einschränken, von welchen Clients Ihr Computer Verbindungen annehmen wird. Das sollte sowohl auf der Ebene des Paketfilters als auch auf der der Serverkonfiguration geschehen. Daneben sollten Sie sich überlegen, ob Sie die Authentifizierung verschlüsseln, oder sogar das Abholen von Mail nur über eine `ssh`-Verbindung erlauben.

Ein POP-Server

POP-Server gehören zu den drei häufigsten und erfolgreichsten Angriffspunkten für Eindringlinge.

Wenn Sie einen lokalen Mailserver mit einem `popd` für Maschinen auf Ihrem LAN betreiben wollen, brauchen Sie die Regeln aus diesem Beispiel *nicht*! In diesem Fall sollten Sie alle Verbindungen aus dem Internet zu Ihrem POP-Server sperren.

Falls Sie ausgewählten Clients aus dem Internet trotzdem einen POP-Service anbieten müssen, ermöglichen das die nächsten zwei Regeln. Verbindungen werden nur für die genannten IP-Adressen erlaubt.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.POP.Clients $UNPRIVPORTS \
-d $IPADDR 110 \
-j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 110 \
-d IPs.meiner.POP.Clients $UNPRIVPORTS \
-j ACCEPT
```

Ein IMAP-Server

IMAP-Server gehören zu den drei häufigsten und erfolgreichsten Angriffspunkten für Eindringlinge.

Wenn Sie einen lokalen Mailserver mit einem `imafd` für Maschinen auf Ihrem LAN betreiben wollen, brauchen Sie die Regeln aus diesem Beispiel *nicht*! In diesem Fall sollten Sie alle Verbindungen aus dem Internet zu Ihrem IMAP-Server sperren.

Falls Sie ausgewählten Clients aus dem Internet trotzdem einen IMAP-Service anbieten müssen, erlauben das die nächsten zwei Regeln. Verbindungen sind nur für die genannten IP-Adressen gestattet.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.IMAP.Clients $UNPRIVPORTS \
-d $IPADDR 143 \
-j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 143 \
-d IPs.meiner.IMAP.Clients $UNPRIVPORTS \
-j ACCEPT
```

3.6.2 Usenet News (NNTP: TCP-Port 119)

Usenet (auch kurz »News« genannt) wird über NNTP übertragen, über TCP und den Port 119. Für das Lesen von News und das Posten von Artikeln benutzen Sie einen lokalen News-Client. Nur wenige Systeme brauchen die Regeln für News-server. Tabelle 3.6 präsentiert das Protokoll in tabellarischer Form.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines loka- len Clients	TCP	NEWS_SERVER	119	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	NEWS_SERVER	119	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	NNTP-Client	1024:65535	input	IPADDR	119	Egal
Antwort des loka- len Servers	TCP	NNTP-Client	1024:65535	output	IPADDR	119	ACK
Anfrage des loka- len Servers	TCP	Newsfeed	119	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	Newsfeed	119	input	IPADDR	1024:65535	ACK

Tab. 3.6: Das NNTP-Protokoll

News als Client lesen und schreiben

Die Client-Regeln erlauben Verbindungen zum Newsserver Ihres Internet-Provi-
 ders. Sie gelten sowohl für Lesen als auch für Schreiben.

```

NEWS_SERVER="IP.meines.News.Servers"           # externer Newsserver

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $NEWS_SERVER 119 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $NEWS_SERVER 119 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
  
```

Einen lokalen Newsserver für fremde Clients zugänglich machen

Auf einem kleinen System wird es wohl kaum einen öffentlich zugänglichen
 Newsserver geben; selbst ein lokaler Newsserver wird nur selten gebraucht. In
 den seltenen Ausnahmefällen sollte man die Firewall so konfigurieren, dass nur
 handverlesene Clients zugreifen dürfen:

```

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.News.Clients $UNPRIVPORTS \
-d $IPADDR 119 \
-j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 119 \
-d IPs.meiner.News.Clients $UNPRIVPORTS \
-j ACCEPT
  
```

Echte Newsfeeds für einen lokalen Usenet-Server zulassen

Daheim werden Sie wohl kaum einen Newsserver aufsetzen, der irgendwoher einen echten Peer-to-Peer-Feed bezieht. Während Newsserver früher ziemlich frei zugänglich waren, gibt es heute nur noch wenige offene Server. Grund ist neben der Spam-Problematik auch die Überlastung der Server.

Wenn Ihre Site groß oder reich genug für einen echten Newsserver ist, dann brauchen Sie von irgendwo einen Newsfeed. Die nächsten beiden Regeln erlauben Ihrem eigenen Newsserver, seinen Feed von einem anderen Server zu beziehen. Der lokale Server greift auf den anderen Server als Client zu. Der einzige Unterschied zu der Client-Regel weiter oben besteht in der Adresse des fremden Rechners.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d IP.meines.News.Feeds 119 \  
-j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s IP.meines.News.Feeds 119 \  
-d $IPADDR $UNPRIVPORTS \  
-j ACCEPT
```

3.6.3 telnet (TCP-Port 23)

Über viele Jahre war `telnet` der *de facto*-Standard für Fernwartung über das Internet. Das Netz wird immer aggressiver, und so sieht man `telnet` heute als recht unsicher an, weil alle Daten im Klartext übertragen werden. Je nach Ausstattung auf der anderen Seite kann `telnet` für Sie trotzdem die einzige Methode sein, über das Internet auf anderen Computern zu arbeiten. Wenn irgend möglich, sollten Sie statt `telnet` aber auf ein verschlüsseltes Protokoll wie `ssh` zurückgreifen.

Die hier angegebenen Client- und Server-Regeln erlauben den Zugriff von und nach überall. Wenn Sie `telnet` einsetzen, ist das nicht unbedingt notwendig; wahrscheinlich können Sie die Liste der erlaubten externen Adressen einschränken.

Tabelle 3.7 zeigt, wie sich das Protokoll auf der Paketebene darstellt.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines lokalen Clients	TCP	ANYWHERE	23	output	IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	23	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	telnet-Client	1024:65535	input	IPADDR	23	Egal
Antwort des lokalen Servers	TCP	telnet-Clients	1024:65535	output	IPADDR	23	ACK

Tab. 3.7: Das telnet-Protokoll

Zugriff auf fremde Sites

Wenn Sie mit `telnet` auf andere Computer zugreifen wollen, benutzen Sie die nächsten beiden Regeln. Eventuell sollten Sie nur den Zugang zu denjenigen Sites erlauben, auf denen Ihre Benutzer tatsächlich arbeiten.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 23 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 23 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Zugriff auf den lokalen Server durch fremde Clients

Auch wenn Sie mit `telnet` auf anderen Computern arbeiten, heißt das noch nicht unbedingt, dass Sie auch ankommende Verbindungen zu Ihrem `telnet`-Server zulassen müssen. Falls doch, verwenden Sie diese Regeln:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 23 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 23 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Wie schon weiter oben gesagt: Statt den Zugriff pauschal von überall her zu gestatten, sollten Sie besser für jeden Host bzw. für jedes Netzwerk, von dem Verbindungen erlaubt sind, separate Regeln erstellen.

3.6.4 ssh (TCP-Port 22)

Die *Secure Shell* `ssh` wird wegen der amerikanischen Exportbeschränkungen für Verschlüsselungssoftware bei den meisten Distributionen nicht mitgeliefert. Die Software ist aus dem Internet aber kostenlos erhältlich. `ssh` sollte, wann immer möglich, anstelle von `telnet` benutzt werden, wenn man über das Internet auf anderen Computern arbeitet. `ssh` verwendet auf beiden Seiten Authentifizierungsverfahren für die beteiligten Computer *und* Benutzer. Alle Daten werden in verschlüsselter Form übertragen. Darüber hinaus hat `ssh` noch viele weitere Features: Es kann vollautomatisch X-Windows-Verbindungen über das Netz leiten. Auch die Umleitung von `ftp` und anderen TCP-basierten Verbindungen über den sicheren `ssh`-Tunnel ist möglich. Wenn der Kommunikationspartner `ssh` zulässt, können Sie alle TCP-Verbindungen mit `ssh` durch die Firewall leiten. `ssh` fungiert dann sozusagen als ein Virtual Private Network (VPN) des kleinen Mannes.

Welche Ports `ssh` benutzt, lässt sich frei konfigurieren. Die hier gezeigten Regeln beziehen sich auf die Voreinstellungen. Dabei geschieht der Verbindungsaufbau zwischen einem unprivilegierten Port beim Client und dem Port 22 des Servers. Auf der Clientseite wird die Verbindung dann auf einen privilegierten Port im Bereich von 1023 bis 513 (absteigend) übertragen, wobei der erste freie Port benutzt wird. Dieser Portwechsel erlaubt die Authentifizierung über `.rhosts` und `hosts.equiv`. Statt dieses Modells kann der `ssh`-Client auch ausschließlich unprivilegierte Ports benutzen. Der Server akzeptiert Verbindungen sowohl von unprivilegierten als auch von privilegierten Ports.

Hinweis

ssh, die `tcp_wrapper` und `rhost-Authentifizierung`

ssh lässt sich nicht direkt über die `tcp_wrapper` starten, aber man kann es so übersetzen, dass es die in `/etc/hosts.allow` und `/etc/hosts.deny` definierten Zugriffsbeschränkungen berücksichtigt.

Eine Authentifizierung über `.rhosts` und `hosts.equiv` sollten Sie auf einer Firewall nicht erlauben. Die in Kapitel 8 vorgestellten Analysetools zur Systemsicherheit warnen Sie, falls die genannten Dateien auf Ihrem Computer existieren.

Mehr Informationen zu `ssh` erhalten Sie von www.ssh.fi.

Die im Folgenden gezeigten Regeln für Client und Server erlauben den Zugang von und nach überall. In der Praxis wird man die externen Adressen einschränken.

Tabelle 3.8 zeigt das Protokoll in tabellarischer Form.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines loka- len Clients	TCP	ANYWHERE	22	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	22	input	IPADDR	1024:65535	ACK
Anfrage eines loka- len Clients	TCP	ANYWHERE	22	output	IPADDR	513:1023	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	22	input	IPADDR	513:1023	ACK
Anfrage eines fremden Clients	TCP	ssh-Clients	1024:65535	input	IPADDR	22	Egal
Antwort des loka- len Servers	TCP	ssh-Clients	1024:65535	output	IPADDR	22	ACK
Anfrage eines fremden Clients	TCP	ssh-Clients	513:1023	input	IPADDR	22	Egal
Antwort des loka- len Servers	TCP	ssh-Clients	513:1023	output	IPADDR	22	ACK

Tab. 3.8: Das ssh-Protokoll

Wenn Sie einen privilegierten Client-Port für die laufende Verbindung wählen, wird der erste freie Port zwischen 1023 und 513 benutzt. Der erlaubte Portbereich entspricht dadurch der Zahl der gleichzeitig möglichen ssh-Verbindungen in jede Richtung.

```
SSH_PORTS="1020:1023"           # (TCP) vier Verbindungen gleichzeitig
```

Zugriff auf fremde ssh-Server

Diese Regeln gestatten den Zugang zu fremden Sites über ssh:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 22 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $SSH_PORTS \
-d $ANYWHERE 22 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $IPADDR $SSH_PORTS -j ACCEPT
```

Zugriff auf den lokalen ssh-Server durch fremde Clients

Die folgenden Regeln ermöglichen fremden Clients Verbindungen zu Ihrem sshd-Server.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 22 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 22 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $SSH_PORTS \
-d $IPADDR 22 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 22 \
-d $ANYWHERE $SSH_PORTS -j ACCEPT
```

3.6.5 ftp (TCP-Ports 20 und 21)

ftp zählt nach wie vor zu den am häufigsten benutzten Methoden, mit denen man Dateien von einem Computer auf den anderen überträgt. Mittlerweile gibt es auch viele Web-Schnittstellen zu ftp.

ftp benutzt zwei privilegierte Ports, einen für Kommandos und einen für Daten. Über Port 21 wird die Verbindung zum Server hergestellt; er dient der Übermittlung von Kommandos. Separate Kanäle für den eigentlichen Datentransfer verwenden den Port 20. Hier werden Auflistungen von Verzeichnisinhalten sowie die Dateien selbst übertragen.

ftp kennt zwei Modi für den Datenaustausch zwischen Client und Server: den *normalen* oder *aktiven* sowie den *passiven* Modus. Der aktive Modus ist die Voreinstellung, wenn Sie das Programm ftp aufrufen. Der passive Modus wurde später entwickelt; er kommt z.B. bei der Verwendung eines Web-Browsers zum Einsatz. Einige wenige FTP-Sites unterstützen nur einen der beiden Modi.

Die Protokolltabelle finden Sie in Tabelle 3.9.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines lokalen Clients	TCP	ANYWHERE	21	output	IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	21	input	IPADDR	1024:65535	ACK
Datenkanalaufbau vom fremden Server, aktiver Modus	TCP	ANYWHERE	20	input	IPADDR	1024:65535	Egal
Antwort auf Datenkanalaufbau durch lokalen Client, aktiver Modus	TCP	ANYWHERE	20	output	IPADDR	1024:65535	ACK
Datenkanalaufbau zum fremden Server, passiver Modus	TCP	ANYWHERE	1024:65535	output	IPADDR	1024:65535	Egal
Antwort auf Datenkanalaufbau durch fremden Server, passiver Modus	TCP	ANYWHERE	1024:65535	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	input	IPADDR	21	Egal
Antwort des lokalen Servers	TCP	ANYWHERE	1024:65535	output	IPADDR	21	ACK
Datenkanalaufbau vom lokalen Server, aktiver Modus	TCP	ANYWHERE	1024:65535	output	IPADDR	20	Egal
Antwort auf Datenkanalaufbau durch fremden Client, aktiver Modus	TCP	ANYWHERE	1024:65535	input	IPADDR	20	ACK
Datenkanalaufbau zum lokalen Server, passiver Modus	TCP	ANYWHERE	1024:65535	input	IPADDR	1024:65535	Egal
Antwort auf Datenkanalaufbau durch lokalen Server, passiver Modus	TCP	ANYWHERE	1024:65535	output	IPADDR	1024:65535	ACK

Tab. 3.9: Das ftp-Protokoll

Zugriffe lokaler Clients auf fremde ftp-Server

Fast jeder wird ftp-Zugang zu fremden Filearchiven erlauben. Abgehende Verbindungen von lokalen Clients zu fremden Servern sollten die Firewall also passieren können.

Abgehende Kontrollverbindungen

Zunächst aktivieren Sie abgehende Kontrollverbindungen zu fremden ftp-Servern:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 21 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 21 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Datenkanäle für den aktiven Modus

Im normalen oder aktiven Modus baut der ftp-Server die Datenverbindung zum ftp-Client auf:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE 20 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 20 -j ACCEPT
```

Dieser ungewöhnliche »Rückruf«, bei dem der Server die Datenverbindung zum Client herstellt, ist mit ein Grund dafür, warum ftp auf der Ebene eines Paketfilters so schwer abzusichern ist. Sie können niemals sicher sein, dass die ankommende Verbindung wirklich von dem angesprochenen ftp-Server kommt. Wenn Sie lokale Dienste auf unprivilegierten Ports (z.B. X-Windows oder ein SOCKS-Server) nicht zuvor explizit gesperrt haben, dann erlauben die beiden Regeln für Datenkanäle im aktiven Modus auch den Zugriff auf solche Dienste.

Datenkanäle für den passiven Modus

Die folgenden beiden Regeln erlauben den neueren passiven Modus für Datenkanäle, wie er z.B. von Web-Browsern benutzt wird:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Der passive Modus ist sicherer als der aktive Modus, weil der ftp-Client sowohl die Kontroll- als auch die Datenverbindung aufbaut. Allerdings besteht die Datenverbindung zwischen zwei unprivilegierten Ports, ist also ebenfalls eine Art Freibrief, diesmal allerdings für Aktivitäten von Ihrem System.

Zugriff auf Ihren lokalen ftp-Server

Ob Sie einen öffentlichen ftp-Dienst anbieten sollen, ist eine schwierige Entscheidung. Obwohl es im Internet nur so von ftp-Sites wimmelt, gestaltet sich die richtige Konfiguration nicht ganz einfach. Viele Sicherheitslücken drohen.

Wenn Sie nur ein paar Dateien auf Ihrer Maschine öffentlich zugänglich machen wollen, dann ist ein Web-Server eventuell eine bessere Lösung. Möchten Sie, dass Andere Dateien bei Ihnen ablegen können, dann sollten Sie den Zugang zum ftp-Server auf den drei Ebenen Firewall – tcp_wrapper – ftp-Serverkonfiguration radikal einschränken.

Ein ftp-Server, der Schreibzugriffe erlaubt, sollte diese zumindest für anonyme Benutzer verbieten. Nur ausgewählte Benutzer von ausgewählten Sites sollten auf streng kontrollierte und begrenzte Bereiche Ihrer Dateisysteme schreiben dürfen. Diese Punkte werden in Kapitel 7 näher diskutiert.

Ankommende Kontrollverbindungen

Zwei Regeln für ankommende Kontrollverbindungen:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 21 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 21 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Datenkanäle für den aktiven Modus

Zwei Regeln für den »Rückruf« Ihres Servers an den Client, zum Aufbau eines Datenkanals:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR 20 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 20 -j ACCEPT
```

Datenkanäle für den passiven Modus

Zwei Regeln für den Aufbau eines Datenkanals im passiven Modus, vom fremden Client zu Ihrem Server:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Warnung

Achtung: Benutzen Sie tftp nicht im Internet!

tftp ist eine einfachere Form von ftp, die ohne Authentifizierung arbeitet und UDP benutzt. tftp soll das Booten von Routern und Workstations ohne eigene Bootmedien über ein LAN ermöglichen. Manche Leute verwechseln es mit einer möglichen Alternative zu ftp. Das ist es aber nicht; benutzen Sie tftp keinesfalls über das Internet!

3.6.6 World Wide Web

Das World Wide Web basiert auf dem Hypertext Transfer Protocol HTTP. Client und Server verwenden ganz normale TCP-Verbindungen. Einige Spezialprotokolle erlauben neben dem allgemeinen HTTP-Zugang auch einen gesicherten Zugang über SSL sowie den Zugriff auf und über Proxies, die z.B. Internet-Provider anbieten. Diese Spezialprotokolle verwenden eigene Ports.

Normaler HTTP-Zugang (TCP-Port 80)

Normalerweise greift man auf Web-Dienste über den http-Port 80 zu. Tabelle 3.10 zeigt das Protokoll auf Paketebene.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines lokalen Clients	TCP	ANYWHERE	80	output	IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	80	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	input	IPADDR	80	Egal
Antwort des lokalen Servers	TCP	ANYWHERE	1024:65535	output	IPADDR	80	ACK

Tab. 3.10: Das HTTP-Protokoll

Zugriff auf fremde Websites als Client

In der heutigen Welt ist es praktisch unvorstellbar, dass jemand von einem privaten Computer *nicht* auf das WWW zugreifen wollte. Die folgenden zwei Regeln erlauben das:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 80 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 80 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Zugriff fremder Clients auf einen lokalen Web-Server

Wenn Sie einen eigenen Web-Server betreiben, erlauben die folgenden Regeln alle typischen Zugriffe auf Ihre Site. Mehr brauchen die wenigsten Sites!

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 80 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 80 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Abgesicherter Web-Zugriff über SSL (TCP-Port 443)

Secure Socket Layer oder SSL dient dem abgesicherten, verschlüsselten Zugang zu WWW-Seiten. Das SSL-Protokoll, dargestellt in Tabelle 3.11, benutzt den TCP-Port 443. Typischerweise wird es bei kommerziellen Web-Sites eingesetzt, auf denen man etwas kaufen kann, beim Online-Banking oder für geschützte Bereiche, in denen man persönliche Informationen eingibt.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines lokalen Clients	TCP	ANYWHERE	443	output	IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	443	input	IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	input	IPADDR	443	Egal
Antwort des lokalen Servers	TCP	ANYWHERE	1024:65535	output	IPADDR	443	ACK

Tab. 3.11: Das SSL-Protokoll

Zugriff auf fremde Websites über SSL als Client

Die meisten Leute wollen früher oder später abgesichert auf Websites zugreifen:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 443 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 443 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Zugriff fremder Clients auf einen lokalen SSL-Webserver

Wenn Sie E-Commerce in der einen oder anderen Form betreiben, müssen Sie vermutlich auch ankommende Verbindungen über SSL zulassen, mit denen jemand auf geschützte Bereiche Ihrer Website zugreift. Wenn das nicht der Fall ist, brauchen Sie keine Regeln für den lokalen Server.

Die normale Apache-Distribution enthält bereits Unterstützung für SSL, aber alle sichereren SSL-Module fehlen wegen der amerikanischen Exportbeschränkungen. Es gibt allerdings sowohl kostenlose als auch kommerzielle Pakete, die Apache um leistungsfähigen SSL-Support ergänzen. Von www.apache.org erhalten Sie nähere Informationen zum Thema.

Zwei Regeln erlauben den Zugriff auf Ihren Webserver über SSL:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 443 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 443 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Zugriff auf Web-Proxies (TCP-Ports 8008 und 8080)

Die meisten Internet-Provider betreiben öffentlich zugängliche Web-Proxies. Sofern Ihr Provider nicht automatisch alle Zugriffe auf das Web auf seinen Proxy umleitet, müssen Sie Ihren Browser entsprechend konfigurieren. Proxies verwenden häufig den unprivilegierten Port 8080 oder auch 8008, je nachdem, wie der Provider sie einrichtet. Die Verwendung eines Proxies erlaubt schnelleren Zugriff auf Seiten, die sich bereits im Zwischenspeicher befinden, und eine gewisse Anonymität durch die Zwischenschaltung des Proxy-Servers. Die Verbindungen zum Web-Server werden nicht mehr direkt durch Ihren Browser aufgebaut, sondern in Ihrem Namen durch den Proxy.

Tabelle 3.12 zeigt die für den Zugriff auf Proxies notwendigen Regeln in tabellarischer Form.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines loka- len Clients	TCP	WEB_PROXY_ SERVER	WEB_ PROXY_ PORT	output	IPADDR	1024:65535	Egal
Antwort des Proxy- Servers	TCP	WEB_PROXY_ SERVER	WEB_ PROXY_ PORT	input	1024:65535	ACK	

Tab. 3.12: Das Protokoll für den Zugriff auf Web-Proxies

Adresse und Portnummer des Proxies erhalten Sie vom Provider. Die Regeln für einen Client sind:

```

WEB_PROXY_SERVER="mein.WWW.Proxy"           # Proxy des Internet-Providers
WEB_PROXY_PORT="Port.meines.Proxies"         # zugehöriger Port
                                              # meistens 8008 oder 8080

```

```

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $WEB_PROXY_SERVER $WEB_PROXY_PORT -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $WEB_PROXY_SERVER $WEB_PROXY_PORT \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

```

3.6.7 finger (TCP-Port 79)

Von einem rein technischen Standpunkt aus gesehen, ist `finger` relativ harmlos. Wegen wachsender Bedenken, persönliche Informationen im Internet zu veröffentlichen, rät man heute aber allgemein davon ab, den `finger`-Service anzubieten. `finger` liefert Informationen über die Benutzer des Systems, einschließlich Loginnamen, Realnamen, aktive Sitzungen, ungelesene E-Mail und Forward-Adressen. Oft wird zusätzlich der Inhalt einer vom Benutzer eingerichteten Datei (`.plan`) angezeigt, die oft Telefonnummer, Postadresse, aktuelle Projekte, Urlaubspläne und so weiter enthält.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines loka- len Clients	TCP	ANYWHERE	79	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	79	input	IPADDR	1024:65535	ACK

Tab. 3.13: Das `finger`-Protokoll

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines fremden Clients	TCP	finger-Client	1024:65535	input	IPADDR	79	Egal
Antwort des lokalen Servers	TCP	finger-Client	1024:65535	output	IPADDR	79	ACK

Tab. 3.13: Das finger-Protokoll (Forts.)

Client-Zugriff auf fremde finger-Server

Der Zugriff auf fremde finger-Server ist unschädlich. Benutzen Sie folgende Regeln:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 79 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 79 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Zugriff auf einen lokalen finger-Server durch fremde Clients

Falls Sie sich dafür entscheiden, einen finger-Service nach außen anzubieten, empfehle ich, dass Sie zumindest nur bestimmte Hosts darauf zugreifen lassen. Der Zugang zu einem finger-Server kann auf den Ebenen von Firewall und tcp_wrappern blockiert werden.

Die folgenden beiden Regeln ermöglichen ausgewählten Hosts den Verbindungsaufbau zu finger.

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.finger.Clients $UNPRIVPORTS \
-d $IPADDR 79 \
-j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 79 \
-d IPs.meiner.finger.Clients $UNPRIVPORTS \
-j ACCEPT
```

3.6.8 whois (TCP-Port 43)

whois greift auf die Registrierungsdatenbanken von Network Solutions, DENIC, RIPE u.ä. zu und liefert technische und administrative Informationen zu IP-Adressen, Host- und Domainnamen in für Menschen lesbarer Form.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines loka- len Clients	TCP	ANYWHERE	43	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	43	input	IPADDR	1024:65535	ACK

Tab. 3.14: Das whois-Protokoll

Die folgenden beiden Regeln erlauben die Abfrage der whois-Server:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 43 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 43 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

3.6.9 gopher (TCP-Port 70)

Der Informationsdienst `gopher` ist für ASCII-Terminals weiterhin verfügbar, wurde aber weithin durch Web-basierte Suchmaschinen und Hypertext-Links abgelöst. Es ist nicht sehr wahrscheinlich, dass Sie auf einem Linux-System `gopher` anstelle eines Webserverns anbieten werden. Die Regeln für Server drucke ich daher gar nicht mit ab.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines loka- len Clients	TCP	ANYWHERE	70	output	IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	70	input	IPADDR	1024:65535	ACK

Tab. 3.15: Das gopher-Protokoll

Die folgenden Regeln erlauben die Verbindung zu einem Server:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 70 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 70 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```


3.6.10 WAIS (TCP-Port 210)

Wide Area Information Servers (WAIS) nennt man heutzutage einfach »Suchmaschinen«. Die meisten Webbrowser wie Netscape beinhalten den notwendigen Client-Code sowie ein grafisches Interface zu WAIS.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines lokalen Clients	TCP	ANYWHERE	210	output	IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	210	input	IPADDR	1024:65535	ACK

Tab. 3.16: Das WAIS-Protokoll

Die nächsten beiden Regeln ermöglichen Client-Zugriffe auf WAIS-Server:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 210 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 210 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

3.7 Häufige UDP-Dienste

Das zustandsfreie UDP-Protokoll ist von Haus aus unsicherer als das verbindungsbasierte TCP. Daher sperren viele sicherheitsbewusste Sites den Zugang zu UDP-Diensten vollständig oder schränken ihn so weit wie möglich ein. Natürlich sind DNS-Zugriffe über UDP notwendig, aber die wenigen Nameserver, auf die man zugreifen muss, kann man auch explizit in der Firewall angeben. Aus diesem Grund behandle ich in diesem Abschnitt nur drei Dienste:

- traceroute
- *Dynamic Host Configuration Protocol* (DHCP)
- *Network Time Protocol* (NTP)

3.7.1 traceroute (UDP-Port 33434)

Bei traceroute handelt es sich um einen UDP-Service, der absichtlich ICMP-Fehlermeldungen provoziert, und zwar vom Empfängersystem eine Meldung vom Typ *destination-unreachable* (Port nicht gefunden), von den dazwischenliegenden Systemen hingegen vom Typ *time-exceeded* (Zeitüberschreitung). trace-

route wertet diese Fehler aus und kann dadurch den Weg anzeigen, den ein Paket von Ihrem Computer zu einem bestimmten Empfänger zurücklegt.

Die Firewall, die wir in diesem Kapitel entwickeln, sperrt per Voreinstellung alle ankommenden UDP-Pakete an die Ports, die `traceroute` normalerweise verwendet. Daher wird Ihr Computer eine ICMP-Antworten auf eingehende `traceroute`-Anfragen verschicken.

Tabelle 3.17 zeigt alle Pakettypen, die an `traceroute` mitwirken.

Beschreibung	Protokoll	Fremde IP	Fremder Port oder ICMP-Typ	Chain	Lokale IP	Lokaler Port oder ICMP-Typ
Abgehendes <code>traceroute</code> -Paket	UDP	ANYWHERE	33434:33523	output	IPADDR	32769:65535
time-exceeded (Meldung einer Zwischenstation)	ICMP	ANYWHERE	11	input	IPADDR	-
destination-unreachable (Port nicht gefunden; Meldung des Empfängersystems)	ICMP	ANYWHERE	3	input	IPADDR	-
Ankommendes <code>traceroute</code> -Paket	UDP	(Provider)	32769:65535	input	IPADDR	33434:33523
time-exceeded (wir sind Zwischenstation)	ICMP	(Provider)	-	output	IPADDR	11
destination-unreachable (Port nicht gefunden; wir sind Empfänger)	ICMP	(Provider)	-	output	IPADDR	3

Tab. 3.17: Das `traceroute`-Protokoll

Man kann `traceroute` für die Verwendung beliebiger Ports oder Portbereiche konfigurieren. Daher ist es nicht möglich, alle ankommenden `traceroute`-Pakete zu blockieren, indem man nur bestimmte Ports angibt. Oft werden jedoch Absender-Ports zwischen 32769 und 65535 und Empfänger-Ports zwischen 33434 und 33523 benutzt. Für diese Voreinstellungen definieren wir symbolische Konstanten:

```

TRACEROUTE_SRC_PORTS="32769:65535"           # Traceroute: Absender-Ports
TRACEROUTE_DEST_PORTS="33434:33523"          # Traceroute: Empfänger-Ports

```

Abgehende `traceroutes`

Wenn Sie selbst `traceroute` verwenden wollen, schalten Sie die entsprechenden UDP-Ports frei. Beachten Sie, dass `traceroute` nur funktioniert, wenn Sie die

ICMP-Nachrichtentypen `time-exceeded` und `destination-unreachable` von überall her akzeptieren.

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR $TRACEROUTE_SRC_PORTS \
-d $ANYWHERE $TRACEROUTE_DEST_PORTS -j ACCEPT
```

Ankommende traceroutes

Weil `traceroute` auch unter den UDP-Diensten zu den unsicheren gehört und dazu missbraucht werden kann, andere UDP-Dienste anzugreifen, gestattet das folgende Beispiel ankommende `traceroutes` nur von den Computern Ihres Internet-Providers:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $MY_ISP $TRACEROUTE_SRC_PORTS \
-d $IPADDR $TRACEROUTE_DEST_PORTS -j ACCEPT
```

Hier müssen abgehende ICMP-Nachrichten vom Typ `time-exceeded` bzw. `destination-unreachable` erlaubt sein.

3.7.2 Zugriff auf den DHCP-Server Ihres Internet-Providers (UDP-Ports 67 und 68)

Wenn DHCP-Übertragungen zwischen Ihrem Computer und Ihrem Provider überhaupt vorkommen, wird es sich immer um einen Datenaustausch zwischen einem Client auf Ihrer Seite und dem Server des Providers handeln. Dabei bezieht der DHCP-Client vom zentralen Server eine vorübergehend gültige, dynamische IP-Adresse aus dem IP-Kontingent des Providers.

Falls Ihr Provider Ihnen eine dynamische Adresse zuweist und das nicht über PPP geschieht, muss der DHCP-Client-Dämon (entweder `dhcpcd` oder `pump`) laufen. Es kommt durchaus vor, dass auf dem lokalen Subnetz des Providers auch völlig unsinnige DHCP-Nachrichten herumfliegen, wenn ein anderer Kunde aus Versehen einen DHCP-Server startet. Deshalb ist es bei DHCP besonders wichtig, dass Sie den Datenaustausch nur zwischen Ihrem Computer und dem DHCP-Server des Providers zulassen.

In Tabelle 3.18 sehen Sie die Beschreibungen der DHCP-Nachrichtentypen, wie sie in RFC 2131 (»Dynamic Host Configuration Protocol«) aufgeführt sind.

DHCP-Nachricht	Beschreibung
DHCPDISCOVER	Broadcast-Nachricht eines Clients, der einen Server sucht.
DHCPOFFER	Antwort eines Servers auf DHCPDISCOVER, der damit Konfigurationseinstellungen anbietet.

Tab. 3.18: DHCP-Nachrichtentypen

DHCP-Nachricht	Beschreibung
DHCPREQUEST	Broadcast-Nachricht eines Clients an alle Server: (a) Bitte an einen Server, die angebotenen Einstellungen zu übertragen; implizit werden damit die Angebote der anderen Server abgelehnt; oder (b) Bitte um Bestätigung, dass eine früher zugewiesene Adresse z.B. nach einem Reboot noch gültig ist; oder (c) Verlängerung der Gültigkeit einer bestimmten Adresse.
DHCPACK	Server an Client: Übermittlung von Konfigurationseinstellungen, einschließlich der Netzwerkadresse.
DHCPNAK	Server an Client: Die vom Client angegebene Netzwerkadresse stimmt nicht; z.B. der Client ist jetzt in einem anderen Subnetz, oder die Gültigkeit der Adresse ist abgelaufen.
DHCPDECLINE	Client an Server: Netzwerkadresse wird bereits von jemand anders benutzt.
DHCPRELEASE	Client an Server: Netzwerkadresse wird nicht mehr weiter benötigt.
DHCPINFORM	Client an Server: Bitte um lokale Konfigurationseinstellungen, wobei der Client bereits eine manuell konfigurierte IP-Adresse besitzt (in Red Hat 6.0 nicht unterstützt).

Tab. 3.18: DHCP-Nachrichtentypen (Forts.)

1. Beim Start des DHCP-Clients schickt er eine Broadcast-Nachricht vom Typ DHCPDISCOVER (»DHCP: entdecken«), um herauszufinden, ob DHCP-Server verfügbar sind.
2. Jeder Server, der diese Nachricht sieht, kann mit einem DHCPOFFER (»DHCP: anbieten«) antworten, in dem er angibt, welche Parameter er für den Client auf Lager hat.
3. Der Client schickt nun *eine* DHCPREQUEST-Nachricht (»DHCP: Anfrage«), ebenfalls an alle Server. Mit dieser Nachricht akzeptiert er einen der Server und informiert gleichzeitig die anderen darüber, dass er ihr Angebot ablehnt.
4. Der auserwählte Server reagiert darauf mit einem DHCPACK und bestätigt gleichzeitig die ursprünglich angebotenen Parameter. An diesem Punkt ist die Adresszuteilung abgeschlossen.
5. Von Zeit zu Zeit muss der Client die Gültigkeit seiner IP-Adresse verlängern und sendet dafür eine Nachricht vom Typ DHCPREQUEST.
6. Wenn die Gültigkeit verlängert werden kann, antwortet der Server mit DHCPACK.
7. Anderenfalls beginnt der Client die gesamte Initialisierung von vorne.

Tabelle 3.19 zeigt das DHCP-Protokoll in gewohnter Weise.

DHCP ist viel komplizierter als hier dargestellt. Die Grundlagen der Client-Server-Kommunikation gibt diese kurze Zusammenfassung aber wieder.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
DHCPDISCOVER; DHCPREQUEST	UDP	255.255.255.255	67	output	0.0.0.0	68
DHCPPOFFER	UDP	0.0.0.0	67	input	255.255.255.255	68
DHCPPOFFER	UDP	DHCP_SERVER	67	input	255.255.255.255	68
DHCPREQUEST; DHCPDECLINE	UDP	DHCP_SERVER	67	output	0.0.0.0	68
DHCPACK; DHCPNAK	UDP	DHCP_SERVER	67	input	(Subnetz/Maske)	68
DHCPACK	UDP	DHCP_SERVER	67	input	IPADDR	68
DHCPREQUEST; DHCPRELEASE	UDP	DHCP_SERVER	67	output	IPADDR	68

Tab. 3.19: Das DHCP-Protokoll

Die folgenden Firewall-Regeln ermöglichen die Kommunikation zwischen Ihrem DHCP-Client und einem fremden Server:

```
DHCP_SERVER="IP.meines.DHCP.Servers"

# Initialisierung oder Re-Initialisierung:
# Wir haben noch keine Adresse, oder ihre Gültigkeit ist ausgelaufen

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
        -s $BROADCAST_0 68 \
        -d $BROADCAST_1 67 -j ACCEPT

# Eine neue Adresse holen

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -s $BROADCAST_0 67 \
        -d $BROADCAST_1 68 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -s $DHCP_SERVER 67 \
        -d $BROADCAST_1 68 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
        -s $BROADCAST_0 68 \
        -d $DHCP_SERVER 67 -j ACCEPT

# Wir sollten nun unsere IP-Adresse wechseln; die Nachricht
# geht schon an unsere neue Adresse, bevor der DHCP-Client
# das Update erhalten hat.
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $DHCP_SERVER 67 \
-d $MY_ISP 68 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $DHCP_SERVER 67 \
-d $IPADDR 68 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR 68 \
-d $DHCP_SERVER 67 -j ACCEPT
```

Beachten Sie, dass DHCP-Nachrichten nicht komplett auf Ihren DHCP-Server beschränkt werden können. Während der Initialisierung kennt der Client weder seine IP-Adresse noch die des Servers; Pakete werden daher nicht gezielt versandt, sondern gehen an die Broadcast-Adresse.

3.7.3 Network-Time-Server abfragen (UDP-Port 123)

Network-Time-Server bieten z.B. über NTP öffentlichen Zugang zu genauen Uhren. So hat man immer eine genaue Systemuhr, selbst wenn die BIOS-Uhr des Computers nicht so genau geht, und kann Zeit und Datum beim Booten oder nach einem Stromausfall neu setzen. Anwender eines kleinen Systems sollten den Dienst nur als Client nutzen; schließlich hat nicht jedermann eine Atomuhr herumliegen.

xntpd ist der Server-Dämon. Er versorgt nicht nur Clients mit der aktuellen Uhrzeit, sondern stellt auch eine Verbindung zu anderen Servern her. Der dadurch mögliche kleine Gewinn an Präzision ist für kleine Sites aber kaum notwendig. Das Client-Programm heißt ntpdate; es beherrscht den Austausch sowohl mit Servern als auch mit anderen Clients. Vermutlich reicht es für Ihre Zwecke völlig aus. In Tabelle 3.20 ist nur das Client-Server-Protokoll aufgeführt.

Beschreibung	Proto-koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
Anfrage eines lokalen Clients	UDP	Time-Server	123	output	IPADDR	1024:65535
Antwort des Servers	UDP	Time-Server	123	input	IPADDR	1024:65535

Tab. 3.20: Das NTP-Protokoll

Als Client können Sie mit ntpdate regelmäßig eine Reihe von öffentlichen Time-Servern abfragen, z.B. aus einem cron-Job heraus. Die IP-Nummern der Time-Server geben Sie über Firewall-Regelpaare wie dieses an:

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR $UNPRIVPORTS \
-d IP.eines.Time.Servers 123 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s IP.eines.Time.Servers 123 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

3.8 Abgewiesene Pakete protokollieren

Wenn Sie eine `ipchains`-Regel um die Option `-l` ergänzen, werden alle passenden Pakete protokolliert. Bei einigen der hier vorgestellten Beispiele ist diese Option bereits enthalten, z.B. bei den Regeln gegen gefälschte Adressen.

Man kann Regeln auch nur zu dem Zweck definieren, dass bestimmte Paketarten protokolliert werden. Dabei interessiert man sich meistens für verdächtige Pakete, die einen Portscan oder Abtastversuch repräsentieren. Wenn Sie einzelne Pakettypen protokollieren wollen, müssen Sie das explizit angeben, damit die Pakete nicht am Ende der Chain per Voreinstellung kommentarlos verworfen werden.

Welche Pakete Sie ins `syslog` aufnehmen, liegt bei Ihnen. Manche Leute wollen alle abgewiesenen Pakete aufzeichnen. Anderswo würde so ein Ansatz die Logfiles schnell überfüllen. Einige Leute verlassen sich darauf, dass diese Pakete ohnehin abgewiesen wurden, und kümmern sich überhaupt nicht weiter darum. Andere wollen mehr wissen über offensichtliche Portscans oder bestimmte andere Paketsorten.

Weil immer die erste passende Regel angewendet wird, könnten Sie natürlich pauschal alle abgewiesenen Pakete (auf die also noch keine Regel gepasst hat) mit einer Regel wie dieser protokollieren:

```
ipchains -A input -i $EXTERNAL_INTERFACE -j DENY -l
```

In manchen Fällen resultieren daraus zu viele Protokolleinträge, oder zumindest zu viele uninteressante. Z.B. wäre es denkbar, alle abgewiesenen ICMP-Pakete aufzuzeichnen, `ping`-Anfragen aber nicht, weil `ping` so häufig ist:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \  
-s $ANYWHERE 1:7 -d $IPADDR -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \  
-s $ANYWHERE 9:18 -d $IPADDR -j DENY -l
```

Oder Sie könnten alle abgewiesenen TCP-Pakete protokollieren, bei UDP aber nur diejenigen an privilegierte Ports:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-d $IPADDR $PRIVPORTS -j DENY -l
```

Vielleicht wollen Sie auch über Zugriffsversuche auf alle Ports informiert werden, außer bei ein paar sehr häufigen Ports, zu denen Sie keine Dienste anbieten:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 0:19 -j DENY -1
```

ftp, telnet und ssh auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 24 -j DENY -1
```

SMTP auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 26:78 -j DENY -1
```

finger und WWW auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 81:109 -j DENY -1
```

pop-3 und sunrpc auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 112:136 -j DENY -1
```

NetBIOS auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 140:142 -j DENY -1
```

IMAP auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 144:442 -j DENY -1
```

SSL auslassen

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 444:65535 -j DENY -1
```

Regeln für UDP

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-d $IPADDR 0:110 -j DENY -1
```



```
# sunrpc auslassen

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR 112:160 -j DENY -1

# SNMP auslassen

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR 163:634 -j DENY -1

# NFS mountd auslassen

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR 636:5631 -j DENY -1

# pcAnywhere auslassen

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR 5633:31336 -j DENY -1

# BackOrifice auslassen

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR 31338:33433 -j DENY -1

# traceroute auslassen
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -s $ANYWHERE 32679:65535 \
        -d $IPADDR 33434:33523 -j DENY

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR 33434:65535 -j DENY -1
```

3.9 Zugriff für problematische Sites pauschal sperren

Wenn jemand Ihre Maschine immer wieder scannt oder sie anderweitig belästigt, sollten Sie ihn eventuell komplett aussperren – zumindest, bis das Problem behoben ist.

Freilich, es kann lästig sein, Ihr `rc.firewall`-Skript zu diesem Zweck ständig zu überarbeiten. Alternativ bietet es sich an, eine separate Datei mit ausgesperrten Sites anzulegen. Diese Regeln werden an den Anfang und nicht ans Ende der `input`-Chain gestellt und gelten somit auch dann, wenn eine spätere Regel den Zugriff erlauben würde. Hier schlage ich für diese Datei den Namen `/etc/rc.d/rc.firewall.blocked` vor. Sicherheitshalber sollten Sie in `rc.firewall` prüfen, ob diese Datei existiert, bevor Sie sie ausführen:

```
# Pakete von ausgesperrten Hosts ablehnen
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi
```

Ein Beispiel für eine Regel in `rc.firewall.blocked` wäre:

```
ipchains -I input -i $EXTERNAL_INTERFACE -s Adresse/Maske -j DENY
```

Jedes Paket vom angegebenen Adressbereich wird abgewiesen, unabhängig von Protokoll und Ports.

Damit sind alle Firewall-Regeln definiert. Sobald sie auch im Kernel installiert sind, können Sie Ihr Linux-System ans Internet anschließen und halbwegs darauf vertrauen, dass Sie vor den meisten Angriffen von außen geschützt sind.

3.10 LAN-Zugang

Wenn Ihre Firewall sich zwischen dem Internet und einem LAN befindet, dürfen die Geräte aus dem LAN weder auf das interne Netzwerkinterface der Firewall noch auf das Internet zugreifen. Die Details der Firewall-Installation für ein LAN behandelt Kapitel 4. Für eine kleine Site, vor allem in einem Privathaushalt, ist die in Kapitel 4 vorgestellte Firewall-Architektur aber Overkill. Typischerweise reicht die in diesem Kapitel entwickelte Firewall völlig aus, auch für ein kleines Büro.

Für ein LAN hinter der Firewall braucht man nur noch ein paar zusätzliche Regeln, damit die Computer des lokalen Netzes auf die Firewall zugreifen dürfen, und damit Pakete zwischen LAN und Internet hin- und hergeroutet werden. Eine so verschaltete Firewall nennt man *Bastion-Firewall*; man spricht auch von einem *abgeschirmten Subnetz*.

3.10.1 LAN-Zugang zum internen Netzwerkinterface der Firewall

In einem Privathaushalt oder einem kleinen Büro muss man den Zugang aus dem lokalen Netz zur Firewall vermutlich nicht einschränken. Die folgenden beiden Regeln öffnen alle Kommunikationskanäle zwischen LAN und Firewall:

```
LAN_INTERFACE_1="eth1"           # Interface zum LAN
LAN__IPADDR_1="192.168.1.1"      # interne IP-Adresse des Interfaces

LAN_1="192.168.1.0/24"           # (privater) Adressbereich des LANs

ipchains -A input -i $LAN_INTERFACE_1 \
    -s $LAN_1 -j ACCEPT

ipchains -A output -i $LAN_INTERFACE_1 \
    -d $LAN_1 -j ACCEPT
```

Beachten Sie, dass diese Regeln zunächst nur den Zugang zur Firewall aus dem LAN zulassen. Ins Internet gelangt dadurch noch niemand. Weil eine Firewall *per definitionem* nicht dynamisch routet und auch nicht automatisch statische Routes verwendet (außer, wenn die Maschine grob falsch konfiguriert ist), benötigt man noch weitere Regeln, die lokale Pakete nach außen weiterleiten.

3.10.2 LAN-Zugriffe auf das Internet: IP-Forwarding und -Masquerading

Zum jetzigen Zeitpunkt sind ausgewählte Ports für die Kommunikation zwischen fremden Maschinen und dem externen Interface der Firewall offen, lokale Aktivitäten zwischen dem LAN und dem internen Interface der Firewall gänzlich uneingeschränkt. Allerdings haben die Computer auf dem lokalen Netz noch keinen Zugang zum Internet. Dies aktiviert man in zwei Schritten: Daten vom und zum lokalen Netz müssen einerseits weitergeleitet (*Forwarding*) und andererseits maskiert werden (*Masquerading*). Forwarding bedeutet, dass die Firewall die Pakete von lokalen Maschinen nimmt und ins Internet weiterleitet und umgekehrt. Masquerading heißt, dass sie die Header der Pakete dabei so umschreibt, dass fremde Maschinen denken, die Pakete seien in Wirklichkeit von der Firewall abgeschickt worden.

IP-Forwarding ist ein Kernel-Service und versetzt ein Linux-System in die Lage, als Router zwischen zwei Netzwerken zu agieren, d.h. Pakete von dem einen Netzwerk in das andere weiterzuleiten. Für ein LAN müssen Sie IP-Forwarding in der Netzwerkkonfiguration im Abschnitt über das Routing aktivieren. Wenn Sie in der Firewall allerdings die Policy `DENY` gewählt haben, dürfen die weitergeleiteten Pakete nicht vom einen zum anderen Netzwerkinterface laufen, ohne dass Sie das zusätzlich explizit erlauben.

Private Systeme werden interne Daten jedoch nicht direkt forwarden wollen. Wenn IP-Adressen aus den privaten Bereichen der Klassen A, B oder C verwendet werden, müssen diese durch die öffentliche IP-Adresse der Firewall ersetzt werden. Das leistet Masquerading (ein weiterer Kernel-Dienst), oder Sie setzen Proxy-Programme ein. Pakete mit einer privaten IP-Adresse dürfen nicht ins Internet geleitet werden; sollte das dennoch geschehen, werden sie vermutlich nicht bis zum Empfänger gelangen. Selbst wenn Sie im lokalen Netz offiziell registrierte, statische IP-Adressen einsetzen, sind Masquerading und Proxies sehr gute Methoden, Ihre internen Computer sicher und transparent vom Internet abzusichern.

Auf der Ebene von `ipchains` erscheinen Forwarding und Masquerading als verschiedene Aspekte des gleichen Services. (In Wirklichkeit handelt es sich um getrennte Mechanismen, die aber im Benutzerinterface – der Firewall-Administration – vereinigt sind.) Forwarding leitet LAN-Pakete vom internen Netzwerkinterface der Firewall über das externe Interface ins Internet weiter. Bevor ein Paket an das externe Interface gelangt, ersetzt der Masquerading-Dienst die Absender-

adresse mit der externen, öffentlichen IP der Firewall. Forwarding und Masquerading zusammen verwandeln Ihre Firewall in einen Router, der gleichzeitig als Filter und Proxy fungiert.

Das folgende Beispiel zeigt, wie Sie alle internen Pakete über das externe Interface weiterleiten. Die ACCEPT- und DENY-Regeln für die output-Chain des externen Interfaces werden *nach* den Forwarding-Regeln angewendet, d.h. obwohl bei Forwarding und Masquerading noch alles erlaubt ist, dürfen nur diejenigen Pakete das externe Interface verlassen, die die Firewall-Regeln erfüllen.

```
ipchains -A forward -i $EXTERNAL_INTERFACE -s $LAN_1 -j MASQ
```

In Masquerading-Regeln dürfen Sie Absender- und Empfänger-Adressen und -Ports verwenden, genau wie bei den anderen Regeln auch. Die Option für das Netzwerkinterface bezieht sich auf das externe Interface, zu dem die Pakete weitergeleitet werden, nicht auf das lokale Interface. Obwohl das Beispiel alles erlaubt, könnten Sie ebenso gut Regeln implementieren, die nur bestimmte Dienste, nur TCP-Pakete o.ä. weiterleiten und maskieren.

3.11 Die Firewall installieren

Die Installation eines Shell-Skriptes ist einfach. Das Skript sollte root gehören

```
chown root.root /etc/rc.d/rc.firewall
```

und nur durch root beschreibbar und ausführbar sein. Im Idealfall dürfen andere es nicht einmal lesen:

```
chmod u=rwx,go-rwx /etc/rc.d/rc.firewall
```

Wenn Sie die Firewall initialisieren wollen, rufen Sie das Skript von der Kommandozeile auf. Sie können das jederzeit tun. Ein Neustart des Computers ist nicht nötig.

```
sh /etc/rc.d/rc.firewall
```

Wie Sie das Skript beim Booten automatisch ausführen lassen, hängt von der Art Ihrer IP-Adresse ab: Ist sie offiziell registriert und damit statisch, oder wird sie dynamisch vergeben?

3.11.1 Installation einer Firewall mit einer statischen IP-Adresse

Wenn Sie eine statische IP-Adresse besitzen, editieren Sie einfach die Datei `/etc/rc.d/rc.local` und hängen die folgende Zeile ans Ende an:

```
sh /etc/rc.d/rc.firewall
```

Falls Sie in den Firewall-Regeln symbolische Hostnamen verwendet haben, denken Sie daran, dass im Skript der Zugang zu DNS erlaubt werden muss, bevor die Ausführung des Skripts bei diesen Hostnamen ankommt. Wenn Sie einen lokalen Nameserver einsetzen, wird `named` automatisch vor dem Aufruf von `rc.local` gestartet, sowohl beim Booten als auch später beim Wechseln in einen anderen Runlevel.

3.11.2 Installation einer Firewall für PPP

Bei den weitaus meisten der privat genutzten Computer hierzulande dürfte die Internet-Anbindung über ein Modem oder ISDN erfolgen. Dank PPP ist der Verbindungsaufbau z.B. zu einem der zahlreichen Internet-by-Call-Anbieter oder zu einem »echten« Internet-Provider in den meisten Fällen sehr einfach und problemlos.

Bei der Einrichtung einer Firewall verkompliziert die normalerweise genutzte PPP-Konfiguration Ihre Situation allerdings erheblich. Die IP-Adresse wird bei jedem Einwahlvorgang vom Provider dynamisch vergeben; beim Bootvorgang ist sie überhaupt noch nicht bekannt. Die Firewall wird deshalb nicht beim Systemstart initialisiert und konfiguriert, sondern erst nach erfolgter Anwahl.

Warnung

Policies

Unmittelbar nach dem Booten sind die Policies für alle Chains auf `ACCEPT` gesetzt. Wenn Sie die Firewall erst nach dem Verbindungsaufbau einrichten, gibt es deshalb eine kurze Phase, in der Ihr Computer bereits aus dem Internet zugänglich ist, Ihre Firewall aber noch nicht greift. Um dem vorzubeugen, nehmen Sie folgende Zeilen in die Datei `/etc/rc.d/rc.local` auf (bzw. in die Entsprechung Ihrer Distribution):

```
ipchains -P input DENY
ipchains -P output DENY
ipchains -P forward DENY
ipchains -A input -i lo -j ACCEPT
ipchains -A output -i lo -j ACCEPT
```

Nachdem der PPP-Dämon `pppd` erfolgreich eine Verbindung hergestellt hat, ruft er das Skript `/etc/ppp/ip-up` auf. Dieses startet – gemäß einer häufig umgesetzten Konvention – seinerseits das Skript `/etc/ppp/ip-up.local`, in dem Sie persönliche Einstellungen in Abweichung von den Vorgaben Ihrer Distribution vornehmen. Da dieses Skript als Parameter u.a. die dynamisch vergebene IP-Adresse erhält, ist es eine ideale Stelle für den Aufruf der Firewall.

Parameter	Inhalt
\$1	Der Verbindung zugeordnetes Netzwerkinterface, z.B. ppp0
\$2	Entsprechendes physikalisches Gerät, z.B. /dev/ttyS1
\$3	Übertragungsgeschwindigkeit
\$4	Dem eigenen Computer zugewiesene IP-Adresse
\$5	IP-Adresse des fremden Computers
\$6	Evtl. zusätzliche Parameter, die beim Aufruf von pppd angegeben wurden

Tab. 3.21: Parameter für `ip-up.local`

Legen Sie die Einstellungen für die Firewall, wie oben vorgeschlagen, in einem separaten Skript ab, z.B. in `/etc/rc.d/rc.firewall`. Die eigene IP-Adresse definieren Sie hierin natürlich nicht explizit. Stattdessen übernimmt `rc.firewall` sie aus einem Parameter:

```
IPADDR=$1
```

Nach dem Verbindungsaufbau rufen Sie dieses Skript auf und geben Ihre IP-Adresse als Kommandozeilenoption an. Damit dies vollautomatisch geschieht, hängen Sie die folgenden Zeilen an `/etc/ppp/ip-up.local` an:

```
# Firewall für die neu erhaltene IP-Adresse umkonfigurieren
/etc/rc.d/rc.firewall $4
```

Zusätzlich erschwert wird die Einrichtung einer Firewall dadurch, dass bei vielen aktuellen Internet-by-Call-Providern die IP-Adressen der Nameserver nicht explizit bekannt gegeben, sondern von `pppd` ausgehandelt werden. Unter Linux dient dazu die Option `usepeerdns`. Dem Skript `ip-up.local` sowie allen von ihm aufgerufen Skripten stehen die so erhaltenen Adressen in den beiden Variablen `DNS1` und `DNS2` zur Verfügung. Benutzen Sie diese, sofern Sie den DNS-Zugang für andere Server als die des eigenen Providers beschränken wollen.

3.11.3 Installation einer Firewall für DHCP

Allmählich hält DSL auch in Deutschland Einzug in Privathaushalte. Die schnelle Technologie wird immer erschwinglicher, stellt aber besondere Anforderungen an die Sicherheit des angeschlossenen Systems. Immerhin ist sie häufig mit einer Flatrate kombiniert, der Computer bleibt also über längere Zeit ununterbrochen am Netz.

Wie bei einer Modemverbindung verfügen Sie auch mit DSL meistens nicht über eine eigene IP-Adresse, sondern erhalten von Ihrem Provider dynamisch eine IP zugewiesen. Diese kann sich sogar bei bestehender Verbindung ändern!

So wie beim Modem PPP für die Aushandlung der Verbindungsdaten eingesetzt wird, kommt bei DSL in der Regel DHCP zum Zuge. Der zugehörige Dämon heißt bei Red Hat 6.2 `pump`.¹

Für DHCP gilt das bereits oben zu PPP Gesagte: Während des Bootvorgangs sollten Sie die Policies aller Firewall-Chains so setzen, dass vor Aktivierung der Firewall alle Pakete abgewiesen werden. Verwenden Sie dazu beispielsweise die folgenden Zeilen in `/etc/rc.d/rc.local`:

```
ipchains -P input DENY
ipchains -P output DENY
ipchains -P forward DENY
ipchains -A input -i lo -j ACCEPT
ipchains -A output -i lo -j ACCEPT
```

Sobald sich die IP-Adresse Ihres Computers ändert, schreibt `pump` die Konfigurationsdatei für Nameserver-Clients, `/etc/resolv.conf`, neu. Gleichzeitig kann es ein externes Skript aufrufen. Dazu ergänzen Sie in der Konfigurationsdatei `/etc/pump.conf` folgende Zeile:

```
script /etc/pump.lease
```

Das Skript `/etc/pump.lease` wird jedesmal aufgerufen, wenn sich eine Änderung der über DHCP ausgehandelten IP-Adresse ergibt. Die übergebenen Argumente entnehmen Sie bitte Tabelle 3.22.

Parameter	Inhalt
\$1	Grund des Aufrufs: <code>up</code> beim Aufbau einer neuen Verbindung, <code>renewal</code> bei einer Änderung der IP-Adresse bei bestehender Verbindung, oder <code>down</code> bei der Unterbrechung der Verbindung.
\$2	Zugeordnetes Netzwerkinterface, z.B. <code>eth0</code>
\$3	Neue IP-Adresse (außer beim Verbindungsabbau)

Tab. 3.22: Parameter für `/etc/pump.lease`

Rufen Sie Ihr Firewallskript – etwa `/etc/rc.d/rc.firewall` – aus `/etc/pump.lease` heraus auf. `pump.lease` könnte so aussehen:

```
#!/bin/bash

# Dieses Skript wird von pump gestartet, sobald sich unsere IP-Adresse
# ändert. Es konfiguriert das System entsprechend um.
```

1. Die Aufgaben von `pppd` und `pump` unterscheiden sich natürlich ganz erheblich; für unsere Zwecke besteht aber durchaus eine gewisse Analogie.

```
# resolv.conf auslesen
n=1
fgrep nameserver /etc/resolv.conf | sed -e 's/nameserver //' |
    while read NSERV; do
        case $n in
            1) NAMESERVER_1=$NSERV ;;
            2) NAMESERVER_2=$NSERV ;;
            3) NAMESERVER_3=$NSERV ;;
        esac
        n=$((n+1))
    done
export NAMESERVER_1 NAMESERVER_2 NAMESERVER_3

EXTERNAL_INTERFACE=$2
IPADDR=$3
export EXTERNAL_INTERFACE IPADDR

/etc/rc.d/rc.firewall
```

Da die Variablen `EXTERNAL_INTERFACE` und `IPADDR` sowie alle mit `NAMESERVER_` beginnenden Variablen hier bereits gesetzt werden, löschen Sie sie aus `rc.firewall`.

Zusätzlich aktualisieren Sie gegebenenfalls die Konfiguration Ihres eigenen Nameservers, sofern Sie einen betreiben.

3.12 Zusammenfassung

Dieses Kapitel hat gezeigt, wie man eine eigenständige Firewall mit `ipchains` entwickelt. Als Policy wurde `DENY` gesetzt. Potenzielle Ärgernisse wie gefälschte Adressen oder Dienste auf unprivilegierten Ports wurden angesprochen. ICMP-Nachrichten, die Kontroll- und Statusmeldungen der IP-Netzwerkschicht, wurden behandelt, danach der DNS Name-Service, der von allen Netzwerkdiensten gebraucht wird, sowie `auth`, ein Dienst zur Benutzeridentifikation. Regelbeispiele zeigten, wie man beliebige Netzwerkdienste über eine Firewall erlaubt und die Menge an protokollierter Information bestimmt. Schließlich wurde die Installation der Firewall demonstriert, und zwar für Sites sowohl mit statischer als auch mit dynamischer IP-Adresse.

Am Ende des Kapitels wurde das Firewall-Skript geringfügig erweitert, sodass man eine Bastion-Firewall für ein kleines LAN erhielt. Vollständige Beispielskripten für `ipchains` wie auch für `ipfwadm` sind in Anhang B abgedruckt.

Kapitel 4 baut auf der Grundlage der Bastion-Firewall eine kompliziertere Firewall-Architektur auf. Dabei wird ein abgeschirmtes Subnetz mit zwei Firewalls vorgeführt, die einen DMZ-Perimeter abgrenzen. Diese etwas ausgetüfteltere Konfiguration kann für eine kleine Firma durchaus angemessen sein.

Kapitel 4

LANs, mehrfache Firewalls und Perimeter- Netze



- 4.1 Sicherheit im LAN
- 4.2 Konfigurationsmöglichkeiten für ein privates LAN mit vertrauenswürdigen Benutzern
- 4.3 Konfigurationsmöglichkeiten für ein größeres oder weniger vertrauenswürdiges LAN
- 4.4 Eine formale Firewall mit abgeschirmtem Subnetz
- 4.5 Zusammenfassung

Eine Paketfilter-Firewall ist ein statischer Router mit Kontrollregeln für den Netzwerkverkehr, die lokale Vorschriften hinsichtlich der erlaubten Pakete durchsetzen. Die Firewall aus Kapitel 3 ist eine grundlegende Bastion-Firewall. Als Router und Paketfilter auf einem Computer mit zwei oder mehr Netzwerkinterfaces (einem, das ans Internet angeschlossen ist, und einem zweiten ins lokale Netz; man spricht von einer *dual-homed*-Maschine) entscheidet die Firewall anhand der ihr vorgegebenen Regeln, ob sie Pakete von einem Interface zum anderen weiterleiten oder sie blockieren soll. Auf einem dual-homed-System, das gleichzeitig Masquerading für den Netzwerkverkehr vom LAN betreibt, fungiert die Maschine als Firewall für ein einfaches *abgeschirmtes Subnetz*: die Kombination aus IP-Masquerading und statischem Routing sorgt für eine Art Proxy-Gateway.

Bei diesem Setup steht also eine Maschine mit zwei Netzwerkkarten zwischen dem LAN und dem Internet. In so einer Situation stellen die Firewall-Regeln für jedes der beiden Netzwerkinterfaces ein I/O-Paar dar. In unserem Fall gibt es zwei solche Paare: Die Firewall filtert zum einen alles, was über das externe Interface ankommt und gesendet wird. Zum anderen filtert sie – in unterschiedlichem Ausmaß – auch alles, was über das interne Interface empfangen und gesendet wird. Die zwei Interfaces werden separat behandelt. Darüber hinaus gelangen Pakete auch nicht direkt vom Internet ins LAN. Die Filterregeln der beiden Interfaces machen den Computer zu einer Bastion-Firewall und einem statischen Router zwischen den beiden angeschlossenen Netzen.

Die Firewall-Konfiguration aus Kapitel 3 ist für ein privates System völlig adäquat. Sie ist so ziemlich das Beste, was man mit einem einzelnen Computer machen kann. Bei einer Firewall, die ein LAN ans Internet anbindet, ist das schon etwas anderes. Die Frage stellt sich aber, ob in einer vertrauenswürdigen Umgebung die verbesserte Sicherheit wirklich den zusätzlichen Aufwand lohnt.

Wenn in den Computer, auf dem eine Bastion-Firewall läuft, jemals eingebrochen wird, ist alles vorbei. Selbst wenn für das interne Netzwerkinterface separate Firewall-Regeln gelten, wird sich der Eindringling recht bald Zugang zum `root`-Account verschaffen, und spätestens dann sind für ihn auch die internen Systeme frei zugänglich. Freilich, wenn Sie die öffentlich angebotenen Dienste sorgfältig auswählen und die Firewall restriktiv einrichten, haben Sie die besten Chancen, dass Sie nie in solch eine Situation geraten. Trotzdem ist eine Bastion-Firewall eben genau das: eine einzige und letzte Bastion. Wenn sie fällt, ist das System offen. Alles oder nichts.

Größere Organisationen und ihre Netze werden sich nicht auf einen einzelnen Rechner verlassen. Man wird einen abgeschirmten Host ohne direktes Routing oder ein abgeschirmtes Subnetz mit Proxies verwenden, dazu eine DMZ als Perimeter-Netzwerk, das sich zwischen der externen (Bastion) und der internen, sekundären Firewall (Choke¹) befindet. Darüber hinaus verfügen öffentlich

1. »Choke« bedeutet wörtlich »Drossel«. In der Übersetzung werden wir den Originalbegriff verwenden.

zugängliche Server in der DMZ über ihre eigenen, spezialisierten Firewalls. D.h.: Diese Sites können viel mehr Rechner einsetzen und haben auch die Fachleute für die Administration.

Hinweis***DMZ oder Perimeter-Netzwerk***

Ein Perimeter-Netzwerk zwischen zwei Firewalls heißt DMZ oder »Demilitarisierte Zone«. Eine DMZ stellt einen geschützten Bereich dar, in dem man öffentliche Server (oder Dienste) laufen lassen kann. Sie ist vom Rest des privaten Netzes isoliert.

Dieses Kapitel behandelt einige der Grundkonzepte der Sicherheit im LAN. Sicherheitsvorschriften definiert man in Abhängigkeit von dem jeweiligen Sicherheitsbedarf, der Bedeutung der Daten, die geschützt werden sollen, und den Kosten, die ein Datenverlust oder Bruch der Privatsphäre verursachen würde. Am Beginn dieses Kapitels stellen wir einige der Fragen, die beantwortet sein sollten, bevor man solche Vorschriften festlegt und über den Einsatz seiner Server entscheidet.

Zu Beginn besprechen wir die LAN-Erweiterungen, die noch in Kapitel 3 für ein privates LAN vorgestellt wurden. Für einen Privathaushalt ist die dort gezeigte Architektur perfekt, aber bereits einige kleine Firmen werden höhere Ansprüche stellen, besonders wenn manche Geräte auf dem LAN öffentliche Internet-Services anbieten, während andere für interne Entwicklungs- und Verwaltungsaufgaben genutzt werden. Daher erweitern wir das Firewall-Beispiel aus Kapitel 3, sodass raffiniertere LAN-Optionen unterstützt werden, als sie im Heimbereich notwendig sind.

Anschließend gehen wir von dem Beispiel aus Kapitel 3 aus und entwickeln eine ganz formale, ausgefeilte, lehrbuchmäßige Firewall. Die Bastion hat zwei Netzwerkkomponenten: eines zum Internet, eines zum Perimeter-Netz, zur DMZ. Computer in der DMZ bieten öffentlich zugängliche Internet-Dienste an. Die DMZ enthält auch eine zweite Firewall, die *Choke*, die weitere lokale Netze von den quasi-öffentlichen Servern in der DMZ trennt. Hinter der Choke sind die privaten Rechner geschützt. Wenn die Bastion geknackt wird, sind die öffentlichen Maschinen in der DMZ dem Angreifer völlig ausgesetzt. Die Choke-Firewall schützt aber selbst in dieser Situation noch das interne LAN vor der Bastion-Firewall und dem Rest des Perimeter-Netzes.

4.1 Sicherheit im LAN

Sicherheit hängt im Wesentlichen davon ab, wie groß Ihr LAN ist, wie es aufgebaut ist und wofür es benutzt wird. Werden Dienste ans Internet exportiert? Laufen die zugehörigen Server auf der Firewall oder auf internen Maschinen? Z.B.

könnten Sie auf der Bastion-Firewall einen anonymen FTP-Zugang betreiben, eine Web-Site hingegen von einem internen Computer, der in der DMZ steht. Wenn Sie auf internen Maschinen Dienste betreiben, sollten Sie diese gegebenenfalls in ein Perimeter-Netzwerk stellen und durch eigenständige Paketfilter und Zugriffsbeschränkungen absichern. Wenn Sie von internen Maschinen etwas anbieten, soll diese Tatsache auch nach außen hin erkennbar sein, oder werden die Services über einen Proxy oder einen transparenten Mechanismus so umgeleitet, dass sie scheinbar von der Firewall kommen?

Welche Informationen wollen Sie überhaupt über die Computer Ihres LANs preisgeben? Wollen Sie einen eigenen DNS betreiben? Sind die lokalen DNS-Datenbanken für die Bastion-Firewall sichtbar?

Können Ihre Mitarbeiter sich aus dem Internet auf den internen Maschinen einloggen? Wie viele interne Maschinen sind in diesem Fall zugänglich, und welche? Haben alle Benutzer die gleichen Zugriffsrechte? Sind externe Dienste von allen internen Maschinen erreichbar? Z.B. müssten sich bei einem »abgeschirmten Host« Ihre Benutzer zuerst auf dem Bastion-Computer einloggen, bevor sie Zugang zum Internet haben. Bei dieser Architektur routet man überhaupt nichts.

Laufen hinter der Firewall private LAN-Dienste? Verwenden Sie beispielsweise NFS, NIS oder NTP, oder die R-Kommandos von Berkeley (`rsh`, `rlogin` und `rcp`)? Müssen Sie solche Dienste daran hindern, private Daten ins Internet zu übertragen, wie es z.B. bei SNMP, DHCP, `timed`, `xntpd`, `ruptime` und `rwho` der Fall ist? Indem Sie solche Dienste nur hinter der Choke-Firewall betreiben, isolieren Sie sie vollständig vom Internet.

Hierher gehört auch die Frage, wie sich interner und externer Zugang zu Internet-Diensten zueinander verhalten. Bieten Sie z.B. FTP intern an, aber nicht extern? Oder haben Sie vielleicht unterschiedliche FTP-Dienste für interne und externe Benutzer? Planen Sie einen privaten Web-Server, oder sollen manche Teile ein und desselben Webservers nur für lokale Nutzer sichtbar sein? Wollen Sie einen lokalen Mailserver für abgehende Mail einsetzen, ankommende Mail aber über einen anderen Mechanismus ausliefern? D.h. soll ankommende Mail direkt an einen Ihrer Computer zugestellt werden, oder holen Sie sie aktiv von einem Gerät Ihres Internet-Providers ab?

4.2 Konfigurationsmöglichkeiten für ein privates LAN mit vertrauenswürdigen Benutzern

In dieser Situation unterscheiden wir zwischen zwei Arten von internem Verkehr. Wie in Bild 4.1 gezeigt, gibt es zum einen lokale Zugriffe auf die Bastion-Firewall durch deren internes Interface. Daneben erfolgen lokale Zugriffe auf das Internet durch das externe Interface der Firewall.

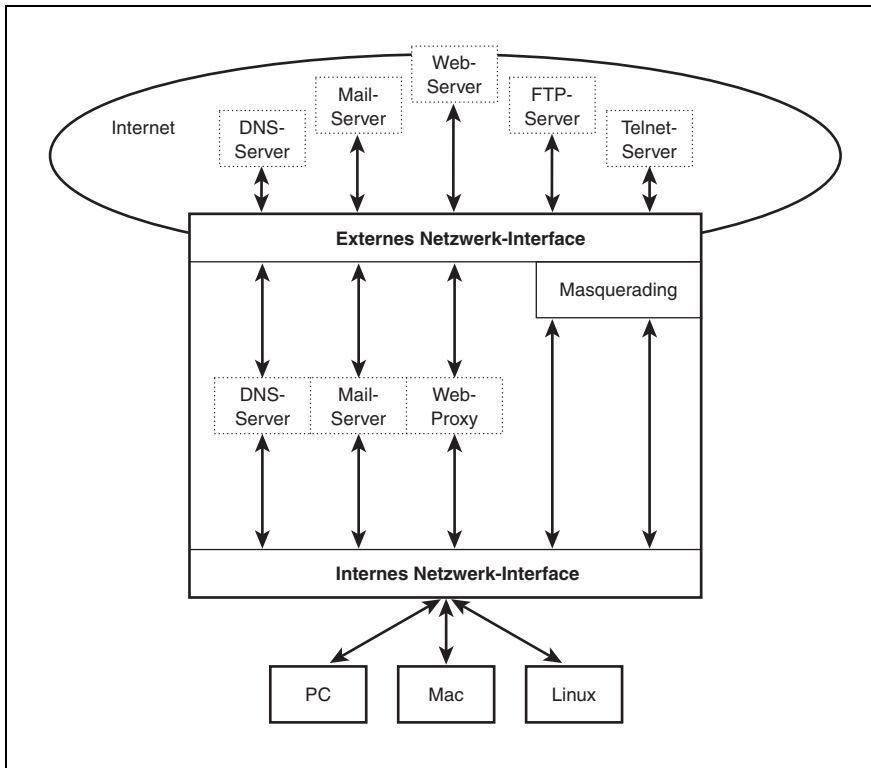


Bild 4.1: LAN-Verkehr zur Firewall und ins Internet

Vermutlich besteht für die meisten kleinen Systeme kein Grund, Pakete zwischen Firewall und LAN zu filtern. Eine Ausnahme entsteht dadurch, dass die meisten privaten Sites nur eine einzige öffentliche IP-Adresse besitzen: IP-Masquerading. Sie werden intern vermutlich also nur filtern, damit die Absender-Adresse von internen Paketen, die ins Internet weitergereicht werden, korrekt umgeschrieben wird.

Wenn Sie eine öffentliche IP für die Firewall haben und alle internen Maschinen private IPs benutzen, dann müssen Sie IP-Masquerading betreiben. Damit setzen Sie eine Art Proxy auf. Selbst wenn Ihre internen Maschinen über ganz offizielle IP-Adressen verfügen, sollten Sie trotzdem keine Pakete direkt zwischen LAN und Internet weiterleiten. Benutzen Sie entweder Proxies auf Anwendungsebene, oder ebenfalls IP-Masquerading.

Masquerading könnte man als Proxy auf sehr niedriger Ebene bezeichnen. Ein Proxy stellt für den Client eine Verbindung zu einem fremden Server her. Alle abgehenden Verbindungen haben ihren Ausgangspunkt in dem Rechner, auf dem der Proxy läuft. Diesen Effekt hat auch Masquerading: Alle abgehenden Pakete

von Ihrem LAN haben als Absender-Adresse die Bastion-Firewall, weil die Absender-IP im Paket-Header durch die externe IP der Bastion ersetzt wurde. Bei ankommenden Paketen wird die Empfänger-Adresse in die private IP des jeweiligen Computers zurückübersetzt.

4.2.1 LAN-Zugriffe auf die Bastion-Firewall

Daheim werden Sie vermutlich keine Beschränkungen für den Zugriff von LAN-Computern auf die Firewall einrichten.

Wenn wir von der Firewall aus Kapitel 3 ausgehen, benötigen wir zwei zusätzliche Konstanten, die das interne, LAN-seitige Interface der Firewall beschreiben. Heißt beispielsweise die Netzwerkkarte zum LAN `eth1` und besteht das LAN aus den privaten Klasse-C-Adressen von `192.168.1.0` bis `192.168.1.255`, dann sehen diese Konstanten wie folgt aus:

```
LAN_INTERFACE_1="eth1"  
LAN_1="192.168.1.0/24"
```

Unbeschränkten Zugriff für alles hinter dem internen Interface ermöglichen wir, indem wir einfach alle Protokolle und alle Ports freigeben:

```
ipchains -A input -i $LAN_INTERFACE_1 -s $LAN_1 -j ACCEPT  
ipchains -A output -i $LAN_INTERFACE_1 -d $LAN_1 -j ACCEPT
```

4.2.2 LAN-Zugriffe auf andere LANs: Lokale Pakete zwischen mehreren lokalen Netzen weiterleiten

Wenn Routing zwischen den Maschinen auf Ihrem lokalen Netz (oder auf Ihren lokalen Netzen) erforderlich ist, müssen Sie den internen Zugriff für die benötigten Ports öffnen – außer, es gibt weitere interne Verbindungen über einen Hub. Ansonsten wird alles lokale Routing durch die Firewall durchgeführt.

Hierfür werden zwei weitere Konstanten definiert. Im Beispiel definieren wir eine zweite interne Netzwerkkarte als `eth0`, das entsprechende LAN als die privaten IP-Adressen von `192.168.3.0` bis `192.168.3.255`:

```
LAN_INTERFACE_2="eth2"  
LAN_2="192.168.3.0/24"
```

Die folgenden Regeln erlauben jeweils den Zugriff auf die Firewall:

```
ipchains -A input -i $LAN_INTERFACE_1 -s $LAN_1 -j ACCEPT  
ipchains -A output -i $LAN_INTERFACE_1 -d $LAN_1 -j ACCEPT  
  
ipchains -A input -i $LAN_INTERFACE_2 -s $LAN_2 -j ACCEPT  
ipchains -A output -i $LAN_INTERFACE_2 -d $LAN_2 -j ACCEPT
```

Nun leiten wir Pakete zwischen den beiden Netzen in beide Richtungen weiter. Es wird kein Masquerading eingesetzt.

```
ipchains -A forward -i $LAN_INTERFACE_2 \  
-s $LAN_1 -d $LAN_2 -j ACCEPT
```

```
ipchains -A forward -i $LAN_INTERFACE_1 \  
-s $LAN_2 -d $LAN_1 -j ACCEPT
```

4.2.3 LAN-Zugriffe aufs Internet: Forwarding und Masquerading

Forwarding bedeutet, dass Pakete unverändert von einem Netzwerkimterface zu einem anderen weitergegeben werden; es handelt sich um einfaches Routing. Forwarding kann in beide Richtungen geschehen. Wenn Ihre internen Computer über registrierte IP-Adressen verfügen, könnten Sie theoretisch allen Traffic forwarden, und die Geräte würden im Internet als eigenständige Computer auftreten. Ebenso können Sie ankommende Pakete forwarden, z.B. alle ankommende Mail an einen Mailserver der DMZ weiterleiten.

Masquerading ist ein separater Kernel-Dienst, dem Forwarding übergeordnet. Pakete in beide Richtungen sind betroffen, aber Masquerading arbeitet trotzdem nicht symmetrisch: Nur abgehende Verbindungen sind erlaubt. Alle Pakete von LAN-Maschinen zu Empfängern im Internet werden weitergeleitet, wobei die IP-Adresse des LAN-Computers durch die externe IP der Firewall ersetzt wird. Wenn eine Antwort ankommt, geschieht das Umgekehrte: Bevor das Paket an die interne Maschine weitergeleitet wird, ersetzt der Masquerading-Code die Empfänger-IP im Paket (die der externen IP der Firewall entspricht) durch die IP-Adresse des wirklichen Empfängers.

Eine ankommende Verbindung kann nicht an eine maskierte interne Adresse weitergereicht werden, weil die interne IP-Adresse für das Internet unsichtbar ist. Jede ankommende Verbindung muss an eine eindeutige, offiziell registrierte, öffentliche IP-Adresse gerichtet sein.

Sowohl Forwarding als auch Masquerading sind Dienste des Kernels; sie müssen bei der Kernelkonfiguration aktiviert und in den Kernel hineinkompiliert werden. Wenn Sie Masquerading benutzen wollen, müssen Sie auch Forwarding einschalten. Die Konfiguration der beiden erfolgt aber getrennt.

Forwarding konfigurieren Sie in den Netzwerkeinstellungen. Bei Red Hat 6.0 geschieht das in der Datei `/etc/sysconfig/network` – suchen Sie nach einer Zeile `FORWARD_IPV4=yes`. Bei Red Hat 6.2 ist die zugehörige Konfigurationsdatei `/etc/sysctl.conf`. Forwarding lässt sich auch über das `control-panel` aktivieren. Wenn IP-Forwarding beim Booten nicht aktiviert war, können Sie es nachträglich mit folgendem Befehl einschalten:

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

Masquerading aktivieren Sie über die entsprechenden Optionen von `ipchains`.

Beim Design einer Linux-Firewall kann man Masquerading als Sonderfall des Forwarding ansehen, und so wird es von `ipchains` auch behandelt.

Die folgende Regel sorgt dafür, dass alle Pakete von Maschinen auf dem LAN das Masquerading durchlaufen:

```
ipchains -A forward -i $EXTERNAL_INTERFACE \  
-s $LAN_1 -j MASQ
```

»Alle Pakete« ist dabei ein bisschen relativ: Weil ja immer die erste passende Regel zutrifft, gilt diese hier natürlich nur für Pakete, die noch nicht anderweitig behandelt worden sind.

Ob Sie nun offizielle oder private IP-Adressen verwenden, Sie sollten auf alle Fälle nicht einfaches Forwarding, sondern Masquerading benutzen. Damit gewinnen Sie ein enormes Maß an Sicherheit. Ankommende Verbindungen von externen Maschinen zu Ihren geschützten lokalen Geräten sind nicht möglich; die lokalen Geräte sind nach außen völlig unsichtbar. Es sieht so aus, als würde alles von dem Firewall-Computer ausgehen.

Services mit atypischen Kommunikationsprotokollen benötigen entweder ein zusätzliches und spezifisches Masquerading-Modul im Kernel oder ein eigenständiges Proxy-Programm. Manche Dienste verwenden mehrere Verbindungen, z.B. baut bei FTP der Server in Reaktion auf die Anfrage des Clients einen separaten Datenkanal auf. Einige Dienste zwingen sowohl die Clients als auch den Server zur Verwendung unprivilegierter Ports. Das ist bei UDP ein besonderes Problem, wo man keinen Verbindungszustand heranziehen kann, um zu prüfen, ob das ankommende Paket akzeptabel ist oder nicht.

Solche atypischen Protokolle mit mehrfachen Verbindungen oder über UDP sind Beispiele dafür, dass sich manche Dienste über einen Paketfilter nicht gut angehen lassen. Sie sind Kandidaten für Spezialmodule oder Proxies auf Anwendungsebene.

Masquerading-Module sind in den üblichen Linux-Distributionen in wachsender Anzahl verfügbar. Red Hat 6.0 enthielt solche Module für FTP, Quake, CU-SeeMe, IRC, RealAudio und LiveVideo. Wenn in der Kernelkonfiguration Forwarding und Masquerading eingeschaltet sind, werden diese Module automatisch übersetzt. Sie müssen sie allerdings noch explizit laden.

Eine Alternative oder eine zusätzliche Sicherheitsmaßnahme ist die Verwendung eines Proxies auf Anwendungsebene, z.B. SOCKS. Auch hier sieht es nach außen hin so aus, als wäre der Firewall-Computer Absender aller Pakete. Proxies ermöglichen häufig auch eine feiner steuerbare Zugriffskontrolle, als sie über Paketfilter möglich ist. Sie sind meistens sehr spezifisch für eine bestimmte Applikation, deren Kommunikationsprotokoll sie dann genau verstehen. Daher können sie Absender und Empfänger von Paketen auf eine Art und Weise prüfen, wie es einem Paketfilter nicht möglich ist.

4.3 Konfigurationsmöglichkeiten für ein größeres oder weniger vertrauenswürdiges LAN

Eine größere Organisation oder Firma wird ausgefeiltere und spezifischere Mechanismen einsetzen als die einfachen Forwarding- und Masquerading-Regeln aus dem vorherigen Abschnitt, die ja für ein vollkommen vertrauenswürdiges LAN in einem Haushalt gedacht waren. In einer weniger vertrauenswürdigen Umgebung schützt man die Firewalls vor internen Benutzern genauso wie vor externen. Portspezifische Firewall-Regeln existieren für das interne Netzwerkinterface ebenso wie für das externe. Diese Regeln sind eventuell identisch mit den Regeln für externe User, oder sie sind ein wenig großzügiger. Was das interne Interface passieren darf, hängt davon ab, welche Systeme im LAN betrieben werden und welche lokalen Dienste auf der Firewall laufen.

Beispielsweise werden Sie möglicherweise alle lokalen Broadcast-Pakete sperren. Wenn Sie nicht allen Ihren internen Benutzern vollständig vertrauen, schränken Sie den Zugriff auf die Firewall von innen vermutlich genauso stark ein wie von außen. Darüber hinaus sollten Sie auf dem Firewall-System nur so viele Benutzeraccounts wie unbedingt nötig einrichten.

Was ich vorhin über Masquerading und Proxies gesagt habe, gilt natürlich auch für ein größeres LAN. Eine kleine Firma hat möglicherweise nur eine einzelne IP-Adresse und ist damit zwingend auf Masquerading angewiesen. Größere Firmen mieten häufig mehrere IP-Adressen oder gleich einen ganzen Netzwerkblock. Diese öffentlichen Adressen verteilt man dann auf die öffentlich zugänglichen Server. In dieser Situation werden abgehende Verbindungen über Forwarding weitergeleitet und ankommende Pakete ganz normal geroutet. Die internen, öffentlich zugänglichen Server stehen nicht in einem Netz aus privaten IP-Adressen, sondern in einem lokalen Subnetz, das eine öffentlich zugängliche DMZ repräsentiert.

4.3.1 Bildung mehrerer Subnetze

IP-Adressen bestehen aus einer Netzwerkadresse und einer Hostadresse innerhalb dieses Netzwerks. Netzwerkadressen der Klassen A, B und C werden jeweils durch die ersten 8, 16 bzw. 24 Bits definiert. Innerhalb jeder Adressklasse bilden die restlichen Bits den Host-spezifischen Anteil der IP-Adresse.

Bei der Bildung von Subnetzen erweitert man den Netzwerk-spezifischen Anteil der IP-Adresse. Dabei definiert man eine lokale Netzwerkmaske, in der die höher-signifikanten Bits der Hostadresse mit zur Netzwerkadresse gerechnet werden. Diese zusätzlichen Netzwerkbits bezeichnen dann mehrere lokale Netze. Fremde Sites sehen diese lokalen Subnetze nicht, sondern behandeln die IPs als ganz normale Adressen der Klassen A, B und C.

Beispiel: Sehen wir uns den privaten Klasse-C-Adressblock 192.168.1.0 an. 192.168.1.0 ist eine Klasse-C-Netzwerkadresse, wobei das Netzwerk bis zu 254

Hosts enthalten kann. Die Netmask für dieses Netz ist 255.255.255.0 und entspricht damit den ersten 24 Bits (eben der Netzwerkadresse) des Netzwerks 192.168.1.0.

Definiert man nicht nur die ersten 24 Bits, sondern die ersten 25 Bits als Netzwerkadresse, erhält man zwei lokale Subnetze. Das signifikanteste Bit des Host-Anteils der Adresse wird jetzt zur Netzwerkadresse dazugerechnet. Der Host-Anteil enthält nicht mehr acht Bit, sondern nur noch sieben. Die Netmask ist nun 255.255.255.128. Zwei Subnetze sind entstanden: 192.168.1.0 mit Hosts von 1 bis 126, und 192.168.1.128 mit Hosts von 129 bis 254. Jedes Subnetz verliert zwei Hostadressen, weil die jeweils niedrigste Adresse (0 bzw. 128) als Netzwerkadresse und die höchste Adresse (127 bzw. 255) als Broadcast-Adresse verwendet wird. Tabelle 4.1 zeigt das noch einmal tabellarisch.

Sub-netz	Netzwerk- adresse	Netmask	erster adres- sierbarer Host	letzter adres- sierbarer Host	Broadcast- Adresse	Zahl der Hosts
0	192.168.1.0	255.255.255.0	192.168.1.1	192.168.1.254	192.168.1.255	254
1	192.168.1.0	255.255.255.128	192.168.1.1	192.168.1.126	192.168.1.127	126
2	192.168.1.128	255.255.255.128	192.168.1.129	192.168.1.254	192.168.1.255	126

Tab. 4.1: Aufteilung des Klasse-C-Netzes 192.168.1.0 in zwei Subnetze

Den Subnetzen 192.168.1.0 und 192.168.1.128 könnte man zwei separate Netzwerkkarten zuteilen. Aus der Sicht des Internets besteht die Site aus einem einzigen Netz mit maximal 254 Hosts. Intern besteht sie aus zwei unabhängigen Netzen mit jeweils maximal 126 Hosts.

Durch die Bildung von Subnetzen erhält man mehrere interne Netze, in denen man verschiedene Arten von Client- oder Server-Maschinen unterbringen und für die man das Routing völlig unabhängig festlegen kann. Beide Netze lassen sich durch unterschiedliche Firewalls schützen. Obwohl es denkbar ist, die LAN-Adresse durch Masquerading zu verbergen, geschieht das normalerweise nicht.

4.3.2 Selektiver interner Zugang nach Host, Adressbereich oder Port

Ebenso wie beim externen Netzwerkinterface können Sie auch beim internen Interface genau bestimmen, welche Pakete passieren dürfen. Beispielsweise könnten Sie – statt einfach alles durchzulassen – nur die Dienste DNS, SMTP, auth, POP und HTTP erlauben. Nehmen wir einmal an, diese Dienste würden von der Firewall für das LAN angeboten. Lokale Computer dürfen andere externe Dienste nicht nutzen.

Hinweis

In diesem Beispiel werden lokale Rechner auf die Dienste DNS, SMTP, auth, POP und HTTP eingeschränkt. Weil POP für das Abholen von Mail verwendet wird und DNS, SMTP und HTTP effektiv Proxy-Dienste sind, haben die lokalen Clients keinen direkten Internetzugang. Sie greifen immer auf lokale Server zu. POP läuft vollständig im LAN ab, bei den anderen Diensten greifen die lokalen Server für den lokalen Client auf fremde Server zu.

Im folgenden Beispiel sei der Firewall-Computer mit den folgenden Daten über eine Netzwerkkarte an das LAN angeschlossen:

```
LAN_INTERFACE="eth1"           # internes Interface zum LAN
FIREWALL="192.168.1.1"         # interne IP-Adresse der Firewall
LAN_ADDRESSES="192.168.1.0/24" # Adressbereich des LANs
```

Die Computer im lokalen Netz benutzen die Firewall als Nameserver:

```
ipchains -A input -i $LAN_INTERFACE -p udp \
-s $LAN_ADDRESSES $UNPRIVPORTS \
-d $FIREWALL 53 -j ACCEPT

ipchains -A output -i $LAN_INTERFACE -p udp \
-s $FIREWALL 53 \
-d $LAN_ADDRESSES $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $LAN_INTERFACE -p tcp \
-s $LAN_ADDRESSES $UNPRIVPORTS \
-d $FIREWALL 53 -j ACCEPT

ipchains -A output -i $LAN_INTERFACE ! -y -p tcp \
-s $FIREWALL 53 \
-d $LAN_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Ebenso verwenden LAN-Computer die Firewall als SMTP- und POP-Server:

Mailversand - SMTP

```
ipchains -A input -i $LAN_INTERFACE -p tcp \
-s $LAN_ADDRESSES $UNPRIVPORTS \
-d $FIREWALL 25 -j ACCEPT

ipchains -A output -i $LAN_INTERFACE ! -y -p tcp \
-s $FIREWALL 25 \
-d $LAN_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Mailempfang - POP

172 ————— Konfigurationsmöglichkeiten für ein weniger vertrauenswürdiges LAN

```
ipchains -A input -i $LAN_INTERFACE -p tcp \  
-s $LAN_ADDRESSES $UNPRIVPORTS \  
-d $FIREWALL 110 -j ACCEPT
```

```
ipchains -A output -i $LAN_INTERFACE ! -y -p tcp \  
-s $FIREWALL 110 \  
-d $LAN_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

sendmail führt auth-Anfragen durch, wenn ein Client eine Mail verschickt:

```
ipchains -A output -i $LAN_INTERFACE -p tcp \  
-s $FIREWALL $UNPRIVPORTS \  
-d $LAN_ADDRESSES 113 -j ACCEPT
```

```
ipchains -A input -i $LAN_INTERFACE ! -y -p tcp \  
-s $LAN_ADDRESSES 113 \  
-d $FIREWALL $UNPRIVPORTS -j ACCEPT
```

Schließlich läuft auf der Firewall, Port 8080, ein lokaler Web-Server mit Proxy- und Cache-Funktionalität. Lokale Geräte verwenden ihn als Proxy, und der Web-Server holt für sie die angeforderten Seiten aus dem Internet. Gleichzeitig können die Seiten lokal zwischengespeichert werden.

```
ipchains -A input -i $LAN_INTERFACE -p tcp \  
-s $LAN_ADDRESSES $UNPRIVPORTS \  
-d $FIREWALL 8080 -j ACCEPT
```

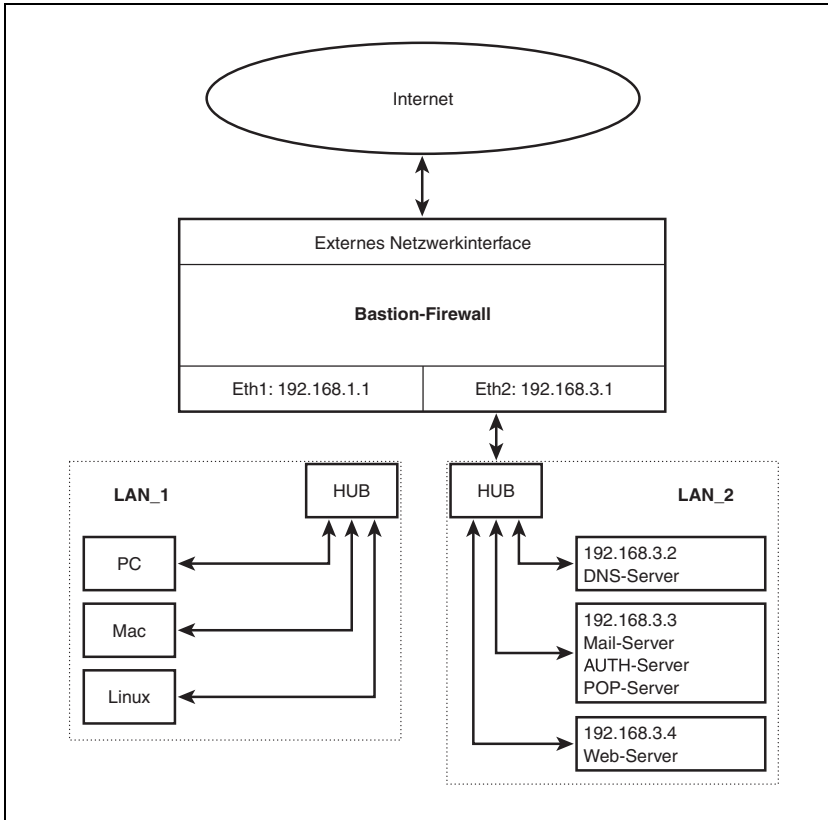
```
ipchains -A output -i $LAN_INTERFACE ! -y -p tcp \  
-s $FIREWALL 8080 \  
-d $LAN_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Konfigurationsmöglichkeiten für mehrere LANs

Indem wir ein zweites internes LAN hinzufügen, können wir dieses Beispiel noch etwas ausdehnen. Wie in Bild 4.2 gezeigt, bietet jetzt nicht mehr die Firewall die Dienste DNS, SMTP, auth, POP und HTTP an, sondern die Server in einem zweiten LAN. In diesem Fall werden Pakete durch die Firewall zwischen den beiden LANs geroutet, aber das Routing könnte auch direkt über einen Hub oder einen Switch laufen.

Die folgenden Variablen definieren für dieses Beispiel die LANs, die Netzwerkkarten und die Server:

```
CLIENT_LAN_INTERFACE="eth1"    # internes Interface zum LAN  
SERVER_LAN_INTERFACE="eth2"    # internes Interface zum LAN  
FIREWALL_1="192.168.1.1"       # interne IP-Adresse der Netzwerkkarte  
                                # der Firewall  
FIREWALL_2="192.168.3.1"       # interne IP-Adresse der Netzwerkkarte  
                                # der Firewall
```

**Bild 4.2:** Clients und Server werden in zwei LANs voneinander getrennt

```

CLIENT_LAN="192.168.1.0/24" # Adressbereich im LAN
SERVER_LAN="192.168.3.0/24" # Adressbereich im LAN
DNS_SERVER="192.168.3.2"   # DNS-Server für das LAN
MAIL_SERVER="192.168.3.3"  # Mailserver für das LAN
POP_SERVER="192.168.3.3"   # POP-Server für das LAN
WEB_SERVER="192.168.3.4"   # Web-Server für das LAN

```

Die Resolver-Einstellungen der lokalen Computer zeigen auf die IP-Adresse des DNS-Servers im Server-LAN. Für das Client-Interface der Firewall sind hier Server-Regeln definiert: Für die Clients ist die Firewall Server-seitig. Entsprechend gelten für das Server-Interface Client-Regeln, denn für den DNS-Server steht die Firewall in Richtung der Clients.

```

ipchains -A input -i $CLIENT_LAN_INTERFACE -p udp \
-s $CLIENT_LAN $UNPRIVPORTS \
-d $DNS_SERVER 53 -j ACCEPT

```

174 ————— Konfigurationsmöglichkeiten für ein weniger vertrauenswürdiges LAN

```
ipchains -A output -i $SERVER_LAN_INTERFACE -p udp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $DNS_SERVER 53 -j ACCEPT
```

```
ipchains -A input -i $SERVER_LAN_INTERFACE -p udp \  
-s $DNS_SERVER 53 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $CLIENT_LAN_INTERFACE -p udp \  
-s $DNS_SERVER 53 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

Zusätzlich benötigen wir Regeln, die das Forwarding von Paketen zwischen den beiden Interfaces erlauben. In diesem Fall enthalten die Forward-Regeln Interface, Client-Adressbereich, Client-Port, Server-Adresse und Server-Port.

```
ipchains -A forward -i $SERVER_LAN_INTERFACE \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $DNS_SERVER 53 -j ACCEPT
```

```
ipchains -A forward -i $CLIENT_LAN_INTERFACE \  
-s $DNS_SERVER 53 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

Und die Situation ist noch ein bisschen komplizierter: Der DNS-Server im Server-LAN muss seine Informationen ja auch irgendwo herbekommen und benötigt dafür Zugriff auf eine externe Quelle. Angenommen, der interne DNS-Server reicht alle unaufgelösten Anfragen an einen externen Server weiter und wird dabei durch Masquerading versteckt, dann sehen die UDP-Regeln der Firewall für internes Server-LAN-Interface und externes Internet-Interface folgendermaßen aus:

```
ipchains -A input -i $SERVER_LAN_INTERFACE -p udp \  
-s $DNS_SERVER 53 \  
-d externer DNS-Server 53 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 53 \  
-d externer DNS-Server 53 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s externer DNS-Server 53 \  
-d $IPADDR 53 -j ACCEPT
```

```
ipchains -A output -i $SERVER_LAN_INTERFACE -p udp \  
-s externer DNS-Server 53 \  
-d $DNS_SERVER 53 -j ACCEPT
```

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p udp \  
-s $DNS_SERVER 53 \  
-d externer DNS-Server 53 -j MASQ
```

Die Clients aus CLIENT_LAN versenden Mail über MAIL_SERVER als SMTP-Server:

```
# Mailversand - SMTP  
# -----
```

```
ipchains -A input -i $CLIENT_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $MAIL_SERVER 25 -j ACCEPT
```

```
ipchains -A output -i $SERVER_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $MAIL_SERVER 25 -j ACCEPT
```

```
ipchains -A input -i $SERVER_LAN_INTERFACE ! -y -p tcp \  
-s $MAIL_SERVER 25 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $CLIENT_LAN_INTERFACE ! -y -p tcp \  
-s $MAIL_SERVER 25 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A forward -i $SERVER_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $MAIL_SERVER 25 -j ACCEPT
```

```
ipchains -A forward -i $CLIENT_LAN_INTERFACE -p tcp \  
-s $MAIL_SERVER 25 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

**Der SMTP-Server im SERVER_LAN leitet die Mail an andere Computer weiter.
Dafür muss er über die Firewall auf das Internet zugreifen:**

```
ipchains -A input -i $SERVER_LAN_INTERFACE -p tcp \  
-s $MAIL_SERVER $UNPRIVPORTS \  
-d $ANYWHERE 25 -j ACCEPT
```

```
ipchains -A output -i $SERVER_LAN_INTERFACE ! -y -p tcp \  
-s $ANYWHERE 25 \  
-d $MAIL_SERVER $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p tcp \  
-s $MAIL_SERVER $UNPRIVPORTS \  
-d $ANYWHERE 25 -j MASQ
```

176 ————— Konfigurationsmöglichkeiten für ein weniger vertrauenswürdiges LAN

sendmail führt eine auth-Anfrage an den Mail-Client durch:

```
ipchains -A input -i $SERVER_LAN_INTERFACE -p tcp \
-s $MAIL_SERVER $UNPRIVPORTS \
-d $CLIENT_LAN 113 -j ACCEPT

ipchains -A output -i $CLIENT_LAN_INTERFACE -p tcp \
-s $MAIL_SERVER $UNPRIVPORTS \
-d $CLIENT_LAN 113 -j ACCEPT

ipchains -A input -i $CLIENT_LAN_INTERFACE ! -y -p tcp \
-s $CLIENT_LAN 113 \
-d $MAIL_SERVER $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $SERVER_LAN_INTERFACE ! -y -p tcp \
-s $CLIENT_LAN 113 \
-d $MAIL_SERVER $UNPRIVPORTS -j ACCEPT

ipchains -A forward -i $CLIENT_LAN_INTERFACE -p tcp \
-s $MAIL_SERVER $UNPRIVPORTS \
-d $CLIENT_LAN 113 -j ACCEPT

ipchains -A forward -i $SERVER_LAN_INTERFACE -p tcp \
-s $CLIENT_LAN 113 \
-d $MAIL_SERVER $UNPRIVPORTS -j ACCEPT
```

Realistischerweise müsste der SMTP-Server im SERVER_LAN auch Mail von Computern im Internet empfangen können. In diesem Beispiel ist das nicht direkt möglich, weil der Mail-Server durch Masquerading versteckt ist. Lösungen zu diesem Problem werden wir später besprechen. Auf der Grundlage des bisher Dargestellten könnten wir annehmen, es liefe auf der Bastion-Firewall ebenfalls ein Mailserver, der Mail annimmt und an den Computer MAIL_SERVER weitergibt.

Die Clients in CLIENT_LAN holen Mail vom POP_SERVER ab:

```
# Mailempfang - POP
# -----

ipchains -A input -i $CLIENT_LAN_INTERFACE -p tcp \
-s $CLIENT_LAN $UNPRIVPORTS \
-d $POP_SERVER 110 -j ACCEPT

ipchains -A output -i $SERVER_LAN_INTERFACE -p tcp \
-s $CLIENT_LAN $UNPRIVPORTS \
-d $POP_SERVER 110 -j ACCEPT

ipchains -A input -i $SERVER_LAN_INTERFACE ! -y -p tcp \
-s $POP_SERVER 110 \
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```



```
ipchains -A output -i $CLIENT_LAN_INTERFACE ! -y -p tcp \  
-s $POP_SERVER 110 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A forward -i $SERVER_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $POP_SERVER 110 -j ACCEPT
```

```
ipchains -A forward -i $CLIENT_LAN_INTERFACE -p tcp \  
-s $POP_SERVER 110 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

Schließlich läuft auf einem Rechner im Server-LAN ein Web-Proxy, der den Port 8080 benutzt. Interne Computer benutzen ihn als Caching-Proxy, und der Web-Server holt die angeforderten Seiten bei Bedarf aus dem Internet.

```
# WWW-Proxy  
# -----
```

```
ipchains -A input -i $CLIENT_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $WEB_SERVER 8080 -j ACCEPT
```

```
ipchains -A output -i $SERVER_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $WEB_SERVER 8080 -j ACCEPT
```

```
ipchains -A input -i $SERVER_LAN_INTERFACE ! -y -p tcp \  
-s $WEB_SERVER 8080 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $CLIENT_LAN_INTERFACE ! -y -p tcp \  
-s $WEB_SERVER 8080 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A forward -i $SERVER_LAN_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $WEB_SERVER 8080 -j ACCEPT
```

```
ipchains -A forward -i $CLIENT_LAN_INTERFACE -p tcp \  
-s $WEB_SERVER 8080 \  
-d $CLIENT_LAN $UNPRIVPORTS -j ACCEPT
```

Dieser Web-Server benötigt damit natürlich Internet-Zugang:

```
ipchains -A input -i $SERVER_LAN_INTERFACE -p tcp \  
-s $WEB_SERVER $UNPRIVPORTS \  
-d $ANYWHERE 80 -j ACCEPT
```

```
ipchains -A output -i $SERVER_LAN_INTERFACE ! -y -p tcp \  
-s $ANYWHERE 80 \  
-d $WEB_SERVER $UNPRIVPORTS -j ACCEPT  
  
ipchains -A forward -i $EXTERNAL_INTERFACE -p tcp \  
-s $WEB_SERVER $UNPRIVPORTS \  
-d $ANYWHERE 80 -j MASQ
```

4.3.3 Masquerading zwischen LAN und Internet

In den bisher gezeigten Beispielen haben Sie bereits einige Regeln für Forwarding und Masquerading gesehen. Solche Regeln können mit unterschiedlicher Genauigkeit angegeben werden.

... nach Interface

Zu Hause ist es am einfachsten, einfach allen Datenaustausch zwischen LAN und Internet zu maskieren. Die folgende Regel sorgt bei allen Paketen für Masquerading, die gemäß der input- und output-Chains der Firewall das externe Interface passieren dürfen:

```
ipchains -A forward -i $EXTERNAL_INTERFACE \  
-s $CLIENT_LAN -j MASQ
```

So eine allgemeine Masquerading-Regel erlaubt nicht etwa, dass aller LAN-Traffic das externe Interface passieren darf: Welche Pakete tatsächlich erlaubt sind, das entscheiden die Regeln für das externe Interface. Diese Regel aktiviert nur das *Masquerading* für alle Pakete, die nach den übrigen Firewall-Regeln zwischen LAN und Internet ausgetauscht werden können.

... nach Dienst

Wenn Sie genauer vorgehen oder den Computern im LAN nur einen Teil der externen Dienste zugänglich machen möchten, legen Sie die Regeln je nach Dienst separat fest. Alles, was nicht in einer Forward- und einer Masquerade-Regel angegeben wird, gelangt nicht ins Internet. Die folgende Regel beispielsweise erlaubt lokalen Web-Browsern den Zugang zu externen Web-Servern:

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p tcp \  
-s $CLIENT_LAN $UNPRIVPORTS \  
-d $ANYWHERE 80 -j MASQ
```

... nach Interface, Dienst und IP-Adresse

Im Extremfall gibt man in den Firewall-Regeln Interface, Dienst und IP-Adressen genau an. Im vorherigen Beispiel mit zwei LANs mussten alle Geräte auf beiden LANs einen Computer auf dem Server-LAN (DNS_SERVER) als Nameserver verwenden. Nur DNS_SERVER selbst darf auf externe Nameserver zugreifen. Weil auf diesem Computer ein Forwarding-Nameserver läuft, der alle Anfragen nach

außen weiterleitet, kommuniziert er mit dem externen Nameserver über den UDP-Port 53. Obwohl `SERVER_LAN_INTERFACE` in der Masquerading-Regel nicht explizit angegeben wird, ist dieses Interface doch impliziert, weil die IP-Adresse `DNS_SERVER` im Server-LAN liegt. D.h. niemand sonst darf auf den externen Nameserver zugreifen.

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p udp \
-s $DNS_SERVER 53 \
-d externer DNS-Server 53 -j MASQ
```

Wenn der Server als DNS-Client arbeiten würde (und nicht als Forwarding-Server), gälten folgende Regeln:

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p udp \
-s $DNS_SERVER $UNPRIVPORTS \
-d externer DNS-Server 53 -j MASQ
```

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p tcp \
-s $DNS_SERVER $UNPRIVPORTS \
-d externer DNS-Server 53 -j MASQ
```

Masquerading und die Rangfolge der Firewall-Regeln

Bei der Firewall aus Kapitel 3, die für einen einzelnen Computer gedacht war, wurden spezifische Ein- und Ausgaberegeln nur für das externe Interface definiert. Bei einem kleinen LAN ist es am einfachsten, beim internen Interface alles zuzulassen, und alles, was für eine externe Adresse gedacht ist, zu forwarden und zu maskieren. Für das gesamte LAN gelten zwar Eingabe-, Ausgabe- und Forward-Regeln, aber die Regeln für das externe Interface sind die einzigen Filterregeln.

Innerhalb jeder Chain, ob `input`, `output` oder `forward`, werden die Regeln in der Reihenfolge angewendet, in der sie definiert wurden. Die erste passende Regel gilt. Wenn man ein LAN betreibt und Forwarding bzw. Masquerading einsetzt, stellt sich die Frage, in welcher Reihenfolge die einzelnen Chains durchlaufen werden.

Wie Bild 4.3 zeigt, kommt jedes Paket auf der `input`-Chain seines Interfaces an. Die Regeln der `input`-Chain werden also als Erstes überprüft. Wenn das Paket nach diesen Regeln passieren darf, entscheidet der Empfänger des Paketes über die nächste Chain. Wenn der Empfänger der lokale Computer ist, gelangt das Paket in die `output`-Chain des `loopback`-Interfaces. Wenn der Empfänger aber eine andere Maschine ist, wird das Paket an die `forward`-Chain weitergegeben. Wenn es diese passiert, wird Masquerading angewendet und das Paket anschließend in die `output`-Chain des `Gateway`-Interfaces geschickt. Die Regeln der `output`-Chain werden also als Letzte überprüft. Wenn das Paket auch hier akzeptiert wird, verlässt es den Rechner bzw. gelangt zum Empfänger.

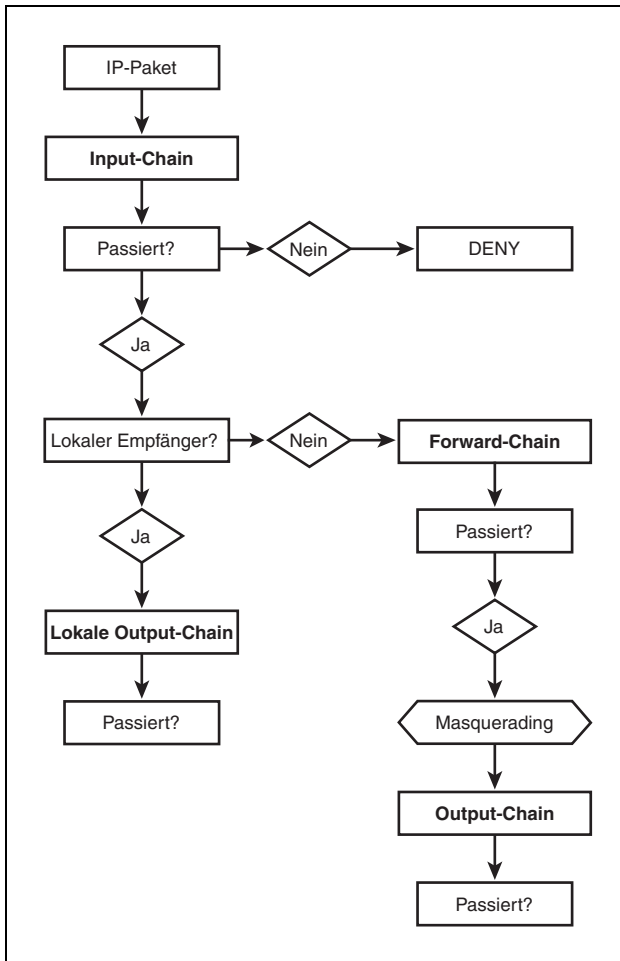


Bild 4.3: Masquerading und die Rangfolge von Firewall-Regeln

Hinweis

Rangfolge der Firewall-Regeln

Im IPCHAINS-HOWTO finden Sie eine vollständigere Diskussion der Rangfolge von Firewall-Regeln.

4.3.4 Portumleitung und transparente Proxies

Portumleitung ist ein Sonderfall von Forwarding und Masquerading auf dem *lokalen* System. Forwarding und Masquerading werden dabei nicht zwischen verschiedenen Netzwerk-Interfaces eingesetzt, sondern alle passenden Pakete – unabhängig von Empfänger-Adresse und -Port – werden an einen Port auf dem lokalen Computer umgeleitet.

Mit diesem Feature realisiert man gerne Proxies auf der Anwendungsebene. Häufig muss man bei der Verwendung eines Proxies ja das Client-Programm umkonfigurieren oder sogar durch andere Software ersetzen, die mit einem Proxy umgehen kann. Dank Portumleitung ist diese Umleitung für das Client-Programm nicht erkennbar, außer wenn ein besonderes Proxy-Protokoll eingesetzt wird. (Z.B. ist HTTP direkt zu einem Web-Server nicht das gleiche wie HTTP zu einem Proxy-Server.)

Angenommen, Sie benutzen einen Proxy für `telnet`. Er erwartet Anfragen auf dem internen LAN-Interface, auf dem normalen `telnet`-Port 23. Alle vom LAN abgehenden `telnet`-Verbindungen werden zu diesem lokalen Proxy umgeleitet. Der Proxy stellt dann die eigentliche Verbindung her. Die zugehörige Firewall-Regel für dieses Szenario lautet:

```
ipchains -A input -i $CLIENT_LAN_INTERFACE -p tcp \
-s $CLIENT_LAN $UNPRIVPORTS \
-d $ANYWHERE 23 -j REDIRECT 23
```

Hinweis

Kernel-Support für transparentes Proxying

Unterstützung für transparentes Proxying muss im Kernel einkompiliert sein, bevor Sie dieses Feature benutzen können. Beim Default-Kernel von Red Hat 6.0 ist das bereits der Fall, bei älteren Distributionen müssen Sie es gegebenenfalls noch nachholen.

4.3.5 Aus dem Internet ankommende Verbindungen zu internen Servern umleiten

Weil Computer aus dem lokalen Netz, die Sie durch Masquerading verstecken, aus dem Internet nicht sichtbar sind, können fremde Clients auf Services nicht zugreifen, die auf diesen lokalen Computern laufen. Allerdings existiert experimenteller Kernel-Code, der ankommende Verbindungen zu internen, maskierten Maschinen erlaubt. Dieses Feature aktivieren Sie, indem Sie das Masquerading-Modul und Unterstützung für `ipportfw masq` in den Kernel hineinkompilieren. Diese experimentellen Features benötigen das Programm `ipmasqadm`, das bei den üblichen Linux-Distributionen nicht enthalten ist und separat aus dem Internet heruntergeladen werden muss.

182 ——— Konfigurationsmöglichkeiten für ein größeres oder weniger vertrauenswürdiges LAN

Beispiel: Angenommen, Sie wollen ankommende Verbindungen zu Ihrem Web-Server an einen internen Computer umleiten, der durch Masquerading verborgen ist. Zusätzlich zu den normalen Firewall-Regeln benötigen Sie noch zwei `ipmasqadm`-Befehle. Der erste löscht die Port-Forwarding-Chain, der zweite leitet ankommende TCP-Verbindungen an den HTTP-Port 80 der Firewall auf einen internen Web-Server auf der IP-Adresse 192.168.3.5 um:

```
ipmasqadm portfw -f
ipmasqadm portfw -a -P tcp -L $IPADDR 80 -R 192.168.3.5 80
```

Hinweis***Port-Forwarding***

Weitere Informationen zu diesen experimentellen Features finden Sie unter <http://juanjo.xlinuxhq.com>, <http://www.monmouth.demon.uk/ipsubs/portforwarding.html>, <ftp://ftp.compsoc.net/users/steve/ipportfw/Linux21>, <http://ip-masq.cjb.net> sowie online im *IP-Masquerade-HOWTO*, */usr/doc/HOWTO/IP-Masquerade-HOWTO*.

Eine Alternative ist ein lokaler Proxy-Server für den Service, den Sie anbieten wollen. Proxies lassen sich sowohl für abgehende als auch für ankommende Verbindungen einsetzen. Man stellt sich einen Proxy oft als Gateway auf Anwendungsebene vor, das abgehende Verbindungen zu externen Diensten maskiert. Proxies für ankommende Verbindungen zu lokalen Diensten dienen meistens der Sicherung der Protokollspezifikation sowie einer detaillierteren Zugriffskontrolle, als sie durch Paketfilter erreicht werden kann. Der Proxy-Server leitet die ankommende Verbindung an einen lokalen Server hinter dem Masquerading weiter. Allerdings setzen Proxies oft spezielle Client-Software oder zumindest eine entsprechende Konfiguration des Clients voraus.

Schließlich gibt es noch einen weiteren Ansatz für Sites, die mehrere offizielle IP-Adressen besitzen. Dabei akzeptiert die Firewall auf dem externen Interface alle Pakete mit einer Empfänger-Adresse innerhalb des lokalen Adressbereichs. Bestimmte Verbindungstypen werden dann an bestimmte lokale Server weitergeleitet. Z.B. leitet die folgende Regel alle ankommenden Verbindungen an einen Web-Server im LAN weiter:

```
ipchains -A forward -i $SERVER_LAN_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $WEB_SERVER 80 -j ACCEPT
```

Bei dieser Konfiguration sind externe Verbindungen nur zum Webserver erlaubt. Fremde Computer haben keinerlei Zugriff auf andere Dienste des internen Servers.

4.4 Eine formale Firewall mit abgeschirmtem Subnetz

Eine kleinere oder mittlere Firma will möglicherweise in eine ausgefeiltere Firewall-Architektur investieren: Ein Computer mit zwei Netzwerkkarten dient als Bastion-Firewall zwischen Internet und internem LAN, wie am Ende von Kapitel 3. Allerdings ist seine interne Netzwerkkarte nicht mehr direkt an das private LAN angeschlossen, sondern liegt in einem Perimeter-Netzwerk, einer DMZ. Öffentlich zugängliche Dienste laufen auf Geräten im Perimeternetz, für die jeweils separate Firewall-Regeln und Sicherheitsbestimmungen existieren. Die öffentlichen Server verfügen möglicherweise über eigene, öffentlich sichtbare IP-Adressen, je nachdem, wie Ihre Ressourcen in dieser Hinsicht verteilt sind.

Bei der Konfiguration einer DMZ wählt man in der Regel einen von zwei Wegen. Bei der ersten Variante verfügt die Bastion-Firewall über drei Netzwerkkarten, eines zum Internet sowie zwei interne, die den Computer mit zwei getrennten LANs verbinden. Das eine LAN ist eine DMZ für die öffentlichen Server; das andere ein privates, lokales Netz. Bild 4.4 zeigt diese Konfiguration.

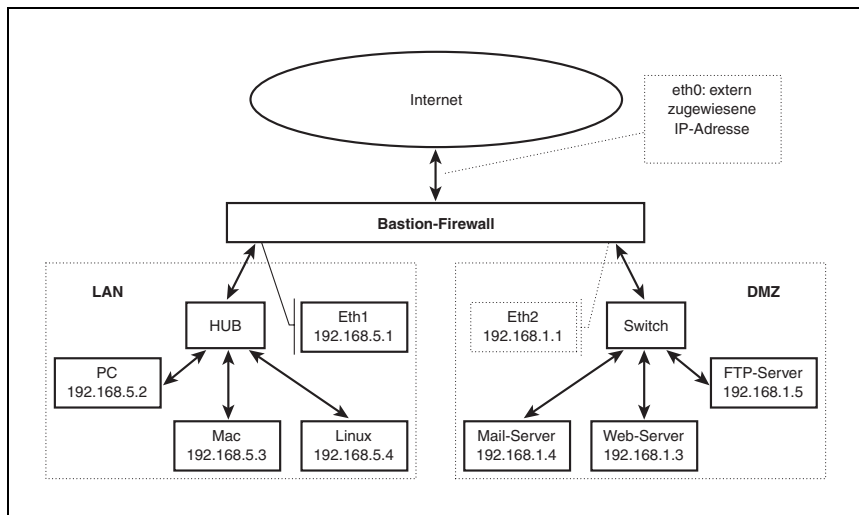


Bild 4.4: DMZ und lokales Netz sind voneinander getrennt

Bei der zweiten Variante verwendet man einen zweiten Computer als Firewall, und zwar als so genannte *Choke-Firewall*². Die Choke sitzt am anderen Ende des Perimeter-Netzwerkes und bildet ein Gateway zwischen DMZ und privatem LAN. Das interne Interface der Choke liegt im privaten LAN. Alle internen Maschinen treten durch Masquerading gegenüber der Bastion und den DMZ-Computern mit der externen IP-Adresse der Choke auf.

2. Wörtl. »Drossel«; bei der Übersetzung wird der englische Fachbegriff benutzt.

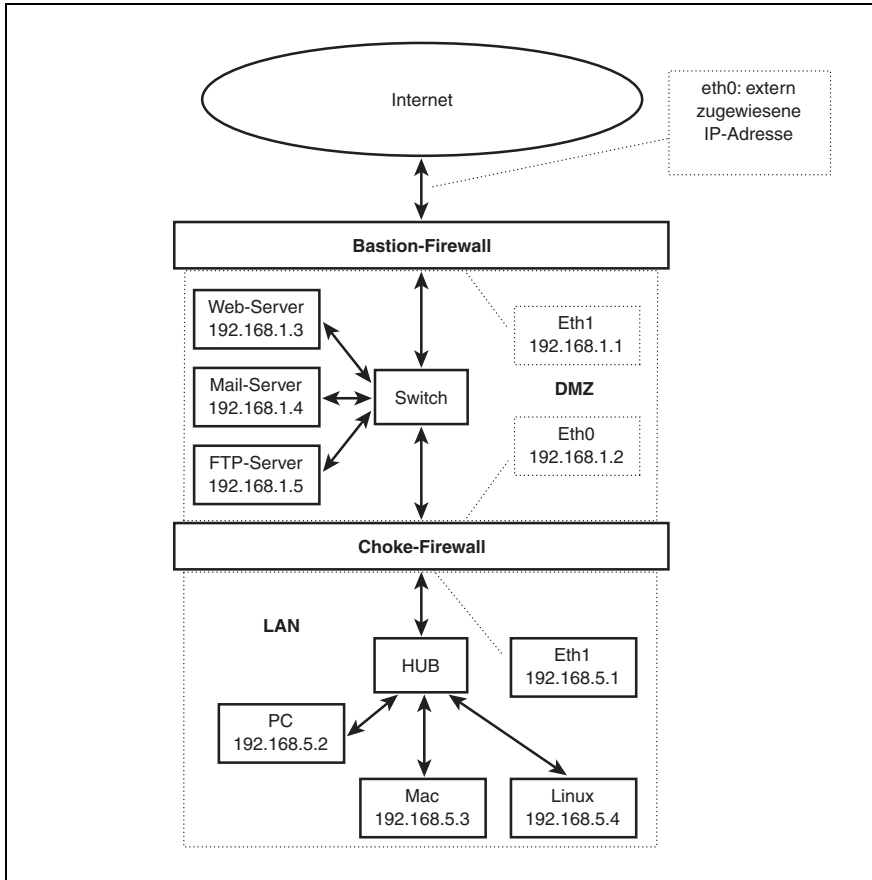


Bild 4.5: Firewall-Architektur mit einem Perimeter-Netzwerk sowie einem abgeschirmtem Subnetz hinter einer Choke-Firewall

Die Bastion-Firewall führt noch einmal Masquerading durch, sodass die Choke dies eigentlich nicht tun müsste. Allerdings vereinfachen sich die Firewall-Regeln für die Bastion, wenn man sich auf die privaten Rechner einfach über die IP-Adresse der Choke beziehen kann.

Im Gegensatz zu unserem Beispiel aus Kapitel 3 existiert bei dieser Konfiguration kein einzelner Knackpunkt mehr, dessen Ausfall die gesamte Site exponieren würde. Dienste, für die unterschiedliche Sicherheitserwägungen gelten, kann man von Netzwerkbereichen mit unterschiedlichen Sicherheitseinstellungen anbieten.

Der Grundgedanke besteht darin, dass man das private Netz durch eine interne Choke-Firewall von der äußeren Bastion-Firewall isoliert. Das Perimeter-Netzwerk aus Bild 4.5 braucht nicht wirklich ein vollständiges Netz mit eigenen Servern zu sein, es veranschaulicht nur das Konzept dieser Trennung. Im Extremfall

reicht ein einfaches Crossover-Kabel zwischen dem internen Interface der Bastion und dem externen Interface der Choke völlig aus.

Eine DMZ, die lediglich aus einem Crossover-Kabel besteht, hört sich vielleicht unsinnig an. Genau wie ein großes Perimeter-Netz erhält man dadurch aber eine Absicherung durch zwei Firewalls, man vermeidet eine einzelne Ausfallstelle. Lokale LAN-Dienste betreibt man auf der Choke, in völliger Isolation von Bastion und Internet.

Wenn Sie ein Crossover-Kabel als DMZ benutzen wollen, brauchen Sie auf dem internen Interface der Bastion keine ausführlichen Firewall-Regeln. Die externen Filter der Choke reichen in diesem Fall aus.

Im Rest dieses Kapitels gehen wir davon aus, dass Bastion und Choke beide als Gateways zu einer DMZ dienen. Die DMZ enthält öffentlich und halböffentlich zugängliche Server. Jedes Netzwerkinterface der beiden Firewall-Computer verfügt über eigene, individuelle Regeln. Wir benötigen also mindestens vier Regelsätze: je einen für externes und internes Interface beider Maschinen. Die Regeln für das externe Interface der Bastion sind fast identisch zu den externen Regeln aus Kapitel 3. (Die Regeln für das interne Interface der Choke könnten mit denen der Firewall mit zwei Netzwerkkarten aus Kapitel 3 übereinstimmen; in diesem Kapitel habe ich sie allerdings ein wenig ergänzt, um einen internen DHCP-Server zu beschreiben.)

Der eigentliche Unterschied zwischen diesem Beispiel und dem aus Kapitel 3 liegt in dem zusätzlichen Perimeter-Netzwerk, der DMZ – die neuen Regeln beziehen sich also auf das interne Interface der Bastion und das externe Interface der Choke. Die Regeln für diese beiden verhalten sich zueinander spiegelbildlich. Für öffentliche Server in der DMZ gibt es noch zusätzliche, individuelle Regeln. Bei diesen öffentlichen Servern der DMZ handelt es sich in der Regel um spezialisierte Maschinen, auf denen jeweils nur ein einziger Dienst läuft. Die entsprechenden Firewall-Regeln sind daher relativ einfach und lassen nur Zugriffe auf diesen einen Dienst zu.

Die symbolischen Konstanten und Grundregeln für die Firewall auf der Choke sind im Wesentlichen identisch zu denen der Bastion. Weil die externen Regeln der Bastion im Vergleich zu Kapitel 3 unverändert bleiben, werden in diesem Kapitel nur die Unterschiede erwähnt, sofern welche auftreten. Das Hauptaugenmerk liegt in diesem Kapitel auf den Firewall-Regeln der Choke, sowie auf der Symmetrie zwischen dem internen Interface der Bastion und dem externen der Choke.

4.4.1 Symbolische Konstanten für die Firewall-Beispiele

Wie beim Firewall-Beispiel aus Kapitel 3 liegt das externe Interface der Bastion auf `eth0` und führt ins Internet. Das interne Interface ist `eth1` mit der IP-Adresse `192.168.1.1`; es führt ins Perimeter-Netzwerk, also in die DMZ. Das externe Interface der Choke ist `eth0` mit der IP `192.168.1.2`, es führt ebenfalls in die

DMZ. Dem internen Interface der Choke auf eth1 ist die IP 192.168.5.1 zugewiesen; es liegt im privaten LAN.

Symbole für die Bastion-Firewall

Die folgenden Konstanten müssen im Firewall-Skript der Bastion ergänzt werden. Sie beziehen sich auf das interne Interface der Bastion, auf die IP-Adresse der Choke und auf den Adressblock für die DMZ.

```
BASTION_DMZ_INTERFACE="eth1"           # internes Interface der Bastion
BASTION_DMZ_IPADDR="192.168.1.1"       # zugehörige IP-Adresse
CHOKE_DMZ_IPADDR="192.168.1.2"         # externes Interface der Choke
DMZ_ADDRESSES="192.168.1.0/24"         # IP-Bereich der DMZ
DMZ_BROADCAST="192.168.1.255"         # Broadcast-Adresse für die DMZ
```

Symbole für die Choke-Firewall

Ein Firewall-Skript lässt sich am leichtesten lesen und überarbeiten, wenn man für immer wiederkehrende Namen und Adressen symbolische Konstanten benutzt. Die folgenden Konstanten werden entweder in den Beispielen aus diesem Kapitel immer wieder verwendet, oder es handelt sich um universelle Konstanten aus den Netzwerk-Standards.

```
CHOKE_DMZ_INTERFACE="eth0"             # externes Interface der Choke
CHOKE_LAN_INTERFACE="eth1"             # internes Interface der Choke
LOOPBACK_INTERFACE="lo"                # Loopback-Interface

CHOKE_DMZ_IPADDR="192.168.1.2"         # externe IP-Adresse der Choke
BASTION_DMZ_IPADDR="192.168.1.1"       # interne IP-Adresse der Bastion
DMZ_ADDRESSES="192.168.1.0/24"         # IP-Bereich der DMZ
DMZ_BROADCAST="192.168.1.255"         # Broadcast-Adresse für die DMZ
CHOKE_LAN_IPADDR="192.168.5.1"         # interne IP-Adresse der Choke
CHOKE_LAN_ADDRESSES="192.168.5.0/24"   # IP-Bereich des privaten LANs
ANYWHERE="any/0"                      # passt auf jede IP-Adresse

LOOPBACK="127.0.0.0/8"                # Loopback-Bereich
CLASS_A="10.0.0.0/8"                  # private Netzwerke der Klasse A
CLASS_B="172.16.0.0/12"                # private Netzwerke der Klasse B
CLASS_C="192.168.0.0/16"                # private Netzwerke der Klasse C
CLASS_D_MULTICAST="224.0.0.0/4"         # Multicast-Adressen der Klasse D
CLASS_E_RESERVED_NET="240.0.0.0/5"     # reservierte Adressen der Klasse E
BROADCAST_SRC="0.0.0.0"                # Absender-Adresse für Broadcasts
BROADCAST_DEST="255.255.255.255"       # Empfänger-Adresse für Broadcasts
PRIVPORTS="0:1023"                    # privilegierte Ports
UNPRIVPORTS="1024:65535"               # unprivilegierte Ports
```

Hier nicht aufgeführte Konstanten definiere ich kurz vor ihrer Verwendung bei den jeweiligen Regeln.

4.4.2 Löschen alter Firewall-Regeln

Wenn Sie neue Regeln für Ihre Firewall festlegen, löschen Sie zuerst immer die eventuell noch gültigen alten Regeln aus den Chains heraus. Ansonsten hängt der Kernel die neuen Regeln einfach ans Ende der alten Chains an. Vermutlich würde dann jedesmal eine der alten Regeln auf ein Paket zutreffen, und die neuen Regeln würden nie benutzt. Das folgende Kommando löscht alle drei eingebauten Ketten (input, output und forward):

```
# Vorbestehende Regeln löschen  
ipchains -F
```

Die Chains sind jetzt leer, Sie können sie ganz von Anfang an definieren. Die alte Policy ist zwar noch gesetzt, aber wir werden als Nächstes eine neue Policy festlegen.

4.4.3 Festlegung der voreingestellten Policy für die Choke

Die Choke soll, soweit nichts anderes festgelegt wurde, alle Pakete in beide Richtungen verwerfen. Dabei wird aber eine ICMP-Fehlermeldung vom Typ 3 zurückgegeben, sodass der Client sofort eine Rückmeldung über den unerlaubten Zugriff erhält und nicht erst auf einen Timeout warten muss.

Für beide Firewalls gilt also die Voreinstellung, alles zu verwerfen, nicht alles anzunehmen.

```
# Per Voreinstellung werden Pakete mit einer Fehlermeldung abgewiesen  
ipchains -P input REJECT  
ipchains -P output REJECT  
ipchains -P forward REJECT
```

An diesem Punkt ist sämtlicher Netzwerkverkehr gesperrt.

4.4.4 Einschalten des Loopback-Interfaces auf der Choke

Für das Loopback-Gerät sollten keine Beschränkungen gelten. Dadurch können Sie lokal beliebige Netzwerkdienste betreiben, auch solche, ohne die das System nicht auskommt. Sie müssen sich dafür nicht darum kümmern, alle entsprechenden Regeln einzeln festzulegen.

```
# Keine Beschränkungen für Loopback  
ipchains -A input -i $LOOPBACK_INTERFACE -j ACCEPT  
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT
```

4.4.5 Gefälschte Absender und andere fehlerhafte Adressen (Klassen A bis C)

In diesem Abschnitt definieren wir Filter für Absender- und Empfänger-Adressen, die normalerweise nicht auf dem Internet vorkommen dürfen.

Auf der Ebene des Paketfilters ist eine der ganz wenigen Fälschungen, die Sie mit Sicherheit erkennen können, Ihre eigene IP-Adresse. Die folgende Regel verwirft alle ankommenden Pakete, die vorgeblich von Ihnen selbst abgeschickt wurden:

```
# Gefälschte Pakete verwerfen, die angeblich von unserer IP kommen
ipchains -A input -i $CHOKE_DMZ_INTERFACE \
        -s $CHOKE_DMZ_IPADDR -j REJECT -1
ipchains -A input -i $CHOKE_LAN_INTERFACE \
        -s $CHOKE_LAN_IPADDR -j REJECT -1
```

Die nächsten beiden Gruppen verbieten ankommende und abgehende Pakete mit einer Absender- oder Empfängeradresse aus den privaten Netzwerkklassen A und B. Solche Pakete sollten außerhalb eines LANs nicht existieren.

```
# Pakete mit privaten Klasse-A-Adressen in Absender oder Empfänger
  verwerfen
ipchains -A input -i $CHOKE_DMZ_INTERFACE -s $CLASS_A -j REJECT -1
ipchains -A output -i $CHOKE_DMZ_INTERFACE -d $CLASS_A -j REJECT -1

ipchains -A input -i $CHOKE_LAN_INTERFACE -s $CLASS_A -j REJECT -1
ipchains -A output -i $CHOKE_LAN_INTERFACE -d $CLASS_A -j REJECT -1

# Pakete mit privaten Klasse-B-Adressen in Absender oder Empfänger
  verwerfen
ipchains -A input -i $CHOKE_DMZ_INTERFACE -s $CLASS_B -j REJECT -1
ipchains -A output -i $CHOKE_DMZ_INTERFACE -d $CLASS_B -j REJECT -1

ipchains -A input -i $CHOKE_LAN_INTERFACE -s $CLASS_B -j REJECT -1
ipchains -A output -i $CHOKE_LAN_INTERFACE -d $CLASS_B -j REJECT -1
```

Adressen aus dem privaten Bereich der Klasse C benutzen wir selbst. Die Choke-Firewall darf also nicht alle privaten Klasse-C-Adressen blockieren, weil sowohl ihr externes als auch ihr internes Interface eine solche Klasse-C-Adresse verwenden. (ipfwadm ist nicht flexibel genug, alle privaten Klasse-C-Adressen außer unseren eigenen zu sperren. Mit ipchains hingegen lässt sich so etwas durch die Verwendung von benutzerdefinierten Chains bewerkstelligen.)

Die folgenden Regeln verbieten Pakete mit einer Absender-Adresse im Loopback-Bereich:

```
# Pakete aus dem oder an den Loopback-Bereich verwerfen
ipchains -A input -i $CHOKE_DMZ_INTERFACE -s $LOOPBACK -j REJECT -1
ipchains -A input -i $CHOKE_LAN_INTERFACE -s $LOOPBACK -j REJECT -1
```

Die nun folgenden Regeln benötigen wir nur, damit die entsprechenden Pakete auch protokolliert werden. Weil wir unsere Firewall so konfigurieren, dass sie alle unbekannten Pakete blockiert, werden Broadcast-Adressen ohnehin schon verworfen und müssten, falls sie irgendwo benötigt werden, explizit erlaubt werden.

```
# Fehlerhafte Broadcast-Pakete verwerfen
ipchains -A input -i $CHOKE_DMZ_INTERFACE \
        -s $BROADCAST_DEST -j REJECT -l
ipchains -A input -i $CHOKE_DMZ_INTERFACE \
        -d $BROADCAST_SRC -j REJECT -l

ipchains -A input -i $CHOKE_LAN_INTERFACE \
        -s $BROADCAST_DEST -j REJECT -l
ipchains -A input -i $CHOKE_LAN_INTERFACE \
        -d $BROADCAST_SRC -j REJECT -l
```

Multicast-Adressen dürfen nur im Empfängerfeld auftauchen. Den Gebrauch gefälschter Multicast-Adressen verbieten wir nicht nur, wir zeichnen ihn auch auf:

```
# Multicast-Adressen der Klasse D dürfen nicht als Absender vorkommen
ipchains -A input -i $CHOKE_DMZ_INTERFACE -s $MULTICAST -j REJECT -l
ipchains -A output -i $CHOKE_DMZ_INTERFACE -s $MULTICAST -j REJECT -l

ipchains -A input -i $CHOKE_LAN_INTERFACE -s $MULTICAST -j REJECT -l
ipchains -A output -i $CHOKE_LAN_INTERFACE -s $MULTICAST -j REJECT -l
```

Das Gleiche gilt schließlich auch für Pakete aus einem reservierten Netz der Klasse E:

```
# Reservierte IP-Adressen der Klasse E verbieten
ipchains -A input -i $CHOKE_DMZ_INTERFACE -s $RESERVED_NET -j REJECT -l
ipchains -A input -i $CHOKE_LAN_INTERFACE -s $RESERVED_NET -j REJECT -l
```

4.4.6 ICMP-Nachrichten filtern

Bei aktiviertem Masquerading behandelt der Kernel ICMP-Nachrichten etwas anders. Wenn nicht auch ICMP-Masquerading eingeschaltet ist, werden nur ICMP-Fehlermeldungen für bestehende Verbindungen weitergeleitet und maskiert. Wenn ICMP-Masquerading aktiv ist, können die internen Maschinen zusätzlich auch andere Kontrollnachrichten verschicken, wie sie z.B. von ping und traceroute verwendet werden. Ansonsten lassen sich solche ICMP-Nachrichten nur zwischen den lokalen Rechnern auf demselben Netzabschnitt verschicken. Ab Red Hat 6.0 ist ICMP-Masquerading schon eingeschaltet, bei früheren Versionen müssen Sie es noch in den Kernel hineinkompilieren.

Wie bei Masquerading im Allgemeinen gilt auch für das ICMP-Masquerading, dass Ihre lokalen Computer einen ICMP-Austausch mit fremden Rechnern anfangen können, aber nicht umgekehrt. Aus dem Internet ist nur das externe Interface der Bastion sichtbar.

Kontrollnachricht source-quench (Typ 4)

Eine ICMP-Nachricht vom Typ 4, `source-quench`, wird generiert, wenn jemand – normalerweise ein Router – Daten schneller schickt, als der Empfänger sie annehmen kann. Damit handelt es sich bei der `source-quench`-Nachricht um eine primitive Art der Flusskontrolle auf der Netzwerkschicht des Internet-Protokolls. Sie wird meistens zwischen zwei benachbarten Maschinen eingesetzt.

Bastion und Choke akzeptieren alle `source-quench`-Nachrichten, die bei der Interaktion mit der DMZ entstehen.

source-quench-Konfiguration für die DMZ-Seite der Bastion

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p icmp \
-s $BASTION_DMZ_IPADDR 4 -d $DMZ_ADDRESSES -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p icmp \
-s $DMZ_ADDRESSES 4 -d $BASTION_DMZ_IPADDR -j ACCEPT
```

source-quench-Konfiguration für die DMZ-Seite der Choke

```
ipchains -A input -i $CHOKES_DMZ_INTERFACE -p icmp \
-s $DMZ_ADDRESSES 4 -d $CHOKES_DMZ_IPADDR -j ACCEPT

ipchains -A output -i $CHOKES_DMZ_INTERFACE -p icmp \
-s $CHOKES_DMZ_IPADDR 4 -d $DMZ_ADDRESSES -j ACCEPT
```

Statusmeldung parameter-problem (Typ 12)

Eine ICMP-Nachricht vom Typ 12 (`parameter-problem`) zeigt an, dass ein empfangenes Paket ungültige oder unerwartete Daten im Header enthält, oder dass die Prüfsumme im Header nicht mit der vom Empfänger errechneten Prüfsumme übereinstimmt.

Bastion und Choke akzeptieren alle `parameter-problem`-Nachrichten, die bei der Interaktion mit der DMZ entstehen.

parameter-problem-Konfiguration für die DMZ-Seite der Bastion

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p icmp \
-s $ANYWHERE 12 -d $DMZ_ADDRESSES -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p icmp \
-s $DMZ_ADDRESSES 12 -d $ANYWHERE -j ACCEPT
```

parameter-problem-Konfiguration für die DMZ-Seite der Choke

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p icmp \  
-s $ANYWHERE 12 -d $CHOKE_DMZ_IPADDR -j ACCEPT  
  
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p icmp \  
-s $CHOKE_DMZ_IPADDR 12 -d $ANYWHERE -j ACCEPT
```

Fehlermeldung destination-unreachable (Typ 3)

Bei der ICMP-Meldung Typ 3 (destination-unreachable) handelt es sich um eine allgemeine Fehlermeldung.

Bastion und Choke akzeptieren alle destination-unreachable-Nachrichten, die bei der Interaktion mit der DMZ entstehen.

destination-unreachable-Konfiguration für die DMZ-Seite der Bastion

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p icmp \  
-s $ANYWHERE 3 -d $DMZ_ADDRESSES -j ACCEPT  
  
ipchains -A input -i $BASTION_DMZ_INTERFACE -p icmp \  
-s $DMZ_ADDRESSES 3 -d $ANYWHERE -j ACCEPT
```

destination-unreachable-Konfiguration für die DMZ-Seite der Choke

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p icmp \  
-s $ANYWHERE 3 -d $CHOKE_DMZ_IPADDR -j ACCEPT  
  
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p icmp \  
-s $CHOKE_DMZ_IPADDR 3 -d $ANYWHERE -j ACCEPT
```

Statusmeldung time-exceeded (Typ 11)

ICMP-Nachrichten vom Typ 11 (time-exceeded) bedeuten, dass ein Timeout aufgetreten ist, genauer gesagt, dass ein Paket öfter als erlaubt von einem Router zum nächsten weitergegeben wurde. Die Time-To-Live (TTL) wurde überschritten. Bei unseren heutigen Netzwerken tritt time-exceeded normalerweise nur noch als Antwort auf traceroute auf.

time-exceeded-Konfiguration für die DMZ-Seite der Bastion

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p icmp \  
-s $BASTION_DMZ_IPADDR 11 -d $DMZ_ADDRESSES -j ACCEPT  
  
ipchains -A input -i $BASTION_DMZ_INTERFACE -p icmp \  
-s $DMZ_ADDRESSES 11 -d $BASTION_DMZ_IPADDR -j ACCEPT
```

time-exceeded-Konfiguration für die DMZ-Seite der Choke

Die Choke darf nur mit der Bastion Nachrichten vom Typ `time-exceeded` austauschen.

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p icmp \
-s $BASTION_DMZ_IPADDR 11 -d $CHOKE_DMZ_IPADDR -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p icmp \
-s $CHOKE_DMZ_IPADDR 11 -d $BASTION_DMZ_IPADDR -j ACCEPT
```

Kontrollmeldungen `echo-request` (Typ 8) und `echo-reply` (Typ 0)

Der Befehl `ping` verwendet zwei verschiedene Nachrichtentypen. Die Anfrage (`echo-request`) hat den ICMP-Typ 8, die Antwort (`echo-reply`) den Typ 0. `ping` ist ein einfaches Tool zur Netzwerkanalyse, das noch aus der Zeit des DARPA-NET stammt.

ping-Konfiguration für die DMZ-Seite der Bastion

Die folgenden beiden Regeln erlauben allen Computern in der DMZ, Rechner im Internet anzupingen:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p icmp \
-s $DMZ_ADDRESSES 8 -d $ANYWHERE -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p icmp \
-s $ANYWHERE 0 -d $DMZ_ADDRESSES -j ACCEPT
```

Umgekehrt darf aber nur die Bastion die Computer in der DMZ anpingen, alle anderen ankommenden pings sind nicht aufgeführt und daher blockiert:

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p icmp \
-s $BASTION_DMZ_IPADDR 8 -d $DMZ_ADDRESSES -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p icmp \
-s $DMZ_ADDRESSES 0 -d $BASTION_DMZ_IPADDR -j ACCEPT
```

ping-Konfiguration für die DMZ-Seite der Choke

Die Choke darf jeden Computer im Internet anpingen:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p icmp \
-s $CHOKE_DMZ_IPADDR 8 -d $ANYWHERE -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p icmp \
-s $ANYWHERE 0 -d $CHOKE_DMZ_IPADDR -j ACCEPT
```

Ankommende pings hingegen sind nur aus der DMZ erlaubt:


```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p icmp \
-s $DMZ_ADDRESSES 8 -d $CHOKE_DMZ_IPADDR -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p icmp \
-s $CHOKE_DMZ_IPADDR 0 -d $DMZ_ADDRESSES -j ACCEPT
```

Hinweis

Kernel-Unterstützung für ICMP-Masquerading

Wenn Sie in einem LAN eine RedHat-Version vor 6.0 benutzen, werden die für ping nötigen ICMP-Pakete auch dann nicht vom LAN zu fremden Maschinen weitergeleitet, wenn IP-Forwarding aktiviert ist. Wenn Sie auf einem internen Computer ping benutzen wollen, muss das IP-Masquerading-Modul im Kernel einkompiliert sein. Andere Kontroll- und Fehlermeldungen im Zusammenhang mit Verbindungen anderer Protokolle werden auch ohne dieses Spezialmodul weitergeleitet.

Hinweis

Die Protokoll-Tabellen für TCP und UDP

In den folgenden Abschnitten stelle ich das jeweils behandelte Protokoll immer zuerst tabellarisch dar, bevor ich die ipchains-Regeln anführe.

Jede Zeile dieser Tabellen beschreibt einen Pakettyp. Für jeden dieser Pakettypen benötigt man eine eigene Firewall-Regel.

Die Spalten der Tabellen sind wie folgt:

Beschreibung

enthält ein paar Stichworte zu Funktion und Herkunft des Pakets.

Protokoll

ist das Transportprotokoll: TCP, UDP oder ICMP.

Fremde IP

gibt an, welche Adresse bzw. welcher Adressbereich als fremde IP-Adresse stehen darf. Je nachdem, ob das Paket von unserem System abgeschickt wird oder nicht, kann sich das auf die Empfänger- oder auf die Absender-Adresse beziehen.

Fremder Port

beschreibt die Ports, die auf der Internet-Seite zulässig sind.

Chain

gibt an, in welcher Firewall-Chain – input oder output – dieses Paket erwartet wird. Ein Paket in der input-Chain kommt aus dem Internet, ein Paket in output kommt von Ihrem System und wird ins Internet geschickt.

Lokale IP

gibt an, welche Adresse bzw. welcher Adressbereich als lokale IP-Adresse stehen darf. Je nachdem, ob das Paket von unserem System abgeschickt wird oder nicht, kann sich das auf die Absender- oder auf die Empfänger-Adresse beziehen.

Lokaler Port

beschreibt die Ports, die auf unserer Seite zulässig sind.

TCP-Flags

sind, wie der Name schon sagt, nur für TCP-Pakete definiert. Für die Firewall-Regeln sind nur die Flags SYN und ACK von Bedeutung.

Durch die Angabe der Chain werden Pakete in ankommende und abgehende unterschieden. Adressen und Ports sind als »fremd« oder »lokal« klassifiziert, und zwar immer in Bezug auf das Netzwerkinterface Ihrer Maschine. Bei ankommenden Paketen beziehen sich also »fremde IP« und »fremder Port« auf die Absender-Felder im Paket-Header, während mit »lokaler IP« und »lokalen Port« die Empfänger-Felder gemeint sind. Bei abgehenden Paketen verhält es sich umgekehrt: »fremde IP« und »fremder Port« meinen die Empfänger-Felder, »lokale IP« und »lokaler Port« die Absender-Felder.

In den wenigen Fällen, in denen ICMP-Pakete beteiligt sind, denken Sie daran, dass ICMP-Pakete im Gegensatz zu TCP- oder UDP-Paketen keinen Absender- und Empfänger-Port kennen. Dafür enthält der Header einen so genannten Nachrichtentyp. ICMP-Pakete werden ja nicht an Programme geschickt, die bestimmte Ports benutzen, sondern es handelt sich bei ihnen um Botschaften von einem Computer an einen anderen. In den Tabellen dieses Buches gebe ich den Nachrichtentyp in der Spalte mit dem Absender-Port an. Bei ankommenden ICMP-Paketen ist das die Spalte »fremder Port«, bei abgehenden ICMP-Paketen die Spalte »lokaler Port«.

4.4.7 DNS (UDP- und TCP-Port 53)

Für den Zugang zum Internet benötigen Sie auch Zugriff auf den *Domain Name Service* DNS, der Hostnamen in die entsprechenden IP-Adressen übersetzt. Ohne DNS können Sie auf einen anderen Rechner nicht zugreifen, wenn Sie seine IP-Adresse nicht kennen und direkt eingeben.

Das Kommunikationsprotokoll von DNS verwendet sowohl UDP als auch TCP. Es existieren mehrere Arten von Verbindungen, darunter normale Verbindungen zwischen einem Client und einem Server, Informationsaustausch zwischen einem Forwarding-Server und einem regulären DNS-Server, sowie Verbindungen zwischen einem primären und einem sekundären Nameserver.

In einem Privathaushalt stehen Ihnen mehrere Alternativen der DNS-Konfiguration offen. Wenn `named` auf der Firewall nicht läuft und diese also ein reiner DNS-

Client ist, werden Sie auch die internen Computer so konfigurieren, dass sie – ebenso wie Ihre Firewall – die Nameserver Ihres Internet-Providers verwenden. Falls aber auf der Firewall ein Nameserver aktiv ist, greifen die lokalen Rechner auf das interne Interface der Firewall als DNS-Server zu.

Eine klassische DNS-Konfiguration für ein LAN

Manche Sites benötigen noch mehr Sicherheit. Ein klassisches DNS-Setup versteckt die Information über lokale Computer vor Benutzern aus dem Internet. Damit können in der internen DNS-Datenbank auch vertrauliche Daten über Benutzer und Accounts gespeichert werden.

Bei solch einer Architektur läuft auf der Bastion-Firewall ein öffentlicher DNS-Server. Dieser ist als autoritativ für die Site konfiguriert, verfügt aber nur über unvollständige Daten: Er weiß nichts über die lokalen Computer. Andere Programme auf dem Bastion-Computer greifen überhaupt nicht auf diesen Nameserver zu. Die Datei `/etc/resolv.conf` verweist sie auf die Choke, die Anfragen von DNS-Clients auf der Bastion bedient. Also: Anfragen aus dem Internet erledigt der öffentliche DNS-Server auf der Bastion; für lokale Anfragen ist der interne DNS-Server auf der Choke zuständig.

Der echte DNS-Server für die Site liegt auf der Choke. Dieser interne Server ist ebenfalls als autoritativ konfiguriert. In seinem Fall ist die Information korrekt. Alle Anfragen der lokalen Clients von Choke und Bastion sowie aus dem privaten LAN gehen an den privaten DNS-Server. Bild 4.6 veranschaulicht das. Wenn der Choke-Server angeforderte Daten nicht kennt, fragt er den Bastion-Server, der die Anfrage seinerseits an einen externen Server weiterleitet.

Wenn in den Firewall-Skripten irgendwo symbolische Hostnamen verwendet werden, erzeugt diese Konfiguration ein Henne-und-Ei-Problem, während die Server starten. Beide sind ja voneinander abhängig. Wenn die Bastion gebootet wird, benutzt sie die Choke als Nameserver; wenn die Choke bootet, greift sie auf die Bastion zu. Daher muss die Bastion beim Booten zunächst externe Nameserver für ihre Clients verwenden. Erst wenn der Nameserver auf der Choke-Firewall läuft, dürfen die Clients von der Bastion diesen statt der externen Server benutzen.

Konfiguration der DMZ-Seite der Bastion als öffentlicher Nameserver

Der öffentliche Nameserver ist auf der Bastion zu Hause. Lokale DNS-Clients, die auf der Bastion laufen, greifen aber auf den privaten Server auf der Choke zu. Wenn der Choke-Server eine Anfrage nicht selbst beantworten kann, leitet er sie an den DNS-Server der Bastion weiter, der die Anfrage seinerseits an einen oder mehrere fremde Server weiterreicht. Wenn keiner der fremden Server antwortet, versucht der DNS der Bastion es mit autoritativen Servern noch einmal.

Tabelle 4.2 zeigt die am DNS-Austausch beteiligten Pakete, die über das DMZ-Interface der Bastion gesendet und empfangen werden.

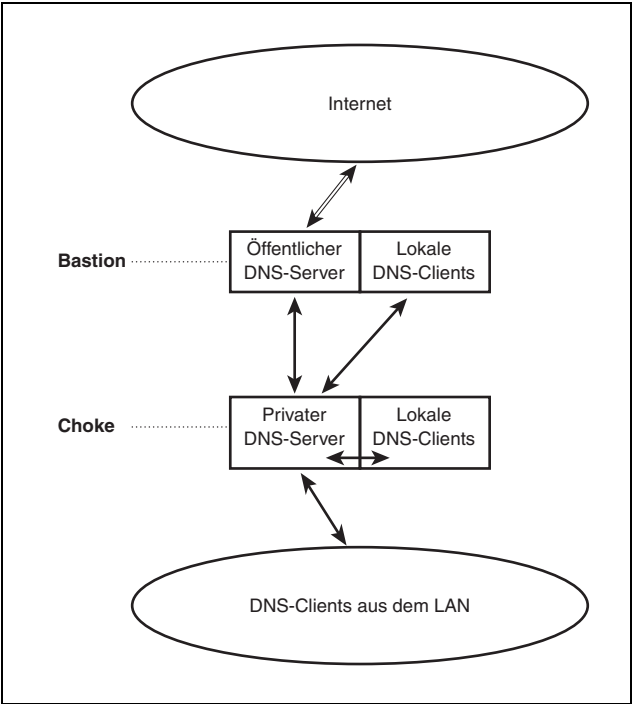


Bild 4.6: Öffentliche und private Nameserver

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage des Servers auf der Choke	UDP	CHOKE_DMZ_ IPADDR	53	input	BASTION_ DMZ_ IPADDR	53	-
Antwort des Servers auf der Bastion	UDP	CHOKE_DMZ_ IPADDR	53	output	BASTION_ DMZ_ IPADDR	53	-
Anfrage eines Clients auf der Bastion	UDP	CHOKE_DMZ_ IPADDR	53	output	BASTION_ DMZ_ IPADDR	1024:65535	-
Antwort des Servers auf der Choke	UDP	CHOKE_DMZ_ IPADDR	53	input	BASTION_ DMZ_ IPADDR	1024:65535	-
Anfrage eines Clients auf der Bastion	TCP	CHOKE_DMZ_ IPADDR	53	output	BASTION_ DMZ_ IPADDR	1024:65535	Egal

Tab. 4.2: DNS-Protokoll auf dem DMZ-Interface der Bastion

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Antwort des Servers auf der Choke	TCP	CHOKE_DMZ_ IPADDR	53	input	BASTION_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.2: DNS-Protokoll auf dem DMZ-Interface der Bastion (Forts.)

Der DNS-Server der Bastion reagiert auf Peer-to-Peer-Anfragen vom DNS-Server der Choke:

```
# Caching- und Forwarding-Nameserver (UDP 53)
# -----

ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR 53 \
-d $BASTION_DMZ_IPADDR 53 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR 53 \
-d $CHOKE_DMZ_IPADDR 53 -j ACCEPT
```

DNS-Clients auf der Bastion richten ihre Anfragen an den Nameserver der Choke:

```
# DNS-Anfragen von lokalen Clients an den Server auf der Choke (TCP/UDP 53)
# -----
---

ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 53 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR 53 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 53 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE ! -y -p tcp \
-s $CHOKE_DMZ_IPADDR 53 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Es schließen sich die ganz normalen DNS-Regeln des externen Interfaces an, für den Zugriff auf externe Nameserver. Die zugehörige Protokolltabelle finden Sie in

Tabelle 3.3. Der Nameserver der Bastion ist ein voller DNS-Server mit eingeschränktem öffentlichem Angebot. Lokale Anfragen werden zunächst an den Server des Internet-Providers weitergeleitet.

```
# Caching- und Forwarding-Nameserver (UDP 53)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR 53 \
-d $NAME_SERVER 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $NAME_SERVER 53 \
-d $IPADDR 53 -j ACCEPT

# DNS: lokale Anfrage, Client an Server (TCP/UDP 53)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $ANYWHERE 53 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE ! -y -p tcp \
-s $ANYWHERE 53 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# DNS: Lokaler Server bedient einen fremden Client (TCP/UDP 53)
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 53 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR 53 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 53 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE ! -y -p tcp \
-s $IPADDR 53 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als privater Nameserver

Der private Nameserver liegt auf der Choke. LAN-Clients – einschließlich der Programme auf der Bastion – greifen auf ihn zu. Wenn der Server eine Anfrage nicht aus dem DNS-Cache beantworten kann, leitet er sie an den Server auf der Bastion weiter.

Die Protokolltabelle finden Sie in Tabelle 4.3.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage des Servers auf der Choke	UDP	BASTION_DMZ_ IPADDR	53	output	CHOKE_ DMZ_ IPADDR	53	-
Antwort des Servers auf der Bastion	UDP	BASTION_DMZ_ IPADDR	53	input	CHOKE_ DMZ_ IPADDR	53	-
Anfrage eines Clients aus der DMZ	UDP	DMZ_ADDRESSES	1024:65535	input	CHOKE_ DMZ_ IPADDR	53	-
Antwort des Servers auf der Choke	UDP	DMZ_ADDRESSES	1024:65535	output	CHOKE_ DMZ_ IPADDR	53	-
Anfrage eines Clients aus der DMZ	TCP	DMZ_ADDRESSES	1024:65535	input	CHOKE_ DMZ_ IPADDR	53	Egal
Antwort des Servers auf der Choke	TCP	DMZ_ADDRESSES	1024:65535	output	CHOKE_ DMZ_ IPADDR	53	ACK

Tab. 4.3: DNS-Protokoll auf dem DMZ-Interface der Choke

Die Firewall-Regeln zu DNS verhalten sich bis zu einem gewissen Maß spiegelbildlich zu den Regeln der Bastion. Der lokale Server leitet Peer-to-Peer-Anfragen an den DNS-Server der Bastion weiter, wenn er sie nicht selbst auflösen kann:

```
# Caching- und Forwarding-Nameserver (UDP 53)
# -----

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR 53 \
-d $BASTION_DMZ_IPADDR 53 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR 53 \
-d $CHOKE_DMZ_IPADDR 53 -j ACCEPT
```

Die Choke bearbeitet Anfragen von Clients aus der gesamten DMZ, einschließlich der Bastion:

```
# DNS-Server bedient Clients aus der DMZ (TCP/UDP 53)
# -----

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 53 -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR 53 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 53 -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE ! -y -p tcp \
-s $CHOKE_DMZ_IPADDR 53 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

4.4.8 Der auth-Service (TCP-Port 113)

Der auth- bzw. identd-Service wird für manche Internet-Dienste benötigt. `identd` liefert den Benutzernamen oder die Benutzerkennung zurück, der eine bestimmte Verbindung zugeordnet ist. Diese Identifikation führen z.B. fremde Mailserver durch, denen Sie eine E-Mail schicken. Sie müssen nicht unbedingt einen `identd`-Server betreiben, aber Sie sollten ankommende Anfragen in irgendeiner Weise behandeln, damit Sie nicht ständig auf Timeouts warten müssen.

Ob Sie `identd` nun nach außen hin anbieten oder nicht, es gibt eigentlich keinen guten Grund, warum Sie diesen Dienst nicht intern benutzen sollten. D.h. Sie sollten in der `/etc/inetd.conf` auf beiden Firewalls den auth-Eintrag aktivieren.

Einstellungen für das DMZ-Interface der Bastion

Tabelle 4.4 enthält die Protokolltabelle.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Bastion als Client einer Anfrage	TCP	DMZ_ADDRESSES	113	output	BASTION_DMZ_IPADDR	1024:65535	Egal
Antwort eines Servers aus der DMZ	TCP	DMZ_ADDRESSES	113	input	BASTION_DMZ_IPADDR	1024:65535	ACK
Ein Rechner aus der DMZ als Client einer Anfrage	TCP	DMZ_ADDRESSES	1024:65535	input	BASTION_DMZ_IPADDR	113	Egal
Antwort des Servers auf der Bastion	TCP	DMZ_ADDRESSES	1024:65535	output	BASTION_DMZ_IPADDR	113	ACK

Tab. 4.4: Das identd-Protokoll auf dem DMZ-Interface der Bastion

Anfragen mit der Bastion als Client

Ihre Maschine handelt als auth-Client, wenn Sie z.B. einen Mail- oder FTP-Server betreiben. Es gibt eigentlich keinen Grund, warum Sie das verbieten sollten.

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $DMZ_ADDRESSES 113 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $DMZ_ADDRESSES 113 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Anfragen mit der Bastion als Server

Wenn Sie in /etc/inetd.conf (oder über die Init-Skripten) den identd-Server aktiviert haben, erlauben die folgenden Regeln ankommende identd-Anfragen:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR 113 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $BASTION_DMZ_IPADDR 113 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Einstellungen für das DMZ-Interface der Choke
 Die Protokolltabelle finden Sie in Tabelle 4.5.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients aus der DMZ	TCP	DMZ_ADDRESSES	1024:65535	input	CHOKE_DMZ_IPADDR	113	Egal
Antwort des Servers auf der Choke	TCP	DMZ_ADDRESSES	1024:65535	output	CHOKE_DMZ_IPADDR	113	ACK
Anfrage eines Clients auf der Choke	TCP	ANYWHERE	113	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort eines fremden Servers	TCP	ANYWHERE	113	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.5: Das identd-Protokoll auf dem DMZ-Interface der Choke

Anfragen mit der Choke als Server

Wenn Sie in `/etc/inetd.conf` (oder über die Init-Skripten) den `identd`-Server aktiviert haben, erlauben die folgenden Regeln ankommende `identd`-Anfragen:

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 113 -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR 113 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Anfragen mit der Choke als Client

Ihre Maschine handelt als `auth`-Client, wenn Sie z.B. einen Mail- oder FTP-Server betreiben. Es gibt eigentlich keinen Grund, warum Sie das verbieten sollten.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 113 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 113 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.9 E-Mail (SMTP: TCP-Port 25; POP: TCP-Port 110; IMAP: TCP-Port 143)

Unabhängig davon, wie Sie Mail zwischen Ihrer Bastion und dem Internet austauschen, werden Sie vermutlich einen zentralen SMTP-Server für den internen Mailversand betreiben. Als Beispiel gehen wir in diesem Abschnitt davon aus, dass entweder die Bastion oder ein Computer in der DMZ als lokales Gateway und als Mail-Server fungiert. Die lokalen Clients holen ankommende Mail vom Server über POP oder IMAP ab.

Vier häufige Client-Server-Kombinationen werden beschrieben:

- Mailversand über ein SMTP-Relay auf der Bastion; Mailempfang über einen POP-Client auf der Bastion
- Mailversand über ein SMTP-Relay auf der Bastion; Mailempfang über einen IMAP-Client auf der Bastion
- Mailversand über ein SMTP-Relay in der DMZ; Mailempfang über einen POP-Client in der DMZ
- Mailversand über ein SMTP-Relay in der DMZ; Mailempfang über einen IMAP-Client in der DMZ

Diese Ansätze sind unabhängig davon, ob Ihr Mail-Relay die abgehende Mail jetzt direkt an den Empfänger ausliefert oder nur an einen SMTP-Server Ihres Internet-Providers weiterleitet. Bei den letzten beiden Methoden stehen Relay und lokaler Mailserver in der DMZ; auf der Bastion selbst muss daher weder ein POP- noch ein IMAP-Server aktiv sein.

Abgehende Mails über die Bastion verschicken (TCP-Port 25)

Wenn Sie abgehende E-Mail über ein fremdes Gateway versenden, schickt Ihr E-Mail-Client alle Mail an den Mailserver Ihres Internet-Providers. Der Provider stellt für Ihre E-Mail eine Brücke zum Rest der Welt dar. Ihr eigener Computer muss nicht wissen, wie er die Empfänger der E-Mails ausfindig macht und eine Verbindung dorthin herstellt. Darum kümmert sich der Server Ihres Providers.

Alternativ können Sie den Server Ihres Providers auch umgehen und einen eigenen Mailserver betreiben. Ihr lokaler Server wird dann die abgehende Mail annehmen, eine DNS-Anfrage nach dem Empfänger durchführen und die Mail an den entsprechenden Computer schicken. Ihr Mail-Client sendet in dieser Situation alles an Ihren eigenen SMTP-Server, nicht mehr an den Ihres Providers.

Wenn Sie abgehende Mail mit einem eigenen SMTP-Server versenden und ankommende Mail als SMTP-Client empfangen, sind Sie in Bezug auf E-Mail selbstständig. Ihr lokaler `sendmail`-Dämon sendet abgehende Mail direkt an die jeweiligen Empfänger und empfängt ankommende Mail.

Konfiguration der DMZ-Seite der Bastion für einen SMTP-Server

Tabelle 4.6 zeigt das Protokoll in tabellarischer Form.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Client sendet eine Mail an das Relay auf der Bastion	TCP	DMZ_ADDRESSES	1024:65535	input	BASTION_DMZ_IPADDR	25	Egal
Antwort des Servers auf der Bastion	TCP	DMZ_ADDRESSES	1024:65535	output	BASTION_DMZ_IPADDR	25	ACK

Tab. 4.6: Konfiguration der DMZ-Seite der Bastion für einen SMTP-Server

Die folgenden Regeln erlauben es lokalen Maschinen, Mail an einen SMTP-Server auf der Bastion zu senden:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR 25 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $BASTION_DMZ_IPADDR 25 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke für SMTP-Clients

Tabelle 4.7 enthält die Protokolltabelle.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Jemand verschickt eine Mail	TCP	BASTION_DMZ_IPADDR	25	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des SMTP-Servers auf der Bastion	TCP	BASTION_DMZ_IPADDR	25	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.7: Konfiguration der DMZ-Seite der Choke für SMTP-Clients

Die folgenden Regeln erlauben den lokalen Clients den Mailversand an die Choke:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR 25 -j ACCEPT
```

```
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \  
-s $BASTION_DMZ_IPADDR 25 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Mailempfang über einen POP-Server auf der Bastion

Es gibt viele Möglichkeiten, mit ankommender Mail umzugehen. In der Regel werden Sie einen POP- oder IMAP-Server betreiben. Die Benutzer im privaten LAN, hinter der Choke, benutzen entsprechend POP- bzw. IMAP-Clients, um auf ihre Postfächer zuzugreifen.

POP ist ein sehr häufig eingesetztes Protokoll, über das ein Endbenutzer seine Mail abholt. Das folgende Beispiel zeigt eine Architektur, bei der auf der Bastion ein POP-Server läuft.

POP-Server auf der Bastion: Konfiguration der Bastion

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	CHOKe_DMZ_ IPADDR	1024:65535	input	BASTION_ DMZ_ IPADDR	110	Egal
Antwort des POP- Servers auf der Bastion	TCP	CHOKe_DMZ_ IPADDR	1024:65535	output	BASTION_ DMZ_ IPADDR	110	ACK

Tab. 4.8: Konfiguration der DMZ-Seite der Bastion, sodass lokale POP-Clients Mail abholen können

Für die Arbeit im lokalen LAN muss der POP-Server auf der Bastion nur lokale Zugriffe erlauben. Externer Zugriff über das externe Interface wird nicht gestattet.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $BASTION_DMZ_IPADDR 110 -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $BASTION_DMZ_IPADDR 110 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

POP-Server auf der Bastion: Konfiguration der Choke

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	BASTION_DMZ_ IPADDR	110	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des POP-Servers auf der Bastion	TCP	BASTION_DMZ_ IPADDR	110	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.9: Konfiguration der DMZ-Seite der Choke, sodass lokale POP-Clients Mail vom POP-Server der Bastion abholen können

Clients holen E-Mail vom POP-Server auf der Bastion ab.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR 110 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $BASTION_DMZ_IPADDR 110 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Mailempfang über einen IMAP-Server auf der Bastion

IMAP ist ein anderes beliebtes Protokoll, über das Clients Mail abholen. Das folgende Beispiel zeigt eine Architektur, bei der auf der Bastion ein IMAP-Server läuft.

IMAP-Server auf der Bastion: Konfiguration der Bastion

Für die Arbeit im lokalen LAN muss der IMAP-Server auf der Bastion nur lokale Zugriffe erlauben. Externer Zugriff über das externe Interface wird nicht gestattet.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	BASTION_ DMZ_ IPADDR	143	Egal
Antwort des IMAP-Servers auf der Bastion	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	BASTION_ DMZ_ IPADDR	143	ACK

Tab. 4.10: Konfiguration der DMZ-Seite der Bastion, sodass lokale IMAP-Clients Mail abholen können

Das folgende Regelpaar erlaubt ankommende Verbindungen von der Choke:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $BASTION_DMZ_IPADDR 143 -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $BASTION_DMZ_IPADDR 143 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

IMAP-Server auf der Bastion: Konfiguration der Choke

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	BASTION_DMZ_ IPADDR	143	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Antwort des IMAP-Servers auf der Bastion	TCP	BASTION_DMZ_ IPADDR	143	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.11: Konfiguration der DMZ-Seite der Choke, sodass lokale IMAP-Clients Mail vom IMAP-Server der Bastion abholen können

Clients holen E-Mail vom IMAP-Server auf der Bastion ab.

```
ipchains -A output -i $CHOKe_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $BASTION_DMZ_IPADDR 143 -j ACCEPT  
  
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \  
-s $BASTION_DMZ_IPADDR 143 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Abgehende Mails über einen SMTP-Server in der DMZ verschicken

Wenn die lokalen Mailserver nicht auf der Bastion selbst laufen, sondern auf einem (anderen) Computer in der DMZ, hat das einige Sicherheitsvorteile. Falls jemand über eines der Server-Programme in den Computer einbricht, fällt nicht die Firewall, sondern nur der Mailserver.

sendmail als SMTP-Server übernimmt die Zustellung sowohl abgehender als auch ankommender E-Mails. Damit sorgt es für Ihre Unabhängigkeit in Bezug auf E-Mail.

Die folgenden Konstanten beschreiben die Maschine, die innerhalb der DMZ als Mailserver dient:

```
MAIL_DMZ_INTERFACE="eth0"
MAIL_SERVER_DMZ_IPADDR="192.168.1.4"
```

Konfiguration des SMTP-Servers in der DMZ

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Empfang einer ankommenden Mail	TCP	ANYWHERE	1024:65535	input	MAIL_SERVER_DMZ_IPADDR	25	Egal
Antwort des DMZ-Servers	TCP	ANYWHERE	1024:65535	output	MAIL_SERVER_DMZ_IPADDR	25	ACK
Senden einer abgehenden Mail	TCP	ANYWHERE	25	output	MAIL_SERVER_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden SMTP-Servers	TCP	ANYWHERE	25	input	MAIL_SERVER_DMZ_IPADDR	1024:65535	ACK

Tab. 4.12: SMTP-Protokoll aus der Sicht des Servers in der DMZ

Die ersten beiden Regeln für den Mailserver in der DMZ gehen davon aus, dass auf der Bastion entweder ein weiterer Mailserver läuft, der ankommende Mail aus dem Internet automatisch an den Mailserver in der DMZ weiterleitet, oder dass die Bastion ankommende Verbindungen direkt an den Server in der DMZ weitergibt. Sie erlauben den Empfang ankommender Mail.

```
ipchains -A input -i $MAIL_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $MAIL_SERVER_DMZ_IPADDR 25 -j ACCEPT

ipchains -A output -i $MAIL_DMZ_INTERFACE -p tcp ! -y \
-s $MAIL_SERVER_DMZ_IPADDR 25 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Die folgenden Regeln erlauben dem DMZ-Mailserver den Versand abgehender E-Mail an Empfänger im Internet:

```
ipchains -A output -i $MAIL_DMZ_INTERFACE -p tcp \
-s $MAIL_SERVER_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 25 -j ACCEPT
```



```
ipchains -A input -i $MAIL_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 25 \
-d $MAIL_SERVER_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Die hier gezeigten Regeln gelten, wie gesagt, auch dann, wenn auf der Bastion kein eigener SMTP-Server läuft. Stattdessen kann die Bastion ankommende Verbindungen über `ipmasqadm portfw` direkt an den DMZ-Server weiterleiten.

Konfiguration der Choke für Verbindungen zum Mailserver in der DMZ

Tabelle 4.13 zeigt, wie sich bei dieser Konfiguration der Versand von Mail für die Choke darstellt.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Client sendet eine abgehende Mail	TCP	MAIL_SERVER_DMZ_IPADDR	25	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des SMTP-Servers in der DMZ	TCP	MAIL_SERVER_DMZ_IPADDR	25	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.13: SMTP-Protokoll aus der Sicht der Choke, wenn ein lokaler Client Mail an den Server in der DMZ sendet

Lokale Maschinen senden Mails an den Server in der DMZ:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $MAIL_SERVER_DMZ_IPADDR 25 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $MAIL_SERVER_DMZ_IPADDR 25 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Bastion für die Weiterleitung ankommender E-Mail an den SMTP-Server in der DMZ

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Empfang einer ankommenden Mail	TCP	ANYWHERE	1024:65535	input	IPADDR	25	Egal
Antwort des Servers in der DMZ	TCP	ANYWHERE	1024:65535	output	IPADDR	25	ACK

Tab. 4.14: SMTP-Protokoll auf dem externen Interface der Bastion, wenn ankommende E-Mail an den SMTP-Server in der DMZ weitergeleitet wird

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Senden einer abge- henden Mail	TCP	ANYWHERE	25	output	IPADDR	1024:65535	Egal
Antwort des frem- den SMTP-Servers	TCP	ANYWHERE	25	input	IPADDR	1024:65535	ACK

Tab. 4.14: SMTP-Protokoll auf dem externen Interface der Bastion, wenn ankommende E-Mail an den SMTP-Server in der DMZ weitergeleitet wird (Forts.)

In diesem Beispiel beschreibe ich den Fall, dass auf der Bastion kein eigener SMTP-Server als Relay läuft. Stattdessen wird `ipmasqadm ipportfw` verwendet, um ankommende Verbindungen direkt an den Server in der DMZ weiterzuleiten.

Die ersten beiden Regelpaare sowie die `ipmasqadm`-Regel erlauben ankommende Verbindungen zum DMZ-Server:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 25 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 25 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $MAIL_SERVER_DMZ_IPADDR 25 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $MAIL_SERVER_DMZ_IPADDR 25 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipmasqadm portfw -a -P tcp -L $IPADDR 25 -R $MAIL_SERVER_DMZ_IPADDR 25
```

Die folgenden Regeln ermöglichen dem Mailserver in der DMZ die Kommunikation mit fremden Rechnern zum Zwecke des Mail-Versands:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $MAIL_SERVER_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 25 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 25 \
-d $MAIL_SERVER_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 25 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipmasqadm portfw -a -P tcp -L $MAIL_SERVER_DMZ_IPADDR 25 -R $IPADDR 25
```

Mailempfang über einen POP-Server auf einem Computer der DMZ

Oft lässt man die Endnutzer Ihre Mail von einem POP-Server abholen. Das folgende Beispiel zeigt die entsprechenden Firewall-Regeln, mit denen Clients im privaten LAN (hinter der Choke) Mail von einem POP-Server auf einem DMZ-Computer abholen können.

Der POP-Server läuft im Beispiel auf dem gleichen Rechner, der auch als SMTP-Server fungiert.

```
POP_DMZ_INTERFACE="eth0"  
POP_SERVER_DMZ_IPADDR="192.168.1.4"
```

Konfiguration des POP-Servers in der DMZ

Tabelle 4.15 zeigt das POP-Protokoll aus der Sicht des in der DMZ befindlichen POP-Servers.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	POP_ SERVER_ DMZ_ IPADDR	110	Egal
Antwort des loka- len Servers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	POP_ SERVER_ DMZ_ IPADDR	110	ACK

Tab. 4.15: Das POP-Protokoll aus der Sicht eines DMZ-Servers

Auf dem Mailserver in der DMZ läuft ein lokaler POP-Server für die Geräte im privaten LAN. Zugriffe aus dem Internet sind nicht erlaubt.

```
ipchains -A input -i $POP_DMZ_INTERFACE -p tcp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $POP_SERVER_DMZ_IPADDR 110 -j ACCEPT  
  
ipchains -A output -i $POP_DMZ_INTERFACE -p tcp ! -y \  
-s $POP_SERVER_DMZ_IPADDR 110 \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Choke für Zugriffe auf den POP-Server in der DMZ

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	POP_SERVER_ DMZ_IPADDR	110	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des POP- Servers in der DMZ	TCP	POP_SERVER_ DMZ_IPADDR	110	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.16: Konfiguration der DMZ-Seite der Choke, sodass lokale POP-Clients Mail vom POP-Server in der DMZ abholen können

Clients holen ihre Mail vom POP-Server in der DMZ ab.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $POP_SERVER_DMZ_IPADDR 110 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $POP_SERVER_DMZ_IPADDR 110 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Mailempfang über einen IMAP-Server auf einem Computer der DMZ

Oft lässt man die Endnutzer Ihre Mail von einem IMAP-Server abholen. Das folgende Beispiel zeigt die entsprechenden Firewall-Regeln, mit denen Clients im privaten LAN (hinter der Choke) Mail von einem IMAP-Server auf einem DMZ-Computer abholen können.

Der IMAP-Server läuft im Beispiel auf dem gleichen Rechner, der auch als SMTP-Server fungiert.

```
IMAP_DMZ_INTERFACE="eth0"
IMAP_SERVER_DMZ_IPADDR="192.168.1.4"
```

Konfiguration des IMAP-Servers in der DMZ

Tabelle 4.17 zeigt das IMAP-Protokoll aus der Sicht des in der DMZ befindlichen IMAP-Servers.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	IMAP_ SERVER_ DMZ_ IPADDR	143	Egal
Antwort des lokalen Servers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	IMAP_ SERVER_ DMZ_ IPADDR	143	ACK

Tab. 4.17: Das IMAP-Protokoll aus der Sicht eines DMZ-Servers

Auf dem Mailserver in der DMZ läuft ein lokaler IMAP-Server für die Geräte im privaten LAN. Zugriffe aus dem Internet sind nicht erlaubt.

```
ipchains -A input -i $IMAP_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $IMAP_SERVER_DMZ_IPADDR 143 -j ACCEPT

ipchains -A output -i $IMAP_DMZ_INTERFACE -p tcp ! -y \
-s $IMAP_SERVER_DMZ_IPADDR 143 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Choke für Zugriffe auf den IMAP-Server in der DMZ

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	IMAP_SERVER_ DMZ_IPADDR	143	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des IMAP-Servers in der DMZ	TCP	IMAP_SERVER_ DMZ_IPADDR	143	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.18: Konfiguration der DMZ-Seite der Choke, sodass lokale IMAP-Clients Mail vom IMAP-Server in der DMZ abholen können

Clients holen ihre Mail vom IMAP-Server in der DMZ ab.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $IMAP_SERVER_DMZ_IPADDR 143 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $IMAP_SERVER_DMZ_IPADDR 143 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.10 Usenet News (TCP: NNTP-Port 119)

Usenet (auch kurz »News« genannt) wird mit dem Protokoll NNTP übertragen, und zwar über TCP und den Port 119. Für das Lesen von News und das Posten eigener Artikel benutzen Sie einen lokalen News-Client.

Auf einem System, das nicht gerade einen Internet-Provider darstellt, wird es wohl kaum einen öffentlich zugänglichen Newsserver geben; selbst ein lokaler Newsserver wird nur selten gebraucht. Falls Sie eine Ausnahme sind, sollten Sie die Firewall so konfigurieren, dass nur handverlesene Clients zugreifen dürfen. Wenn Sie einen öffentlichen Newsserver betreiben, dann bitte nicht auf der Bastion. Normalerweise wird der Newsserver irgendwo in der DMZ stehen, und die Bastion leitet ankommende NNTP-Verbindungen an ihn weiter.

Konfiguration der Bastion als Newsserver bzw. für die Weiterleitung zu einem Newsserver in der DMZ

Tabelle 4.19 beschreibt tabellarisch das NNTP-Protokoll aus Sicht der Bastion. Beachten Sie bei dieser Tabelle besonders, dass – wie auch bei den anderen Protokolltabellen – alle Regeln aus der Sicht des DMZ-Interfaces der Bastion beschrieben sind. Z.B. ist in der dritten Zeile der NNTP-Client nicht wirklich eine lokale Adresse, und das Paket kommt zunächst erst einmal an. Aus der Perspektive des DMZ-seitigen Netzwerkinterfaces aber liegt der NNTP-Client tatsächlich »hinter uns«, also in Richtung lokal, und die Anfrage wird auf der *output*-Chain an den Computer in der DMZ weitergeleitet.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients aus der DMZ an einen fremden News-server	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	NEWS_SERVER	119	Egal
Antwort des fremden Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	NEWS_SERVER	119	ACK
Anfrage eines fremden Clients	TCP	NEWS_SERVER_DMZ_IPADDR	119	output	NNTP-Clients	1024:65535	Egal
Antwort des Servers in der DMZ	TCP	NEWS_SERVER_DMZ_IPADDR	119	input	NNTP-Clients	1024:65535	ACK
Anfrage des Servers in der DMZ	TCP	NEWS_SERVER_DMZ_IPADDR	1024:65535	input	NEWS_FEED	119	Egal
Antwort des fremden Servers	TCP	NEWS_SERVER_DMZ_IPADDR	1024:65535	output	NEWS_FEED	119	ACK

Tab. 4.19: NNTP-Protokoll aus Sicht des DMZ-Interfaces der Bastion, die Anfragen an einen Newsserver in der DMZ weiterleitet

Die Serverregeln erlauben zunächst Verbindungen zum Newsserver Ihres Internet-Providers. Damit sind sowohl das Lesen von News als auch das Posten neuer Artikel abgedeckt.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKO_DMZ_IPADDR $UNPRIVPORTS \  
-d $NEWS_SERVER 119 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $NEWS_SERVER 119 \  
-d $CHOKO_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Wenn auf einem Computer in der DMZ ein lokaler Newsserver läuft und für bestimmte Clients aus dem Internet auch öffentlich zugänglich ist, benötigen wir noch ein Regelpaar für diese Clients:

```
NEWS_SERVER_DMZ_IPADDR="192.168.1.6"
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \  
-s Ips.meiner.News.Clients $UNPRIVPORTS \  
-d $NEWS_SERVER_DMZ_IPADDR 119 -j ACCEPT
```

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $NEWS_SERVER_DMZ_IPADDR 119 \  
-d Ips.meiner.News.Clients $UNPRIVPORTS -j ACCEPT
```

Falls die DMZ-Geräte private IP-Adressen erhalten haben, muss zusätzlich Port-Forwarding aktiviert werden:

```
ipmasqadm portfw -a -P tcp -L Ips.meiner.News.Clients 119 \  
-R $NEWS_SERVER_DMZ_IPADDR 119
```

Sofern der lokale Newsserver neben den lokalen Diskussionsforen auch öffentliche Newsgruppen anbieten soll, benötigt er einen Newsfeed von einem anderen Server. Die folgenden Regeln erlauben den Zugriff auf einen solchen Newsfeed:

```
NEWS_FEED="IP.meines.News.Feeds"
```

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $NEWS_SERVER_DMZ_IPADDR $UNPRIVPORTS \  
-d $NEWS_FEED 119 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $NEWS_FEED 119 \  
-d $NEWS_SERVER_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Choke als NNTP-Client

Die Protokolltabelle für die Choke ist in Tabelle 4.20 abgedruckt.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	NEWS_SERVER	119	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden Newsservers	TCP	NEWS_SERVER	119	input	CHOKE_DMZ_IPADDR	1024:65535	ACK
Anfrage eines Clients hinter der Choke	TCP	NEWS_SERVER_DMZ_IPADDR	119	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des News-servers in der DMZ	TCP	NEWS_SERVER_DMZ_IPADDR	119	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.20: Das NNTP-Protokoll aus der Sicht der DMZ-Seite der Choke

Das erste Regelpaar erlaubt den Zugriff auf einen fremden Newsserver:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS\
-d $NEWS_SERVER 119 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $NEWS_SERVER 119 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS-j ACCEPT
```

Das zweite Paar lässt den Zugang zu unserem eigenen Newsserver in der DMZ zu:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS\
-d $NEWS_SERVER_DMZ_IPADDR 119 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $NEWS_SERVER_DMZ_IPADDR 119 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS-j ACCEPT
```

4.4.11 telnet (TCP-Port 23)

In Abhängigkeit davon, wieweit Sie Ihren eigenen Benutzern vertrauen, wird der lokale Einsatz von `telnet` möglicherweise akzeptabel sein. Je nachdem, welche Betriebssysteme Sie außer Unix noch einsetzen, mag `telnet` sogar die einzige Option für sein Einsatzgebiet sein.

Die hier gezeigten Regeln erlauben abgehende `telnet`-Verbindungen nach überall. Vermutlich können Sie die Liste der akzeptierten Server deutlich einschränken. Gegebenenfalls sollten Sie den Zugang auch in `/etc/hosts.allow` einschränken.

Konfiguration der DMZ-Seite der Bastion

Tabelle 4.21 zeigt das Client-Server-Protokoll für telnet.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	ANYWHERE	23	Egal
Antwort des Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	ANYWHERE	23	ACK
Anfrage eines Clients auf der Bastion	TCP	DMZ_ADDRESSES	23	output	BASTION_DMZ_IPADDR	1024:65535	Egal
Antwort des Servers aus der DMZ	TCP	DMZ_ADDRESSES	23	input	BASTION_DMZ_IPADDR	1024:65535	ACK

Tab. 4.21: Das telnet-Protokoll aus Sicht der Bastion

Beachten Sie, dass wir ankommende telnet-Verbindungen nur von der Choke-Firewall zulassen, nicht von überall. Andere Maschinen in der DMZ dürfen also keine telnet-Sitzungen aufbauen.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 23 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 23 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Mit den folgenden beiden Regeln können Sie von der Bastion aus jeden Computer in der DMZ per telnet administrieren:

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $DMZ_ADDRESSES 23 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $DMZ_ADDRESSES 23 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke

Die Protokolltabelle finden Sie in Tabelle 4.22.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	ANYWHERE	23	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	23	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK
Anfrage eines Clients auf der Bas- tion	TCP	BASTION_DMZ_ IPADDR	1024:65535	input	CHOKE_ DMZ_ IPADDR	23	Egal
Antwort des Ser- vers auf der Choke	TCP	BASTION_DMZ_ IPADDR	1024:65535	output	CHOKE_ DMZ_ IPADDR	23	ACK

Tab. 4.22: Das telnet-Protokoll aus Sicht der Choke

Alle abgehenden telnet-Verbindungen werden akzeptiert, damit Benutzer im LAN über telnet auf Computern im Internet arbeiten können.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 23 -j ACCEPT
```

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 23 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Die nächsten beiden Regeln erlauben der Bastion Verbindungen zur Choke. Andere Computer in der DMZ dürfen nicht mit telnet arbeiten.

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 23 -j ACCEPT
```

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR 23 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.12 ssh (TCP-Port 22)

Die hier vorgestellten Regeln erlauben uneingeschränkten Zugang. In der Praxis werden Sie die Liste der externen Adressen vermutlich auf wenige Sites begrenzen, zumal beide Verbindungspartner die Benutzeraccounts erkennen müssen. Der Start von sshd aus inetd wird nicht empfohlen; sshd lässt sich aber so kompilieren, dass es selber auf die Zugriffskontrolllisten in /etc/hosts.allow und /etc/hosts.deny achtet.

Konfiguration der DMZ-Seite der Bastion für Zugriffe auf ssh-Server

Tabelle 4.23 enthält die Protokolltabelle.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	22	Egal
Antwort des Ser- vers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	22	ACK
Anfrage eines Clients hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	513:1023	input	ANYWHERE	22	Egal
Antwort des Ser- vers	TCP	CHOKE_DMZ_ IPADDR	513:1023	output	ANYWHERE	22	ACK

Tab. 4.23: Das ssh-Protokoll aus der Sicht der Bastion als ssh-Server

Die folgenden Regeln erlauben Verbindungen von der Choke und den durch sie maskierten Clients zu ssh-Servern. Damit sind Server sowohl auf der Bastion als auch auf externen Sites abgedeckt.

```

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 22 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $SSH_PORTS \
-d $ANYWHERE 22 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $CHOKE_DMZ_IPADDR $SSH_PORTS -j ACCEPT

```

Konfiguration der DMZ-Seite der Bastion als ssh-Client

Tabelle 4.24 enthält die Protokolltabelle.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf der Bastion	TCP	DMZ_ADDRESSES	22	output	BASTION_DMZ_IPADDR	1024:65535	Egal
Antwort des Servers in der DMZ	TCP	DMZ_ADDRESSES	22	input	BASTION_DMZ_IPADDR	1024:65535	ACK
Anfrage eines Clients auf der Bastion	TCP	DMZ_ADDRESSES	22	output	BASTION_DMZ_IPADDR	513:1023	Egal
Antwort des Servers in der DMZ	TCP	DMZ_ADDRESSES	22	input	BASTION_DMZ_IPADDR	513:1023	ACK

Tab. 4.24: Das ssh-Protokoll aus der Sicht der Bastion als ssh-Client

Über die nächsten Regeln kann die Bastion auf alle Computer in der DMZ zugreifen:

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $DMZ_ADDRESSES 22 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $DMZ_ADDRESSES 22 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $SSH_PORTS \
-d $DMZ_ADDRESSES 22 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $DMZ_ADDRESSES 22 \
-d $BASTION_DMZ_IPADDR $SSH_PORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als ssh-Client

Tabelle 4.25 enthält die Protokolltabelle.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients hinter der Choke	TCP	ANYWHERE	22	output	CHOKE_DMZ_IPADDR	1024:65535	Egal

Tab. 4.25: Das ssh-Protokoll aus der Sicht der Choke als ssh-Client

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Antwort des Servers	TCP	ANYWHERE	22	input	CHOKE_DMZ_IPADDR	1024:65535	ACK
Anfrage eines Clients hinter der Choke	TCP	ANYWHERE	22	output	CHOKE_DMZ_IPADDR	513:1023	Egal
Antwort des Servers	TCP	ANYWHERE	22	input	CHOKE_DMZ_IPADDR	513:1023	ACK

Tab. 4.25: Das ssh-Protokoll aus der Sicht der Choke als ssh-Client (Forts.)

Die folgenden Regeln gelten für alle ssh-Verbindungen zu fremden Sites.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 22 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $SSH_PORTS \
-d $ANYWHERE 22 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $CHOKE_DMZ_IPADDR $SSH_PORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als ssh-Server

Tabelle 4.26 enthält die Protokolltabelle.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients auf der Bastion	TCP	BASTION_DMZ_IPADDR	1024:65535	input	CHOKE_DMZ_IPADDR	22	Egal
Antwort des Servers auf oder hinter der Choke	TCP	BASTION_DMZ_IPADDR	1024:65535	output	CHOKE_DMZ_IPADDR	22	ACK

Tab. 4.26: Das ssh-Protokoll aus der Sicht der Choke als ssh-Server

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf der Bas- tion	TCP	BASTION_DMZ_ IPADDR	513:1023	input	CHOKE_ DMZ_ IPADDR	22	Egal
Antwort des Ser- vers auf der Choke	TCP	BASTION_DMZ_ IPADDR	513:1023	output	CHOKE_ DMZ_ IPADDR	22	ACK

Tab. 4.26: Das ssh-Protokoll aus der Sicht der Choke als ssh-Server (Forts.)

Die letzte Regelgruppe für `ssh` schließlich erlaubt Verbindungen von der Bastion zum `sshd`-Server auf der Choke.

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $BASTION_FIREWALL $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 22 -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR 22 \
-d $BASTION_FIREWALL $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $BASTION_FIREWALL $SSH_PORTS \
-d $CHOKE_DMZ_IPADDR 22 -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR 22 \
-d $BASTION_FIREWALL $SSH_PORTS -j ACCEPT
```

4.4.13 ftp (TCP-Ports 21 und 22)

`ftp` kennt zwei Modi für den Datenaustausch zwischen Client und Server: den *normalen* oder *aktiven* sowie den *passiven* Modus. Der aktive Modus ist die Voreinstellung, wenn Sie das Programm `ftp` aufrufen. Der passive Modus wurde später entwickelt; er kommt z.B. bei der Verwendung eines Web-Browsers zum Einsatz. Einige wenige FTP-Sites unterstützen nur einen der beiden Modi.

Ich stelle an dieser Stelle drei verschiedene Ansätze für die Implementierung des `ftp`-Dienstes vor:

- Die Bastion dient als Gateway zu fremden `ftp`-Servern; gleichzeitig läuft auf ihr gegebenenfalls ein lokaler Server. Die Choke ist Client.
- Die Bastion ist Client, die Choke der Server.
- Ein separater `ftp`-Server existiert in der DMZ; Bastion und Choke sind Clients.

Bastion als Server und/oder Gateway, Choke als Client

Praktisch alle Sites wollen auf andere Systeme als ftp-Clients zugreifen. Der erste Abschnitt erlaubt lokalen Maschinen den Zugang zu ftp-Servern auf der Bastion und auf fremden Computern.

Ankommende Verbindungen von der Choke zum Server auf der Bastion

Tabelle 4.27 enthält die Protokolltabelle.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	21	Egal
Antwort des Servers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	21	ACK
Datenkanalaufbau vom Server, aktiver Modus	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	20	Egal
Antwort auf Datenkanalaufbau durch Client auf oder hinter der Choke, aktiver Modus	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	20	ACK
Datenkanalaufbau vom Client auf oder hinter der Choke zum Server, passiver Modus	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	1024:65535	Egal
Antwort auf Datenkanalaufbau durch Server, passiver Modus	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	1024:65535	ACK

Tab. 4.27: Das ftp-Protokoll für einen Server auf der Bastion

Die folgenden Regeln ermöglichen Verbindungen von ftp-Clients auf der Choke und damit auch von Computern auf dem privaten LAN hinter der Choke-Firewall.

```
# Anfrage eines FTP-Clients
# -----

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 21 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 21 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus
# -----

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $ANYWHERE 20 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus
# -----

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Abgehende Verbindungen von der Choke zu einem Server

Tabelle 4.28 zeigt die am ftp-Protokoll beteiligten Pakete aus Sicht der Choke, wenn ein Client auf ihr oder auf einem Computer im privaten LAN eine ftp-Sitzung startet.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	ANYWHERE	21	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des Ser- vers	TCP	ANYWHERE	21	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK
Datenkanalaufbau durch Server, akti- ver Modus	TCP	ANYWHERE	20	input	CHOKE_ DMZ_ IPADDR	1024:65535	Egal

Tab. 4.28: Das ftp-Protokoll für die Choke als Client

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Antwort auf Datenkanalaufbau durch Client, aktiver Modus	TCP	ANYWHERE	20	output	CHOKE_DMZ_IPADDR	1024:65535	ACK
Datenkanalaufbau durch Client, passiver Modus	TCP	ANYWHERE	1024:65535	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort auf Datenkanalaufbau durch Server, passiver Modus	TCP	ANYWHERE	1024:65535	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.28: Das ftp-Protokoll für die Choke als Client (Forts.)

Die folgenden Regeln ermöglichen Verbindungen von ftp-Clients auf der Choke und damit auch von Computern auf dem privaten LAN hinter der Choke-Firewall.

```
# Anfrage eines FTP-Clients
# -----

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 21 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 21 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus
# -----

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $ANYWHERE 20 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus
# -----

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Bastion als Client, Choke als interner ftp-Server

Oft ist es einfach viel zu bequem, Dateien zwischen der Bastion und den internen Maschinen per ftp zu übertragen. Der folgende Abschnitt ist insofern ein Spezialfall: Die Bastion darf als ftp-Client auf die Choke zugreifen. In einer streng abgesicherten Umgebung sollte man an sich *keinen* Client-Zugang von der Bastion erlauben!

Abgehende Verbindungen von einem Client auf der Bastion zum Server auf der Choke

Tabelle 4.29 enthält die entsprechende Protokolltabelle:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf der Bastion	TCP	CHOKE_DMZ_IPADDR	21	output	BASTION_DMZ_IPADDR	1024:65535	Egal
Antwort des Servers auf der Choke	TCP	CHOKE_DMZ_IPADDR	21	input	BASTION_DMZ_IPADDR	1024:65535	ACK
Datenkanalaufbau durch Choke-Server, aktiver Modus	TCP	CHOKE_DMZ_IPADDR	20	input	BASTION_DMZ_IPADDR	1024:65535	Egal
Bastion-Client antwortet auf Datenkanalaufbau, aktiver Modus	TCP	CHOKE_DMZ_IPADDR	20	output	BASTION_DMZ_IPADDR	1024:65535	ACK
Datenkanalaufbau durch Bastion-Client, passiver Modus	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	BASTION_DMZ_IPADDR	1024:65535	Egal
Choke-Server antwortet auf Datenkanalaufbau, passiver Modus	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	BASTION_DMZ_IPADDR	1024:65535	ACK

Tab. 4.29: Das ftp-Protokoll mit der Bastion als Client

Die folgenden Regeln erlauben abgehende ftp-Verbindungen von einem Client auf der Bastion zur Choke:

```
# Anfrage des FTP-Clients
# -----

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKEDMZ_IPADDR 21 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKEDMZ_IPADDR 21 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus
# -----

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKEDMZ_IPADDR 20 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKEDMZ_IPADDR 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus
# -----

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKEDMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKEDMZ_IPADDR $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Ankommende Verbindungen von einem Client auf der Bastion zum Server auf der Choke

Tabelle 4.30 enthält die Protokolltabelle:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage von einem Client auf der Bastion	TCP	BASTION_DMZ_IPADDR	1024:65535	input	CHOKEDMZ_IPADDR	21	Egal
Antwort des Servers auf der Choke	TCP	BASTION_DMZ_IPADDR	1024:65535	output	CHOKEDMZ_IPADDR	21	ACK

Tab. 4.30: Das ftp-Protokoll aus der Sicht der Choke als Server

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Datenkanalaufbau durch Choke-Server, aktiver Modus	TCP	BASTION_DMZ_ IPADDR	1024:65535	output	CHOKE_ DMZ_ IPADDR	20	Egal
Bastion-Client antwortet auf Datenkanalaufbau, aktiver Modus	TCP	BASTION_DMZ_ IPADDR	1024:65535	input	CHOKE_ DMZ_ IPADDR	20	ACK
Datenkanalaufbau durch Bastion-Client, passiver Modus	TCP	BASTION_DMZ_ IPADDR	1024:65535	input	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Choke-Server antwortet auf Datenkanalaufbau, passiver Modus	TCP	BASTION_DMZ_ IPADDR	1024:65535	output	CHOKE_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.30: Das ftp-Protokoll aus der Sicht der Choke als Server (Forts.)

Die folgenden Regeln erlauben ankommende ftp-Verbindungen von einem Client auf der Bastion zur Choke:

```
# Anfrage eines FTP-Clients
# -----

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 21 -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR 21 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus
# -----

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR 20 \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus
# -----
```

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $BASTION_DMZ_IPADDR $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Separater ftp-Server in der DMZ; Bastion und Choke sind Clients

Der folgende Abschnitt ähnelt in gewisser Weise dem vorhergehenden; auch hier darf die Bastion als Client auf den ftp-Server zugreifen. Diesmal ist der Server allerdings auf einer separaten Maschine in der DMZ installiert.

Der ftp-Server in der DMZ

Tabelle 4.31 beschreibt das ftp-Protokoll vom Standpunkt des Servers in der DMZ aus.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients	TCP	DMZ_ADDRESSES	1024:65535	input	FTP_ SERVER_ DMZ_ IPADDR	21	Egal
Antwort des Ser- vers in der DMZ	TCP	DMZ_ADDRESSES	1024:65535	output	FTP_ SERVER_ DMZ_ IPADDR	21	ACK
Datenkanalaufbau durch DMZ-Ser- ver, aktiver Modus	TCP	DMZ_ADDRESSES	1024:65535	output	FTP_ SERVER_ DMZ_ IPADDR	20	Egal
Client beantwortet Datenkanalaufbau, aktiver Modus	TCP	DMZ_ADDRESSES	1024:65535	input	FTP_ SERVER_ DMZ_ IPADDR	20	ACK
Datenkanalaufbau durch Client, passi- ver Modus	TCP	DMZ_ADDRESSES	1024:65535	input	FTP_ SERVER_ DMZ_ IPADDR	1024:65535	Egal
DMZ-Server beant- wortet Datenkanal- aufbau, passiver Modus	TCP	DMZ_ADDRESSES	1024:65535	output	FTP_ SERVER_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.31: Das ftp-Protokoll für den Server in der DMZ

Die folgenden Regeln erlauben ankommende ftp-Verbindungen, die von jedem Gerät innerhalb der DMZ ausgehen dürfen, also auch von Choke und Bastion.

```
# Anfrage eines FTP-Clients
# -----

ipchains -A input -i $DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $FTP_SERVER_DMZ_IPADDR 21 -j ACCEPT

ipchains -A output -i $DMZ_INTERFACE -p tcp ! -y \
-s $FTP_SERVER_DMZ_IPADDR 21 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus
# -----

ipchains -A output -i $DMZ_INTERFACE -p tcp \
-s $FTP_SERVER_DMZ_IPADDR 20 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $DMZ_INTERFACE -p tcp ! -y \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $FTP_SERVER_DMZ_IPADDR 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus
# -----

ipchains -A input -i $DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $FTP_SERVER_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $DMZ_INTERFACE -p tcp ! -y \
-s $FTP_SERVER_DMZ_IPADDR $UNPRIVPORTS \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Die Choke als Client eines ftp-Servers in der DMZ

Tabelle 4.32 enthält die modifizierte Protokolltabelle für die Choke als ftp-Client, die diesmal auf den ftp-Server in der DMZ zugreift.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients von oder hinter der Choke	TCP	FTP_SERVER_DMZ_IPADDR	21	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des DMZ-Servers	TCP	FTP_SERVER_DMZ_IPADDR	21	input	CHOKE_DMZ_IPADDR	1024:65535	ACK
Datenkanalaufbau durch den DMZ-Server, aktiver Modus	TCP	FTP_SERVER_DMZ_IPADDR	20	input	CHOKE_DMZ_IPADDR	1024:65535	Egal
Choke-Client beantwortet Datenkanalaufbau, aktiver Modus	TCP	FTP_SERVER_DMZ_IPADDR	20	output	CHOKE_DMZ_IPADDR	1024:65535	ACK
Datenkanalaufbau durch einen Client auf oder hinter der Choke, passiver Modus	TCP	FTP_SERVER_DMZ_IPADDR	1024:65535	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
DMZ-Server beantwortet Datenkanalaufbau, passiver Modus	TCP	FTP_SERVER_DMZ_IPADDR	1024:65535	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.32: Das ftp-Protokoll aus der Sicht der Choke als Client eines DMZ-Servers

Die folgenden Regeln erlauben den Zugriff von der Choke und dem dahinter liegenden privaten LAN auf den ftp-Server in der DMZ:

```
# Anfrage eines FTP-Clients
# -----

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $FTP_SERVER_DMZ_IPADDR 21 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $FTP_SERVER_DMZ_IPADDR 21 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus
# -----

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $FTP_SERVER_DMZ_IPADDR 20 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $FTP_SERVER_DMZ_IPADDR 20 -j ACCEPT  
  
# Datenkanalaufbau im passiven Modus  
# -----  
  
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $FTP_SERVER_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \  
-s $FTP_SERVER_DMZ_IPADDR $UNPRIVPORTS \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.14 WWW

WWW-Dienste basieren auf dem HTTP-Protokoll. Für besondere Zwecke gibt es leicht modifizierte Versionen, z.B. für die Verwendung von Web-Proxies oder, als HTTPS, für abgesicherte Datenübertragung (über SSL).

Hier beschreibe ich drei Ansätze für die Organisation der Web-Services in einem lokalen Netz:

- Die Bastion arbeitet als Webserver oder leitet Anfragen als Gateway an fremde Server weiter; die Choke ist Client.
- Die Bastion ist Server bzw. Gateway; in der DMZ existiert ein separater Webserver; die Choke ist Client.
- Die Bastion ist Server bzw. Gateway; die Choke dient als lokaler Web-Proxy.

Bastion als Webserver oder Gateway, Choke als Client

Es ist kaum vorstellbar, dass eine private Site in der heutigen Welt ohne WWW-Zugriff auskommen wollte. Der erste Abschnitt erlaubt internen Hosts den Zugriff auf Webserver – sowohl auf die Bastion als auch auf fremde Hosts.

Ankommende Verbindungen von internen Clients zum Server auf der Bastion

Tabelle 4.33 enthält die Protokolltabelle für einen HTTP-Server.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	80	Egal
Antwort des Ser- vers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	80	ACK

Tab. 4.33: Das HTTP-Protokoll für den Server auf der Bastion

Die folgenden beiden Regeln erlauben ankommende HTTP-Verbindungen von Clients auf der Choke und damit auch von Maschinen im privaten LAN hinter der Choke-Firewall:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 80 -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 80 \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.34 enthält die serverseitige Protokolltabelle für SSL:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	443	Egal
Antwort des Ser- vers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	443	ACK

Tab. 4.34: Das SSL-Protokoll für den Server auf der Bastion

Die nächsten Regeln erlauben ankommende SSL-Verbindungen von der Choke und den Clients aus dem privaten LAN dahinter:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 443 -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 443 \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.35 zeigt die serverseitige Protokolltabelle für Web-Proxies:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	WEB_ PROXY_ SERVER	WEB_PROXY_ PORT	Egal
Antwort des Servers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	WEB_ PROXY_ SERVER	WEB_PROXY_ PORT	ACK

Tab. 4.35: Das Web-Proxy-Protokoll für den Server auf der Bastion

Das folgende Regelpaar erlaubt ankommende Verbindungen zum Web-Proxy, sowohl von der Choke als auch von den Computern im privaten Netz dahinter.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $WEB_PROXY_SERVER $WEB_PROXY_PORT -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_PROXY_SERVER $WEB_PROXY_PORT \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Abgehende Verbindungen von der Choke zu externen Servern
Tabelle 4.36 zeigt das Client-Protokoll für HTTP.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	ANYWHERE	80	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	80	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.36: Das HTTP-Protokoll aus der Sicht der Choke als Client

Dieser Abschnitt erlaubt den lokalen Rechnern Zugriffe sowohl auf die Bastion als auch auf fremde Webserver:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 80 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 80 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.37 zeigt das Client-Protokoll für SSL.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	ANYWHERE	443	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	443	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.37: Das SSL-Protokoll aus der Sicht der Choke als Client

Lokale Computer dürfen auf abgesicherte Webserver auf der Bastion und auf fremden Sites zugreifen:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 443 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 443 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.38 zeigt das Client-Protokoll für den Zugriff auf Web-Proxies.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	WEB_PROXY_SERVER	WEB_PROXY_PORT	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des Proxy-Servers	TCP	WEB_PROXY_SERVER	WEB_PROXY_PORT	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.38: Das Web-Proxy-Protokoll aus der Sicht der Choke als Client

Lokale Computer dürfen auf den entsprechend definierten Web-Proxy zugreifen:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $WEB_PROXY_SERVER $WEB_PROXY_PORT -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_PROXY_SERVER $WEB_PROXY_PORT \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Bastion als Gateway, öffentlicher Webserver in der DMZ, Choke als Client

Die Bastion leitet Anfragen in beide Richtungen weiter. Zusätzlich steht ein Web-Server in der DMZ. Die Choke fungiert als reiner Client.

Weiterleitung von Anfragen an Server in DMZ und Internet durch die Bastion

Tabelle 4.39 enthält das vollständige Client/Server-Protokoll für HTTP.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients aus der DMZ	TCP	DMZ_ADDRESSES	1024:65535	input	ANYWHERE	80	Egal
Antwort des fremden Servers	TCP	DMZ_ADDRESSES	1024:65535	output	ANYWHERE	80	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	output	WEB_SERVER_DMZ_IPADDR	80	Egal
Antwort des Servers in der DMZ	TCP	ANYWHERE	1024:65535	input	WEB_SERVER_DMZ_IPADDR	80	ACK

Tab. 4.39: Das Client/Server-Protokoll für HTTP aus Sicht der Bastion

Eigene Computer dürfen auf Webserver sowohl auf der Bastion als auch im Internet zugreifen. Gleichzeitig können fremde Clients den Webserver in der DMZ benutzen:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $DMZ_ADDRESSES $UNPRIVPORTS \
-d $ANYWHERE 80 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 80 \
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $WEB_SERVER_DMZ_IPADDR 80 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_SERVER_DMZ_IPADDR 80 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.40 enthält die entsprechende Protokolltabelle für SSL.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients aus der DMZ	TCP	DMZ_ADDRESSES	1024:65535	input	ANYWHERE	443	Egal
Antwort des frem- den Servers	TCP	DMZ_ADDRESSES	1024:65535	output	ANYWHERE	443	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	output	WEB_ SERVER_ DMZ_ IPADDR	443	Egal
Antwort des Ser- vers in der DMZ	TCP	ANYWHERE	1024:65535	input	WEB_ SERVER_ DMZ_ IPADDR	443	ACK

Tab. 4.40: Das Client/Server-Protokoll für SSL aus Sicht der Bastion

Eigene Computer dürfen auf abgesicherte Webserver sowohl der Bastion als auch im Internet zugreifen. Gleichzeitig können fremde Clients den abgesicherten Webserver in der DMZ benutzen:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $DMZ_ADDRESSES $UNPRIVPORTS \  
-d $ANYWHERE 443 -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 443 \  
-d $DMZ_ADDRESSES $UNPRIVPORTS -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $WEB_SERVER_DMZ_IPADDR 443 -j ACCEPT  
  
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $WEB_SERVER_DMZ_IPADDR 443 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.41 schließlich enthält die zugehörige Protokolltabelle für Zugriffe auf Web-Proxies.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	WEB_PROXY_SERVER	WEB_PROXY_PORT	Egal
Antwort des Proxy-Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	WEB_PROXY_SERVER	WEB_PROXY_PORT	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	output	WEB_SERVER_DMZ_IPADDR	WEB_PROXY_PORT	Egal
Antwort des Proxy-Servers in der DMZ	TCP	ANYWHERE	1024:65535	input	WEB_SERVER_DMZ_IPADDR	WEB_PROXY_PORT	ACK

Tab. 4.41: Das Client/Server-Protokoll für Web-Proxies aus Sicht der Bastion

Die Choke sowie die Computer im durch sie abgesicherten privaten Netz dürfen auf den angegebenen Proxy-Server zugreifen. Ebenfalls aufgeführt sind hier die Regeln für den Zugriff eines fremden Clients auf einen Proxy unseres Web-Servers in der DMZ.

```

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $WEB_PROXY_SERVER $WEB_PROXY_PORT -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_PROXY_SERVER $WEB_PROXY_PORT \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $WEB_SERVER_DMZ_IPADDR $WEB_PROXY_PORT -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_SERVER_DMZ_IPADDR $WEB_PROXY_PORT \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

```

Der öffentliche Webserver in der DMZ

Bei Red Hat werden zwei Proxy-Pakete mitgeliefert. Das eine ist das Proxy-Modul des Web-Servers Apache, das andere ist ein eigenständiges Paket namens Squid. Den Port, auf dem der Proxy arbeitet, legen Sie bei beiden Programmen frei fest. (Bei Squid ist die Voreinstellung der Port 3130.)

In den neueren Red Hat-Versionen sind sowohl das Proxy-Modul von Apache als auch Squid enthalten. Früher musste man Apache neu kompilieren, um das Proxy-Modul einzuschalten. Squid war zwar frei verfügbar, aber nicht Teil der normalen Red Hat-Distribution.

Tabelle 4.42 enthält die Protokolltabelle für HTTP aus der Sicht eines Servers in der DMZ.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	input	WEB_SERVER_DMZ_IPADDR	80	Egal
Antwort des DMZ-Servers	TCP	ANYWHERE	1024:65535	output	WEB_SERVER_DMZ_IPADDR	80	ACK

Tab. 4.42: Das HTTP-Protokoll aus Sicht eines DMZ-Servers

Ankommende HTTP-Verbindungen sind von überall her erlaubt, einschließlich Computern im Internet, der Bastion und der Choke:

```
WEB_DMZ_INTERFACE="etho"

ipchains -A input -i $WEB_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $WEB_SERVER_DMZ_IPADDR 80 -j ACCEPT

ipchains -A output -i $WEB_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_SERVER_DMZ_IPADDR 80 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.43 zeigt die Server-Protokolltabelle für SSL.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	input	WEB_SERVER_DMZ_IPADDR	443	Egal
Antwort des DMZ-Servers	TCP	ANYWHERE	1024:65535	output	WEB_SERVER_DMZ_IPADDR	443	ACK

Tab. 4.43: Das SSL-Protokoll aus Sicht eines DMZ-Servers

Abgesicherte WWW-Anfragen über SSL dürfen ebenfalls von jedermann gestellt werden.

```
WEB_DMZ_INTERFACE="eth0"

ipchains -A input -i $WEB_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $WEB_SERVER_DMZ_IPADDR 443 -j ACCEPT

ipchains -A output -i $WEB_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_SERVER_DMZ_IPADDR 443 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.44 zeigt die Server-Protokolltabelle für einen Proxy:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients	TCP	Proxy-Clients	1024:65535	input	WEB_ SERVER_ DMZ_ IPADDR	WEB_PROXY_ PORT	Egal
Antwort des DMZ- Servers	TCP	Proxy-Clients	1024:65535	output	WEB_ SERVER_ DMZ_ IPADDR	WEB_PROXY_ PORT	ACK

Tab. 4.44: Das Protokoll für Web-Proxies aus Sicht eines DMZ-Servers

Ein kleines System wird mit hoher Wahrscheinlichkeit dem Internet keinen Zugang zum eigenen Web-Proxy gestatten. Normalerweise wird ein lokaler Proxy einen lokalen Cache bereitstellen, und möglicherweise auch als Proxy für abgehende Verbindungen eingesetzt werden. Besondere Firewall-Regeln werden in diesem Fall nicht gebraucht. Den anderen Computern im Internet gegenüber benimmt sich der Proxy-Server wie ein ganz normaler Web-Client.

Falls Fremde doch Zugang zu Ihrem Proxy benötigen, verwenden Sie Regeln wie die folgenden:

```
WEB_DMZ_INTERFACE="eth0"

ipchains -A input -i $WEB_DMZ_INTERFACE -p tcp \
-s Ips.meiner.Proxy.Clients $UNPRIVPORTS \
-d $WEB_SERVER_DMZ_IPADDR $WEB_PROXY_PORT -j ACCEPT

ipchains -A output -i $WEB_DMZ_INTERFACE -p tcp ! -y \
-s $WEB_SERVER_DMZ_IPADDR $WEB_PROXY_PORT \
-d Ips.meiner.Proxy.Clients $UNPRIVPORTS -j ACCEPT
```


Die Choke als Client von Web-Servern in DMZ und Internet

Tabelle 4.45 enthält die clientseitige Protokolltabelle für HTTP.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	ANYWHERE	80	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	80	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.45: Das HTTP-Protokoll aus Sicht der Choke als Client

Das folgende Regelpaar erlaubt abgehende HTTP-Verbindungen von der Choke zu beliebigen Web-Servern:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 80 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 80 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.46 enthält die clientseitige Protokolltabelle für SSL.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	ANYWHERE	443	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	443	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.46: Das SSL-Protokoll aus Sicht der Choke als Client

Das folgende Regelpaar erlaubt abgehende HTTP-Verbindungen von der Choke zu abgesicherten Web-Servern in der ganzen Welt:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 443 -j ACCEPT
```

```
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 443 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Tabelle 4.47 enthält die clientseitige Protokolltabelle für den Zugriff auf Web-Proxies.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients auf oder hinter der Choke	TCP	WEB_PROXY_SER- VER	WEB_PROXY_ PORT	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Antwort des Proxy- Servers	TCP	WEB_PROXY_SER- VER	WEB_PROXY_ PORT	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.47: Das Protokoll für den Zugriff auf Web-Proxies aus Sicht der Choke als Client

Das folgende Regelpaar erlaubt abgehende HTTP-Verbindungen von der Choke zum angegebenen Proxy-Server:

```
ipchains -A output -i $CHOKe_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $WEB_PROXY_SERVER $WEB_PROXY_PORT -j ACCEPT  
  
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \  
-s $WEB_PROXY_SERVER $WEB_PROXY_PORT \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Bastion als Webserver oder Gateway, Choke als lokaler Web-Proxy

Obwohl man natürlich alle Web-Services von einem zentralen internen Server aus anbieten könnte, tut man das nur selten. So eine Konfiguration birgt nämlich ein erhöhtes Risiko für Sicherheitslücken durch fehlkonfigurierte Server und CGI-Skripten. Außerdem isoliert man sehr gerne private von öffentlich zugänglicher Information. D.h. dass Sites, die sowohl einen privaten internen als auch einen öffentlichen Webserver benötigen, normalerweise mehrere Webserver auf unterschiedlichen Computern in getrennten LANs betreiben. Z.B. könnte man den Server für die öffentliche Website auf der Bastion-Firewall oder auf einem Computer im Perimeter-Netz platzieren.

Für eine kleine private Site oder die Site einer kleinen Firma besteht die Möglichkeit, den öffentlichen Server auf der Bastion und den privaten Proxy-Server auf einer internen Maschine, in diesem Fall der Choke, unterzubringen. Dabei muss der öffentlich zugängliche Server nicht unbedingt auch SSL anbieten. Der interne Webserver dient als Proxy und ist von der Bastion aus nicht erreichbar.

Bei diesem Szenario benötigen Sie keine zusätzlichen Regeln. Aus der Perspektive der Bastion erscheint der interne Web-Proxy wie ein ganz normaler Web-Client. Der einzige Unterschied ist der, dass die zuvor angegebenen Proxy-Regeln nicht mehr erforderlich sind, außer Ihr Internet-Provider zwingt Sie dazu, seinen Proxy zu verwenden.

An den DMZ-Regeln von Bastion und Choke sind keine Änderungen nötig.

4.4.15 finger (TCP-Port 79)

Abgehende Verbindungen zu fremden `finger`-Servern können nicht schaden. Interner Zugang zu einem `finger`-Server auf der Bastion ist vermutlich nicht besonders sinnvoll. Davon, dass Sie fremde Clients aus dem Internet auf einen internen `finger`-Server zugreifen lassen, würde ich abraten. Ohnehin müssten Sie in einem durch Masquerading geschützten System dafür zusätzlichen Aufwand betreiben.

Weiterleitung von `finger`-Anfragen durch die Bastion

Tabelle 4.48 enthält das Client/Server-Protokoll für `finger`.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	ANYWHERE	79	Egal
Antwort des fremden Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	ANYWHERE	79	ACK
Anfrage eines fremden Clients	TCP	CHOKE_DMZ_IPADDR	79	output	Zugelassene finger-Clients	1024:65535	Egal
Antwort des Choke-Servers	TCP	CHOKE_DMZ_IPADDR	79	input	Zugelassene finger-Clients	1024:65535	ACK

Tab. 4.48: Weiterleitung von `finger`-Anfragen durch die Bastion

Anfragen an externe `finger`-Server sind unschädlich. Die folgenden Regeln erlauben sie:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 79 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 79 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Die Server-Regeln sind für eine private Site nicht besonders sinnvoll. Ein kleiner Internet-Provider mit Benutzeraccounts auf seinen Rechnern würde vermutlich einen finger-Server betreiben:

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 79 -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKE_DMZ_IPADDR 79 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Client/Server-Konfiguration für finger auf der Choke

Tabelle 4.49 enthält die Client/Server-Protokolltabelle für finger auf der Choke:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	79	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	79	input	CHOKE_DMZ_IPADDR	1024:65535	ACK
Anfrage eines fremden Clients	TCP	ANYWHERE	1024:65535	input	CHOKE_DMZ_IPADDR	79	Egal
Antwort des Servers auf der Choke	TCP	ANYWHERE	1024:65535	output	CHOKE_DMZ_IPADDR	79	ACK

Tab. 4.49: Das finger-Protokoll für die Choke

Die folgenden beiden Regeln erlauben abgehende finger-Anfragen von der Choke:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 79 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 79 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Die nächsten Regeln erlauben ankommende finger-Anfragen an die Choke:

```
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKe_DMZ_IPADDR 79 -j ACCEPT

ipchains -A output -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \
-s $CHOKe_DMZ_IPADDR 79 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

4.4.16 whois (TCP-Port 43)

whois greift auf die Registrierungsdatenbanken von Network Solutions, DENIC, RIPE u.ä. zu und liefert technische und administrative Informationen zu IP-Adressen, Host- und Domainnamen in für Menschen lesbarer Form.

Weiterleitung von whois-Anfragen durch die Bastion

Tabelle 4.50 zeigt die Client-Protokolltabelle für whois.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKe_DMZ_ IPADDR	1024:65535	input	ANYWHERE	43	Egal
Antwort des frem- den Servers	TCP	CHOKe_DMZ_ IPADDR	1024:65535	output	ANYWHERE	43	ACK

Tab. 4.50: Das whois-Client-Protokoll für die Bastion

Die folgenden Regeln erlauben Anfragen von whois-Clients auf der Choke und von Computern im privaten LAN hinter der Choke-Firewall:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKe_FIREWALL $UNPRIVPORTS \
-d $ANYWHERE 43 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 43 \
-d $CHOKe_FIREWALL $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Choke als whois-Client

Tabelle 4.51 zeigt die Client-Protokolltabelle für whois.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	43	output	CHOKE_DMZ_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	43	input	CHOKE_DMZ_IPADDR	1024:65535	ACK

Tab. 4.51: Das whois-Protokoll für die Choke

Die folgenden Regeln erlauben Anfragen von `whois`-Clients auf der Choke sowie von Computern im privaten LAN hinter der Choke-Firewall:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 43 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 43 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.17 gopher (TCP-Port 70)

`gopher` existiert zwar nach wie vor, ist aber mittlerweile sehr weitgehend von Web-basierten Suchmaschinen abgelöst worden. Netscape enthält einen eingebauten `gopher`-Client.

Weiterleitung von `gopher`-Anfragen durch die Bastion

Tabelle 4.52 zeigt die Client-Protokolltabelle für `gopher`.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	ANYWHERE	70	Egal
Antwort des fremden Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	ANYWHERE	70	ACK

Tab. 4.52: Das `gopher`-Client-Protokoll für die Bastion

Die folgenden Regeln erlauben Anfragen von `gopher`-Clients auf der Choke und auch von Computern im privaten LAN hinter der Choke-Firewall:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 70 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 70 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Choke als gopher-Client

Tabelle 4.53 zeigt die Client-Protokolltabelle für gopher.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage durch einen Client von oder hinter der Choke	TCP	ANYWHERE	70	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Antwort vom frem- den Server	TCP	ANYWHERE	70	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.53: Das gopher-Protokoll für die Choke

Die nächsten beiden Regeln erlauben abgehende gopher-Anfragen von Clients auf der Choke:

```
ipchains -A output -i $CHOKe_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 70 -j ACCEPT  
  
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 70 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.18 WAIS (TCP-Port 210)

Die *Wide Area Information Server* (WAIS) kennt man heute nur noch als Suchmaschinen. Die meisten Webbrowser, so auch Netscape, enthalten eine grafische Benutzerschnittstelle sowie die nötige Client-Software für den Zugriff auf WAIS-Server.

Weiterleitung von WAIS-Anfragen durch die Bastion

Tabelle 4.54 zeigt die Client-Protokolltabelle für WAIS.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Cli- ents von oder hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	210	Egal
Antwort des frem- den Servers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	210	ACK

Tab. 4.54: Das WAIS-Client-Protokoll für die Bastion

Die folgenden beiden Regeln ermöglichen internen Clients den Zugriff auf fremde WAIS-Server:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 210 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 210 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der Choke als WAIS-Client

Tabelle 4.55 zeigt die Client-Protokolltabelle für WAIS.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	210	output	CHOKE_ DMZ_ IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	ANYWHERE	210	input	CHOKE_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.55: Das WAIS-Client-Protokoll für die Choke

Die folgenden beiden Regeln gelten für den Client-Zugriff auf fremde WAIS-Ser-
ver:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 210 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 210 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```


4.4.19 RealAudio und QuickTime (Ports 554 u.a.)

RealAudio und QuickTime benutzen beide das *Real Time Streaming Protocol* (RTSP) über den TCP-Port 554 sowie das *Real Time Transport Protocol* (RTP) über ein Paar unprivilegierter TCP- oder UDP-Ports. RTSP stellt die Kontrollverbindung dar, der eigentliche Datentransfer erfolgt über RTP.

Welche Ports als unprivilegiertes Paar benutzt werden, hängt von der jeweils eingesetzten Client-Software ab. QuickTime von Apple beispielsweise verwendet die Ports 7070 und 7071 für den TCP-Datenkanal. Für den UDP-Kanal benutzt es das erste freie Port-Paar im Bereich von 6970 bis 6999.

Beide Services können über HTTP benutzt werden, was dann eigene Firewall-Regeln unnötig macht. Beide können den Datenkanal sowohl über TCP als auch über UDP aufbauen. HTTP und TCP bieten relativ sichere Verbindungen, UDP hat einen höheren Datendurchsatz.

Die Red Hat-Distribution enthält ein Proxy-Modul für RealAudio, das man für ankommende UDP-Datagramme sowie für den Schutz UDP-basierter RealAudio- und Quick-Time-Übertragungen einsetzt. Evtl. wird zusätzlich `ipmasqadm portfw` benötigt, um ankommende UDP-Pakete an einen LAN-Computer weiterzuleiten, der durch Masquerading verborgen ist.

Externe Konfiguration der Bastion als RealAudio-Client

Tabelle 4.56 enthält das Client-Verbindungsprotokoll für das externe Internet-Interface der Bastion.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	554	output	BASTION_EXTERNAL_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	554	input	BASTION_EXTERNAL_IPADDR	1024:65535	ACK
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	1024:65535	output	BASTION_EXTERNAL_IPADDR	7070:7071	Egal
Antwort des fremden Servers	TCP	ANYWHERE	1024:65535	input	BASTION_EXTERNAL_IPADDR	7070:7071	ACK

Tab. 4.56: Das RealAudio-Client-Protokoll für das externe Interface der Bastion

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	UDP	ANYWHERE	1024:65535	output	BASTION_EXTERNAL_IPADDR	6970:6999	-
Antwort des fremden Servers	UDP	ANYWHERE	1024:65535	input	BASTION_EXTERNAL_IPADDR	6970:6999	-

Tab. 4.56: Das RealAudio-Client-Protokoll für das externe Interface der Bastion (Forts.)

Das erste Regelpaar erlaubt Client-Kontrollverbindungen zwischen der Bastion und fremden RealAudio-Servern:

```
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p tcp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 554 -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 554 \
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT
```

Das zweite Paar ermöglicht Client-Datenkanäle zwischen der Bastion und fremden RealAudio-Servern über das TCP-Protokoll:

```
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p tcp \
-s $BASTION_EXTERNAL_IPADDR 7070:7071 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $BASTION_EXTERNAL_IPADDR 7070:7071 -j ACCEPT
```

Die letzten beiden Regeln gelten für Datenkanäle zwischen der Bastion und dem RealAudio-Server über UDP:

```
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR 6970:6999 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $ANYWHERE $UNPRIVPORTS \
-d $BASTION_EXTERNAL_IPADDR 6970:6999 -j ACCEPT
```

Hinweis

RealAudio/QuickTime und Masquerading

Wenn die Computer im LAN durch Masquerading verborgen sind, benötigen Sie noch die folgenden Einstellungen:

```
/sbin/modprobe ip_masq_raudio.o ports=554,7070,7071,6970,6971
/usr/sbin/ipmasqadm portfw -f
/usr/sbin/ipmasqadm portfw -a -P udp -L $BASTION_EXTERNAL_IPADDR
6970 -R $CHOKE_DMZ_IPADDR 6970
/usr/sbin/ipmasqadm portfw -a -P udp -L $BASTION_EXTERNAL_IPADDR
6971 -R $CHOKE_DMZ_IPADDR 6971
```

Weiterleitung von RealAudio-Daten über die DMZ-Seite der Bastion

Tabelle 4.57 zeigt das Protokoll für die Weiterleitung von RealAudio-Daten.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	ANYWHERE	554	Egal
Antwort des fremden Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	ANYWHERE	554	ACK
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_IPADDR	7070:7071	input	ANYWHERE	1024:65535	Egal
Antwort des fremden Servers	TCP	CHOKE_DMZ_IPADDR	7070:7071	output	ANYWHERE	1024:65535	ACK
Anfrage eines Clients von oder hinter der Choke	UDP	CHOKE_DMZ_IPADDR	6970:6999	input	ANYWHERE	1024:65535	-
Antwort des fremden Servers	UDP	CHOKE_DMZ_IPADDR	6970:6999	output	ANYWHERE	1024:65535	-

Tab. 4.57: Das RealAudio-Client-Protokoll: Weiterleitung über die DMZ-Seite der Bastion

Das erste Regelpaar gilt wieder für den Austausch von Kontrollinformationen zwischen der Choke und dem fremden RealAudio-Server:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 554 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 554 \
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Als Nächstes lassen wir Datenverbindungen über TCP zu:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKe_DMZ_IPADDR 7070:7071 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKe_DMZ_IPADDR 7070:7071 -j ACCEPT
```

Schließlich benötigen wir noch zwei Regeln für den Austausch von UDP-Datagrammen zwischen Choke und RealAudio-Server:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKe_DMZ_IPADDR 6970:6999 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKe_DMZ_IPADDR 6970:6999 -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als RealAudio-Client

Tabelle 4.58 zeigt die Protokolltabelle für RealAudio aus der Sicht der Choke.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	554	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	554	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	1024:65535	output	CHOKe_ DMZ_ IPADDR	7070:7071	Egal
Antwort des fremden Servers	TCP	ANYWHERE	1024:65535	input	CHOKe_ DMZ_ IPADDR	7070:7071	ACK

Tab. 4.58: Das RealAudio-Client-Protokoll: DMZ-Seite der Choke

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	UDP	ANYWHERE	1024:65535	output	CHOKE_ DMZ_ IPADDR	6970:6999	-
Antwort des frem- den Servers	UDP	ANYWHERE	1024:65535	input	CHOKE_ DMZ_ IPADDR	6970:6999	-

Tab. 4.58: Das RealAudio-Client-Protokoll: DMZ-Seite der Choke (Forts.)

Die ersten beiden Regeln akzeptieren den Austausch von Kontrollinformationen zwischen einem lokalen Client und dem fremden Server:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 554 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 554 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Die nächsten Regeln ermöglichen Datenverbindungen mit dem RealAudio-Server über TCP:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR 7070:7071 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 7070:7071 -j ACCEPT
```

Als Letztes stehen zwei Regeln für den Austausch von Daten über UDP:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR 6970:6999 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR 6970:6999 -j ACCEPT
```

4.4.20 Internet Relay Chat IRC (TCP-Port 6667)

Der Port 6667 ist der klassische Port für einen IRC-Server. Der Server kann wahlweise aber auch auf einem anderen Port laufen; in diesem Fall müssen die folgenden Regeln entsprechend angepasst werden.

Die Linux-Distribution enthält ein IRC-Proxy-Modul, das wegen des besonderen Protokolls eingesetzt werden muss. Dabei kommunizieren Clients untereinander über unprivilegierte Ports. Das Modul lässt auch ankommende Verbindungen von anderen Clients zu. Firmen und kommerzielle Firewalls sollten IRC wegen der inhärenten Sicherheitsrisiken nicht durch die Firewall hindurchlassen.

Externe Konfiguration der Bastion als IRC-Client

Tabelle 4.59 zeigt das Client-Protokoll für das externe Interface der Bastion. Die meisten Leute verwenden einen externen IRC-Server; Regeln für einen Server sind hier deshalb nicht angegeben.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	6667	output	BASTION_EXTERNAL_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	6667	input	BASTION_EXTERNAL_IPADDR	1024:65535	ACK
Verbindungsaufbau vom Choke-Client zu einem fremden Client	TCP	ANYWHERE	1024:65535	output	BASTION_EXTERNAL_IPADDR	1024:65535	Egal
Fremder Client antwortet auf Verbindungsaufbau vom Choke-Client	TCP	ANYWHERE	1024:65535	input	BASTION_EXTERNAL_IPADDR	1024:65535	ACK
Verbindungsaufbau von einem fremden Client zum Choke-Client	TCP	ANYWHERE	1024:65535	input	BASTION_EXTERNAL_IPADDR	1024:65535	Egal
Choke-Client antwortet auf Verbindungsaufbau von einem fremden Client	TCP	ANYWHERE	1024:65535	output	BASTION_EXTERNAL_IPADDR	1024:65535	ACK

Tab. 4.59: Das IRC-Protokoll für das externe Interface der Bastion

Die folgenden Regeln erlauben die Kommunikation zwischen lokalen IRC-Clients einerseits und fremden Servern sowie fremden Clients andererseits.

```
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p tcp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 6667 -j ACCEPT
```

```
ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 6667 \  
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p tcp \  
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT  
  
ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p tcp ! -y \  
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Hinweis

IRC und Masquerading

Wenn die LAN-Computer durch Masquerading verborgen sind, verwenden Sie zusätzlich die folgende Befehlszeile:

```
/sbin/modprobe ip_masq_irc.o
```

Weiterleitung von IRC-Daten über die DMZ-Seite der Bastion

Tabelle 4.60 zeigt die Protokolltabelle für die Weiterleitung von IRC-Daten über das DMZ-Interface der Bastion. Weil man meistens einen externen IRC-Server verwendet, sind keine Regeln für einen lokalen Server enthalten.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_ IPADDR	1024:65535	input	ANYWHERE	6667	Egal
Antwort des frem- den Servers	TCP	CHOKE_DMZ_ IPADDR	1024:65535	output	ANYWHERE	6667	ACK

Tab. 4.60: Das IRC-Protokoll: Weiterleitung über die DMZ-Seite der Bastion

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Verbindungsaufbau vom Choke-Client zu einem fremden Client	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	ANYWHERE	1024:65535	Egal
Fremder Client antwortet auf Verbindungsaufbau vom Choke-Client	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	ANYWHERE	1024:65535	ACK
Verbindungsaufbau von einem fremden Client zum Choke-Client	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	ANYWHERE	1024:65535	Egal
Choke-Client antwortet auf Verbindungsaufbau von einem fremden Client	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	ANYWHERE	1024:65535	ACK

Tab. 4.60: Das IRC-Protokoll: Weiterleitung über die DMZ-Seite der Bastion (Forts.)

Die nächsten Regeln gelten für das DMZ-Interface der Bastion; sie erlauben die Kommunikation zwischen lokalen IRC-Clients einerseits und fremden IRC-Clients und -Servern andererseits.

```

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE 6667 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE 6667 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

```



```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als IRC-Client

Tabelle 4.61 zeigt die clientseitige Protokolltabelle für IRC aus der Sicht der Choke. Weil man in der Regel einen externen IRC-Server benutzt, fehlen die Regeln für einen lokalen Server.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	TCP	ANYWHERE	6667	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	ANYWHERE	6667	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK
Verbindungsaufbau vom Choke-Client zu einem fremden Client	TCP	ANYWHERE	1024:65535	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Fremder Client beantwortet Verbindungsaufbau vom Choke-Client	TCP	ANYWHERE	1024:65535	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK
Verbindungsaufbau von einem fremden Client zum Choke-Client	TCP	ANYWHERE	1024:65535	input	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Choke-Client beantwortet Verbindungsaufbau von einem fremden Client	TCP	ANYWHERE	1024:65535	output	CHOKe_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.61: Das IRC-Protokoll: DMZ-Seite der Choke

Die folgenden Regeln erlauben lokalen Clients die Kommunikation mit fremden IRC-Servern und -Clients.

```
ipchains -A output -i $CHOKe_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 6667 -j ACCEPT  
  
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 6667 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT  
  
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p tcp ! -y \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

4.4.21 CU-SeeMe (UDP-Port 7648, 7649 und 24032; TCP-Ports 7648 und 7649

CU-SeeMe kommt ohne Zugriff auf einen fremden Server nicht aus. Darum beinhalten die hier vorgestellten Beispiele keine Server-Regeln.

Wegen der inhärenten Sicherheitsrisiken von UDP sollten Sie den CU-SeeMe-Server explizit angeben. Für Clients, die durch Masquerading verborgen sind, verwenden Sie das CU-SeeMe-Proxy-Modul, das in neueren Linux-Distributionen enthalten ist.

Hinweis

CU-SeeMe

Nähere Informationen zu CU-SeeMe im Zusammenspiel mit Firewalls erhalten Sie von www.cu-seeme.com und www.wpine.com.

Externe Konfiguration der Bastion als CU-SeeMe-Client

Tabelle 4.62 enthält die Client-Protokolltabelle für das externe, internetseitige Interface der Bastion.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients von oder hinter der Choke	UDP	CU-SeeMe-Server	7648:7649	output	BASTION_EXTERNAL_IPADDR	1024:65535	-

Tab. 4.62: Das CU-SeeMe-Client-Protokoll für das externe Interface der Bastion

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Antwort des fremden Servers	UDP	CU-SeeMe-Server	7648:7649	input	BASTION_EXTERNAL_IPADDR	1024:65535	-
Anfrage eines Clients von oder hinter der Choke	UDP	CU-SeeMe-Server	24032	output	BASTION_EXTERNAL_IPADDR	1024:65535	-
Antwort des fremden Servers	UDP	CU-SeeMe-Server	24032	input	BASTION_EXTERNAL_IPADDR	1024:65535	-
Anfrage eines Clients von oder hinter der Choke	TCP	CU-SeeMe-Server	7648:7649	output	BASTION_EXTERNAL_IPADDR	1024:65535	Egal
Antwort des fremden Servers	TCP	CU-SeeMe-Server	7648:7649	input	BASTION_EXTERNAL_IPADDR	1024:65535	ACK

Tab. 4.62: Das CU-SeeMe-Client-Protokoll für das externe Interface der Bastion (Forts.)

Die folgenden Regeln ermöglichen CU-SeeMe-Clients auf der Bastion die Kommunikation mit fremden Servern:

```
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d IP.meines.CU-SeeMe-Servers 7648:7649 -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s IP.meines.CU-SeeMe-Servers 7648:7649 \
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d IP.meines.CU-SeeMe-Servers 24032 -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s IP.meines.CU-SeeMe-Servers 24032 \
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p tcp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d IP.meines.CU-SeeMe-Servers 7648:7649 -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p tcp ! -y \
-s IP.meines.CU-SeeMe-Servers 7648:7649 \
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT
```

Weiterleitung von CU-SeeMe-Daten über die DMZ-Seite der Bastion

Die Protokolltabelle für das DMZ-Interface der Bastion finden Sie in Tabelle 4.63.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	UDP	CHOKE_DMZ_IPADDR	1024:65535	input	CU-SeeMe-Server	7648:7649	-
Antwort des fremden Servers	UDP	CHOKE_DMZ_IPADDR	1024:65535	output	CU-SeeMe-Server	7648:7649	-
Anfrage eines Clients von oder hinter der Choke	UDP	CHOKE_DMZ_IPADDR	1024:65535	input	CU-SeeMe-Server	24032	-
Antwort des fremden Servers	UDP	CHOKE_DMZ_IPADDR	1024:65535	output	CU-SeeMe-Server	24032	-
Anfrage eines Clients von oder hinter der Choke	TCP	CHOKE_DMZ_IPADDR	1024:65535	input	CU-SeeMe-Server	7648:7649	Egal
Antwort des fremden Servers	TCP	CHOKE_DMZ_IPADDR	1024:65535	output	CU-SeeMe-Server	7648:7649	ACK

Tab. 4.63: Das CU-SeeMe-Protokoll: Weiterleitung über die DMZ-Seite der Bastion

Die folgende Regelgruppe lässt Daten zwischen lokalen CU-SeeMe-Clients und fremden Servern passieren.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d IP.meines.CU-SeeMe-Servers 7648:7649 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s IP.meines.CU-SeeMe-Servers 7648:7649 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d IP.meines.CU-SeeMe-Servers 24032 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s IP.meines.CU-SeeMe-Servers 24032 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p tcp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d IP.meines.CU-SeeMe-Servers 7648:7649 -j ACCEPT  
  
ipchains -A output -i $BASTION_DMZ_INTERFACE -p tcp ! -y \  
-s IP.meines.CU-SeeMe-Servers 7648:7649 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als CU-SeeMe-Client

Das Client-Protokoll für CU-SeeMe auf der Choke wird in Tabelle 4.64 zusammengefasst.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	UDP	CU-SeeMe-Ser- ver	7648:7649	output	CHOKe_ DMZ_ IPADDR	1024:65535	-
Antwort des frem- den Servers	UDP	CU-SeeMe-Ser- ver	7648:7649	input	CHOKe_ DMZ_ IPADDR	1024:65535	-
Anfrage eines Clients von oder hinter der Choke	UDP	CU-SeeMe-Ser- ver	24032	output	CHOKe_ DMZ_ IPADDR	1024:65535	-
Antwort des frem- den Servers	UDP	CU-SeeMe-Ser- ver	24032	input	CHOKe_ DMZ_ IPADDR	1024:65535	-
Anfrage eines Clients von oder hinter der Choke	TCP	CU-SeeMe-Ser- ver	7648:7649	output	CHOKe_ DMZ_ IPADDR	1024:65535	Egal
Antwort des frem- den Servers	TCP	CU-SeeMe-Ser- ver	7648:7649	input	CHOKe_ DMZ_ IPADDR	1024:65535	ACK

Tab. 4.64: Das CU-SeeMe-Protokoll: DMZ-Seite der Choke

Die nächsten Regeln gelten für den Datenaustausch zwischen lokalen CU-SeeMe-Clients auf der Choke und dem zugehörigen Server.

```
ipchains -A output -i $CHOKe_DMZ_INTERFACE -p udp \  
-s $CHOKe_DMZ_IPADDR $UNPRIVPORTS \  
-d IP.meines.CU-SeeMe-Servers 7648:7649 -j ACCEPT  
  
ipchains -A input -i $CHOKe_DMZ_INTERFACE -p udp \  
-s IP.meines.CU-SeeMe-Servers 7648:7649 \  
-d $CHOKe_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $CHOKEDMZ_INTERFACE -p udp \  
-s $CHOKEDMZ_IPADDR $UNPRIVPORTS \  
-d IP.meines.CU-SeeMe-Servers 24032 -j ACCEPT  
  
ipchains -A input -i $CHOKEDMZ_INTERFACE -p udp \  
-s IP.meines.CU-SeeMe-Servers 24032 \  
-d $CHOKEDMZ_IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A output -i $CHOKEDMZ_INTERFACE -p tcp \  
-s $CHOKEDMZ_IPADDR $UNPRIVPORTS \  
-d IP.meines.CU-SeeMe-Servers 7648:7649 -j ACCEPT  
  
ipchains -A input -i $CHOKEDMZ_INTERFACE -p tcp ! -y \  
-s IP.meines.CU-SeeMe-Servers 7648:7649 \  
-d $CHOKEDMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.22 Quake (UDP-Ports 26000 sowie 1025 bis 1200)

Ein Quake-Server in der Voreinstellung wartet auf dem UDP-Port 26000 auf Anfragen von Spielteilnehmern. Wenn der Server einen neuen Spieler annimmt, teilt er ihm den UDP-Port mit, auf dem er mit seinem Client-Programm kommunizieren wird. Der Server-Port für den privaten Datenaustausch mit dem Client liegt üblicherweise im Bereich von 1025 bis 1200 UDP.

Wegen der inhärenten Sicherheitsrisiken von UDP sollten Sie den erlaubten Quake-Server explizit angeben. Zusätzlich verwenden Sie das Quake-Proxy-Modul, das bei neueren Linux-Distributionen mitgeliefert wird, um ankommende UDP-Pakete an Clients weiterzuleiten, die durch Masquerading verborgen sind.

Hinweis

Quake

Weitere Hinweise zur Implementierung von Firewalls, mit denen Quake zu-rechtkommt, finden Sie unter der URL <http://www.gamers.org/dEngine/quake/spec>.

Externe Konfiguration der Bastion für Quake-Clients

Aus der Sicht des externen, internetseitigen Netzwerkinterfaces der Bastion sieht das Quake-Protokoll aus wie in Tabelle 4.65.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP-Flags
Anfrage eines Clients von oder hinter der Choke	UDP	Quake-Server	26000	output	BASTION_EXTERNAL_IPADDR	1024:65535	-
Antwort des fremden Servers	UDP	Quake-Server	26000	input	BASTION_EXTERNAL_IPADDR	1024:65535	-
Anfrage eines Clients von oder hinter der Choke	UDP	Quake-Server	1025:1200	output	BASTION_EXTERNAL_IPADDR	1024:65535	-
Antwort des fremden Servers	UDP	Quake-Server	1025:1200	input	BASTION_EXTERNAL_IPADDR	1024:65535	-

Tab. 4.65: Das Quake-Protokoll für das externe Interface der Bastion

Die folgenden vier Regeln decken die Kommunikation zwischen Bastion und Quake-Server ab.

```
ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d IP.meines.Quake.Servers 26000 -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s IP.meines.Quake.Servers 26000 \
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS \
-d IP.meines.Quake.Servers 1025:1200 -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s IP.meines.Quake.Servers 1025:1200 \
-d $BASTION_EXTERNAL_IPADDR $UNPRIVPORTS -j ACCEPT
```

Weiterleitung von Quake-Daten über die DMZ-Seite der Bastion

Die Bastion leitet Quake-Daten über ihr DMZ-Interface an die Choke weiter. Die zugehörige Protokolltabelle ist in Tabelle 4.66 abgedruckt.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	UDP	CHOKE_DMZ_ IPADDR	1024:65535	input	Quake- Server	26000	-
Antwort des fremden Servers	UDP	CHOKE_DMZ_ IPADDR	1024:65535	output	Quake- Server	26000	-
Anfrage eines Clients von oder hinter der Choke	UDP	CHOKE_DMZ_ IPADDR	1024:65535	input	Quake- Server	1025:1200	-
Antwort des fremden Servers	UDP	CHOKE_DMZ_ IPADDR	1024:65535	output	Quake- Server	1025:1200	-

Tab. 4.66: Das Quake-Protokoll: Weiterleitung über die DMZ-Seite der Bastion

Die folgenden Regeln gelten auf dem DMZ-Interface der Bastion für die Client-Kommunikation zwischen lokalen Quake-Clients und fremden Servern.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d IP.meines.Quake.Servers 26000 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s IP.meines.Quake.Servers 26000 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d IP.meines.Quake.Servers 1025:1200 -j ACCEPT

ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s IP.meines.Quake.Servers 1025:1200 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Ein Quake-Server auf der Bastion

Ein Quake-Server kann nicht durch Masquerading verborgen werden. Sie müssen ihn entweder auf der Bastion selbst oder auf einem internen Server mit offizieller IP-Adresse betreiben.

Tabelle 4.67 enthält die Protokolltabelle für das externe Interface der Bastion, wenn ein lokaler Quake-Server mit fremden Quake-Clients kommunizieren soll.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines fremden Clients	UDP	ANYWHERE	1024:65535	input	BASTION_EXTERNAL_IPADDR	2600	-
Antwort des Servers auf der Bastion	UDP	ANYWHERE	1024:65535	output	BASTION_EXTERNAL_IPADDR	26000	-
Anfrage eines fremden Clients	UDP	ANYWHERE	1024:65535	input	BASTION_EXTERNAL_IPADDR	1025:1200	-
Antwort des Servers auf der Bastion	UDP	ANYWHERE	1024:65535	output	BASTION_EXTERNAL_IPADDR	1025:1200	-

Tab. 4.67: Das Quake-Server-Protokoll für die Bastion

Die folgenden Regeln erlauben den Datenaustausch zwischen fremden Quake-Clients und einem Server auf der Bastion.

```
ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $ANYWHERE $UNPRIVPORTS \
-d $BASTION_EXTERNAL_IPADDR 26000 -j ACCEPT

ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR 26000 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $ANYWHERE $UNPRIVPORTS \
-d $BASTION_EXTERNAL_IPADDR 1025:1200 -j ACCEPT

ipchains -A output -i $BASTION_EXTERNAL_INTERFACE -p udp \
-s $BASTION_EXTERNAL_IPADDR 1025:1200 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Konfiguration der DMZ-Seite der Choke als Quake-Client

Das Client-Protokoll für Quake auf der Choke ist in Tabelle 4.68 zusammengefasst.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port	TCP- Flags
Anfrage eines Clients von oder hinter der Choke	UDP	Quake-Server	26000	output	CHOKE_DMZ_IPADDR	1024:65535	-
Antwort des fremden Servers	UDP	Quake-Server	26000	input	CHOKE_DMZ_IPADDR	1024:65535	-
Anfrage eines Clients von oder hinter der Choke	UDP	Quake-Server	1025:1200	output	CHOKE_DMZ_IPADDR	1024:65535	-
Antwort des fremden Servers	UDP	Quake-Server	1025:1200	input	CHOKE_DMZ_IPADDR	1024:65535	-

Tab. 4.68: Das Quake-Protokoll: DMZ-Seite der Choke

Die folgenden Regeln erlauben die Kommunikation zwischen lokalen Quake-Clients und fremden Servern.

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d IP.meines.Quake.Servers 26000 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \
-s IP.meines.Quake.Servers 26000 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \
-d IP.meines.Quake.Servers 1025:1200 -j ACCEPT

ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \
-s IP.meines.Quake.Servers 1025:1200 \
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

4.4.23 Der Network Time Service NTP (UDP-Port 123)

Der *Network Time Service* NTP (»Netzwerkzeitdienst«) ermöglicht die Synchronisation Ihrer Systemuhr mit der eines Zeit-Providers. Damit hat man immer eine genaue Uhrzeit, besonders auch dann, wenn die interne Uhr falsch geht, oder nach dem Booten bzw. nach einem Stromausfall. Darüber hinaus eignet sich `xntpd` auch sehr gut als interner Dienst, um die Uhren aller lokalen Computer synchron zu halten.

Hinweis

Private LAN-Dienste

Bereits in Kapitel 1 wurde erwähnt, dass viele beliebte Unix-Services eigentlich nur für den Einsatz in einem internen LAN entwickelt wurden. Solche Services können Sicherheitsrisiken darstellen oder Ihre Nachbarn ärgern, wenn Sie externe Zugriffe darauf erlauben oder wenn versehentlich Pakete aus diesen Diensten ins Internet gelangen. In Kapitel 3 haben wir uns aus der Perspektive eines Paketfilters darüber unterhalten und auch ein paar Beispiele davon gesehen, welche Daten wir lieber nicht im Internet verbreiten wollen.

Einige dieser Services sind potenziell so gefährlich, dass man sie auf einem Bastion-Computer besser nicht laufen lässt. Manche Leute gehen sogar so weit, dass sie die entsprechende Software ganz von Firewall-Maschinen löschen. Natürlich könnte man die Dienste zwar auf der Bastion, aber hinter dem Schutz der Firewall betreiben. Der Grundgedanke hinter einer Bastion ist aber doch, dass die externe Maschine so gut wie nur irgend möglich abgesichert ist. Freilich, auf einem Einzelplatzsystem oder in einem ganz ganz kleinen Privatnetz wird man gewisse Kompromisse eingehen müssen. Wichtig ist aber auch hier, dass der Administrator die Risiken und möglichen Schutzmechanismen kennt, bevor er sich für eine Strategie entscheidet.

Intern können genau diese Dienste äußerst nützlich sein; manche LANs kommen ohne sie nicht aus. Sie machen einen wesentlichen Teil der Leistung und Flexibilität von Unix aus. Eine zusätzliche, interne Choke-Firewall bietet eine zusätzliche Schutzschicht, wenn man solche Services im LAN anbietet – von internen LAN-Servern oder auch von der Choke selbst.

In den folgenden Beispielen gehen die *ipchains*-Regeln davon aus, dass der Dienst den internen Clients von der Choke angeboten wird.

Die Bastion als lokaler NTP-Server

Tabelle 4.69 gibt eine Übersicht, welche Pakete an Client- und Server-Anfragen für einen NTP-Server auf der Bastion beteiligt sind.

Beschreibung	Protokoll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
Anfrage von einem Client auf der Choke	UDP	CHOKE_DMZ_IPADDR	1024:65535	input	BASTION_DMZ_IPADDR	123
Antwort des Servers auf der Bastion	UDP	CHOKE_DMZ_IPADDR	1024:65535	output	BASTION_DMZ_IPADDR	123
Anfrage von einem Server auf der Choke	UDP	CHOKE_DMZ_IPADDR	123	input	BASTION_DMZ_IPADDR	123

Tab. 4.69: Das NTP-Protokoll für einen Server auf der Bastion

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
Antwort des Servers auf der Bastion	UDP	CHOKE_DMZ_IPADDR	123	output	BASTION_DMZ_IPADDR	123

Tab. 4.69: Das NTP-Protokoll für einen Server auf der Bastion (Forts.)

Die Bastion synchronisiert sich von Zeit zu Zeit mit einem öffentlichen Zeit-Server. Die so erhaltene Zeit verteilt sie über einen lokalen `xntpd`-Server an die internen Maschinen.

Die Regeln für den Zugriff auf einen fremden Server finden Sie in Kapitel 3. Die folgenden Serverregeln erlauben Clients aus der DMZ die Abfrage des lokalen Servers auf der Bastion:

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $DMZ_LAN_ADDRESSES $UNPRIVPORTS \
-d $BASTION_DMZ_IPADDR 123 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR 123 \
-d $DMZ_LAN_ADDRESSES $UNPRIVPORTS -j ACCEPT
```

Server synchronisieren sich untereinander über Peer-to-Peer-Kommunikation. Für ein größeres LAN wird man möglicherweise auch einen sekundären `xntpd`-Server auf einer anderen Maschine einrichten.

```
ipchains -A input -i $BASTION_DMZ_INTERFACE -p udp \
-s $CHOKE_DMZ_IPADDR 123 \
-d $BASTION_DMZ_IPADDR 123 -j ACCEPT
```

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR 123 \
-d $CHOKE_DMZ_IPADDR 123 -j ACCEPT
```

Die Choke als NTP-Client und -Server

Tabelle 4.70 enthält die Protokolltabelle für die Choke.

Beschreibung	Proto- koll	Fremde IP	Frem- der Port	Chain	Lokale IP	Lokaler Port
Anfrage eines Clients auf der Choke	UDP	BASTION_DMZ_IPADDR	123	output	CHOKE_DMZ_IPADDR	1024:65535
Antwort des Bastion-Servers	UDP	BASTION_DMZ_IPADDR	123	input	CHOKE_DMZ_IPADDR	1024:65535

Tab. 4.70: Das NTP-Protokoll für Client und Server auf der Choke

Beschreibung	Proto- koll	Fremde IP	Frem- der Port	Chain	Lokale IP	Lokaler Port
Anfrage des Choke-Ser- vers	UDP	BASTION_DMZ_ IPADDR	123	output	CHOKE_DMZ_ IPADDR	123
Antwort des Bastion-Ser- vers	UDP	BASTION_DMZ_ IPADDR	123	input	CHOKE_DMZ_ IPADDR	123

Tab. 4.70: Das NTP-Protokoll für Client und Server auf der Choke (Forts.)

Das nächste Regelpaar erlaubt den Zugriff auf den `xntpd`-Server der Bastion mit dem Client-Programm `ntupdate`:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \  
-s $CHOKE_DMZ_IPADDR $UNPRIVPORTS \  
-d $BASTION_DMZ_IPADDR 123 -j ACCEPT
```

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \  
-s $BASTION_DMZ_IPADDR 123 \  
-d $CHOKE_DMZ_IPADDR $UNPRIVPORTS -j ACCEPT
```

Falls auf der Choke ein eigener `xntpd`-Server läuft, ermöglichen Sie ihm mit den folgenden Regeln den Peer-to-Peer-Zugriff auf die Bastion:

```
ipchains -A output -i $CHOKE_DMZ_INTERFACE -p udp \  
-s $CHOKE_DMZ_IPADDR 123 \  
-d $BASTION_DMZ_IPADDR 123 -j ACCEPT
```

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \  
-s $BASTION_DMZ_IPADDR 123 \  
-d $CHOKE_DMZ_IPADDR 123 -j ACCEPT
```

4.4.24 Protokolle an einen anderen Computer schicken (UDP-Port 514)

Die Protokollierung über das `syslog` lässt sich gut zentralisieren, was in einer Umgebung mit sehr vielen Servern gewisse Vorteile hat. Die Konfiguration von `syslogd` wird in Kapitel 6 behandelt; die Protokollierung an einen anderen Computer ist in Kapitel 7 besprochen.

Als Beispiel gehen wir davon aus, dass die Bastion eine Kopie ihres `syslogs` an die Choke schickt.

Bastion sendet syslog-Nachrichten

Tabelle 4.71 zeigt die Protokolltabelle.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
Bastion sendet eine Proto- kollmeldung	UDP	CHOKE_DMZ_IPADDR	514	output	BASTION_ DMZ_IPADDR	514

Tab. 4.71: Bastion sendet syslog-Nachrichten

Die Datei `/etc/syslog.conf` enthält einen zusätzlichen Eintrag, der eine Kopie aller Meldungen an die Choke schickt:

```
*,.* @choke
```

syslogd-Server tauschen sich untereinander über ein Peer-to-Peer-Protokoll aus. Dabei schreibt die Bastion an den Server auf der Choke:

```
ipchains -A output -i $BASTION_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR 514 \
-d $CHOKE_DMZ_IPADDR 514 -j ACCEPT
```

Choke empfängt syslog-Nachrichten

Tabelle 4.72 zeigt die zugehörige Tabelle für die Choke.

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
Bastion sendet eine Proto- kollmeldung	UDP	BASTION_DMZ_ IPADDR	514	input	CHOKE_DMZ_ IPADDR	514

Tab. 4.72: Choke empfängt syslog-Nachrichten

Auf der Choke modifizieren Sie für dieses Beispiel das Init-Skript für den `syslogd` in der Datei `/etc/rc.d/init.d/syslog`. Dabei übergeben Sie dem Dämon zusätzlich die Option `-r` und weisen ihn damit an, syslog-Meldungen nicht nur auf dem üblichen Unix-Domain-Socket entgegenzunehmen, sondern zusätzlich auch auf dem UDP-Port 514:

```
daemon syslogd -r
```

Die folgende Firewall-Regel sorgt dann dafür, dass die ankommenden Protokollmeldungen von der Bastion auch angenommen werden:

```
ipchains -A input -i $CHOKE_DMZ_INTERFACE -p udp \
-s $BASTION_DMZ_IPADDR 514 \
-d $CHOKE_DMZ_IPADDR 514 -j ACCEPT
```

4.4.25 Die Choke als lokaler DHCP-Server (UDP-Ports 67 und 68)

Tabelle 4.73 fasst das DHCP-Protokoll zusammen:

Beschreibung	Proto- koll	Fremde IP	Fremder Port	Chain	Lokale IP	Lokaler Port
DHCPDISCOVER; DHCPREQUEST	UDP	0.0.0.0	68	input	255.255.255.255	67
DHCPOFFER	UDP	255.255.255.255	68	output	0.0.0.0	67
DHCPOFFER	UDP	255.255.255.255	68	output	CHOKE_LAN_IPADDR	67
DHCPREQUEST; DHCPDECLINE	UDP	0.0.0.0	68	input	CHOKE_LAN_IPADDR	67
DHCPACK; DHCPNAK	UDP	CHOKE_LAN_ADDRESSES/CHOKE_LAN_NETMASK	68	output	CHOKE_LAN_IPADDR	67
DHCPACK	UDP	CHOKE_LAN_ADDRESSES	68	output	CHOKE_LAN_IPADDR	67
DHCPREQUEST; DHCPLEASE	UDP	CHOKE_LAN_ADDRESSES	68	input	CHOKE_LAN_IPADDR	67

Tab. 4.73: Das DHCP-Server-Protokoll für die Choke

Sie sollten zwar niemals DHCP-Server-Pakete ins Internet schicken, aber manche Leute benutzen einen privaten DHCP-Server für die Zuweisung von IP-Adressen an lokale Maschinen. DHCP ist dazu nicht nur auf einem größeren LAN mit vielen Computern nützlich, sondern auch für ein winziges privates LAN. Einige Leute betreiben sogar auf einem einzelnen Computer einen `dhcpd`-Server, wenn sie nämlich einen Laptop zwischen Arbeitsplatz und Wohnung hin- und hertransportieren. Wenn in der Arbeit die IP-Adressen dynamisch zugewiesen werden, erleichtert der DHCP-Server zu Hause den Transport des Laptops von einem Netzwerk ins andere.

In diesem Beispiel läuft der `dhcpd`-Server auf der Choke und weist den Computern im privaten LAN dynamisch ihre IP-Adressen zu.

```
ipchains -A input -i $CHOKE_LAN_INTERFACE -p udp \
-s $BROADCAST_0 68 \
-d $BROADCAST_1 67 -j ACCEPT

ipchains -A output -i $CHOKE_LAN_INTERFACE -p udp \
-s $BROADCAST_0 67 \
-d $BROADCAST_1 68 -j ACCEPT

ipchains -A output -i $CHOKE_LAN_INTERFACE -p udp \
-s $CHOKE_LAN_IPADDR 67 \
-d $BROADCAST_1 68 -j ACCEPT
```

```
ipchains -A input -i $CHOKE_LAN_INTERFACE -p udp \  
-s $BROADCAST_0 68 \  
-d $CHOKE_LAN_IPADDR 67 -j ACCEPT  
  
ipchains -A output -i $CHOKE_LAN_INTERFACE -p udp \  
-s $CHOKE_LAN_IPADDR 67 \  
-d $CHOKE_LAN_ADDRESSES/$CHOKE_LAN_NETMASK 68 -j ACCEPT  
  
ipchains -A output -i $CHOKE_LAN_INTERFACE -p udp \  
-s $CHOKE_LAN_IPADDR 67 \  
-d $CHOKE_LAN_ADDRESSES 68 -j ACCEPT  
  
ipchains -A input -i $CHOKE_LAN_INTERFACE -p udp \  
-s $CHOKE_LAN_ADDRESSES 68 \  
-d $CHOKE_LAN_IPADDR 67 -j ACCEPT
```

4.4.26 LAN-Zugriff auf die Choke

In einem Privathaushalt oder einer kleinen Firma benötigen Sie vermutlich keine Zugriffsbeschränkungen vom privaten LAN auf die Choke. Die folgenden beiden Regeln öffnen die Kommunikation zwischen LAN und Choke vollständig:

```
ipchains -A input -i $CHOKE_LAN_INTERFACE \  
-s $CHOKE_LAN_ADDRESSES -j ACCEPT  
  
ipchains -A output -i $CHOKE_LAN_INTERFACE \  
-d $CHOKE_LAN_ADDRESSES -j ACCEPT
```

4.4.27 IP-Masquerading

Masquerading ist für das private LAN eigentlich nicht notwendig: Die Bastion wendet ja schon auf allen internen Nachrichtenverkehr aus der DMZ Masquerading an. Aus Gründen der Vollständigkeit lassen wir die Choke dennoch Pakete aus dem privaten LAN maskieren und nicht einfach unverändert weiterleiten. Je nachdem, wie spezifisch die entsprechende Firewall-Regel auf der Bastion ist, kann Masquerading durch die Choke die Konfiguration der Bastion auch deutlich vereinfachen, wenn die Bastion nämlich nicht erwartet, dass Adressen von mehreren internen Netzwerken auftauchen.

Die folgende Regel wendet Masquerading auf alle Pakete aus dem privaten LAN an:

```
ipchains -A forward -i $CHOKE_DMZ_INTERFACE \  
-s $CHOKE_LAN_ADDRESSES -j MASQ
```

Wenn Sie stattdessen lieber explizit vorgehen, können Sie z.B. mit den folgenden beiden Regeln Masquerading für TCP und ICMP einschalten, nicht aber für UDP:


```
ipchains -A forward -i $CHOKE_DMZ_INTERFACE -p tcp \  
-s $CHOKE_LAN_ADDRESSES -j MASQ
```

```
ipchains -A forward -i $CHOKE_DMZ_INTERFACE -p icmp \  
-s $CHOKE_LAN_ADDRESSES -j MASQ
```

4.4.28 Protokollierung

Für die typische Zielgruppe dieses Buches, die ja doch in einer relativ vertrauenswürdigen Umgebung arbeitet, ist eine Protokollierung aller abgewiesenen Pakete wahrscheinlich nicht notwendig. Trotzdem ist die Protokollierung ein sehr nützliches Hilfsmittel bei der Fehlersuche in der Firewall sowie zum besseren Verständnis der Kommunikationsprotokolle. Weil zwischen dem privaten LAN und der Choke alle Pakete erlaubt sind, müsste man die Protokollierung gezielt für einzelne Ports aktivieren.

4.5 Zusammenfassung

Dieses Kapitel hat einige der Möglichkeiten beschrieben, die Ihnen beim Schutz eines LANs mit einer Firewall zur Verfügung stehen. Sicherheitsbestimmungen definiert man in Abhängigkeit von den individuellen Sicherheitsbedürfnissen der Site, der Bedeutung der geschützten Daten sowie den Kosten, die ein Verlust von Daten oder Privatsphäre verursachen würde. Ausgehend von einem einfachen Privat-LAN und der zugehörigen Firewall aus Kapitel 3 haben wir immer komplexere Konfigurationen diskutiert.

Das Hauptanliegen dieses Kapitels besteht darin, die Beispiel-Firewall aus Kapitel 3 zu einer formalen, lehrbuchmäßigen Firewall zu erweitern. Die Bastion hat dabei zwei Netzwerkinterfaces, eines zum Internet und eines zu einem Perimeter-Netzwerk, einer DMZ. Computer in der DMZ bieten öffentlich zugängliche Dienste an. Eine zweite Firewall, die Choke, ist ebenfalls an die DMZ angeschlossen und trennt sie vom internen, privaten Netzwerk. Die privaten Computer werden durch die Choke geschützt, und zwar auch noch im Falle einer kompromittierten Bastion oder eines Einbruchs in eine andere Maschine aus dem Perimeter-Netzwerk.

Manche Dienste, beispielsweise IRC oder RealAudio, eignen sich wegen ihrer speziellen Kommunikationsprotokolle nicht für den Einsatz in einer durch Paketfilter geschützten Umgebung, z.B. weil sie ankommende Verbindungen vom Server benötigen oder mehrfache Client-Server-Verbindungen über TCP und UDP verwenden. Solche Dienste erfordern andere Maßnahmen, z.B. spezielle Masquerading-Module des Kernels oder Proxies auf Anwendungsebene. Dieses Kapitel stellt Firewall-Regeln für einige solche Dienste vor, die durch die einfachen Beispiele aus Kapitel 3 nicht adäquat geschützt werden konnten.

Kapitel 5

Fehlersuche



- 5.1 Ein paar allgemeine Tipps für die Firewall-Entwicklung
- 5.2 Anzeigen der Firewall-Regeln
- 5.3 Die Regeln für die input-, output- und forward-Chains
- 5.4 Die Firewall-Regeln mit Einzelpaketen testen
- 5.5 Suche nach offenen Ports
- 5.6 Fehlersuche für ssh – ein Beispiel aus der Praxis
- 5.7 Zusammenfassung

Die Firewall ist also vorbereitet, installiert und aktiv. Und jetzt – geht gar nichts. Sie sind eingesperrt – oder noch schlimmer, nicht einmal X-Windows funktioniert mehr. Was geht da vor sich? Was tun? Wo fängt man überhaupt an?

Es ist unglaublich schwierig, die Regeln für eine Firewall ganz richtig hinzubekommen. Wenn Sie die Entwicklung per Hand erledigen, machen Sie unweigerlich Fehler. Selbst wenn Sie ein automatisches Tool zur Firewall-Erzeugung verwenden, müssen Sie das entstandene Skript noch anpassen.

In diesem Kapitel werden Features von `ipchains` und anderen Programmen besprochen, die zusätzliche Informationen über die Firewall liefern. Bei der Fehlersuche sind solche Daten von unschätzbarem Wert. Das vorliegende Kapitel demonstriert, was sie über die Firewall aussagen können. Aber seien Sie gewarnt: Die Tools, die wir verwenden, sind unhandlich; der ganze Prozess ist äußerst mühsam.

Zusätzliche Informationen zum Thema erhalten Sie aus der Manual-Page zu `ipchains` sowie aus dem `IPCHAINS-HOWTO`.

5.1 Ein paar allgemeine Tipps für die Firewall-Entwicklung

Die Fehlersuche in der Firewall kostet Zeit und Nerven. Abkürzungen gibt es im Fehlerfalle keine. Die folgenden Tipps können für ein wenig Erleichterung sorgen:

- Führen Sie die Regeln immer von einem vollständigen Test-Skript aus. Das Skript muss vorher alle existierenden Regeln löschen und die Policies neu setzen. Ansonsten wissen Sie nie so genau, welche Regeln gerade in welcher Reihenfolge gelten.
- Geben Sie keine neuen Regeln von der Kommandozeile ein. Und vor allem: Setzen Sie die Policy nicht von der Kommandozeile. Wenn Sie über X-Windows oder von einem anderen System (auch auf dem eigenen LAN) eingeloggt sind, könnten Sie sofort den Zugang verlieren.
- Lassen Sie das Test-Skript immer von der Konsole laufen. Debuggen Sie es nicht von einem anderen Rechner aus. Unter X-Windows ist die Arbeit vielleicht bequemer, aber es besteht trotzdem die Gefahr, dass Sie zwischendurch auch den Zugang zu X verlieren. Falls das passiert, wechseln Sie zu einer anderen virtuellen Konsole, um wieder Zugang zu erhalten.
- Wenn irgend möglich, beschränken Sie sich immer nur auf einen Dienst. Ergänzen Sie neue Regeln nur einzeln oder zumindest paarweise. Prüfen Sie die Ergebnisse immer sofort. Auf diese Weise lassen sich Probleme schneller abgrenzen.

- Denken Sie daran: Die erste passende Regel gewinnt. Die Reihenfolge der Regeln ist sehr wichtig. Benutzen Sie die Anzeigeeoptionen von `ipchains` und verschaffen Sie sich so ein Gefühl dafür, wie die Regeln zueinander stehen. Verfolgen Sie in der Vorstellung den Weg eines Paketes durch die Regelliste.
- Es gibt immer mindestens zwei unabhängige Chains, `input` und `output`. Wenn die `input`-Regeln in Ordnung sind, kann das Problem immer noch in der `output`-Chain liegen und umgekehrt.
- Wenn das Firewall-Skript hängen bleibt, benutzt eine Regel vermutlich einen symbolischen Hostnamen statt einer numerischen IP-Adresse, und die DNS-Regeln sind noch nicht aktiv. Regeln, die Hostnamen statt IP-Adressen verwenden, dürfen erst nach den DNS-Regeln stehen.
- Sehen Sie zweimal nach, ob die Syntax für `ipchains` stimmt. Bei der Richtung der Pakete vertippt man sich besonders leicht; vertauschen Sie probenhalber Absender- und Empfänger-Adressen und -Ports. Achten Sie auf die Groß- und Kleinschreibung der Optionen.
- Wenn ein Syntaxfehler auftritt, bricht das Firewall-Skript möglicherweise ab, ohne die folgenden Regeln zu installieren. Die Fehlermeldungen von `ipchains` sind – gelinde gesagt – etwas kryptisch. Wenn Ihnen nicht klar ist, wo das Problem liegt, führen Sie das Skript mit den Shell-Optionen `-x` oder `-v` aus, z.B. `sh -v /etc/rc.d/rc.firewall`. `-v` zeigt die Programmzeilen an, während Sie vom Interpreter der Shell gelesen werden, `-x` zeigt sie vor ihrer Ausführung an.
- Wenn ein bestimmter Service nicht funktioniert, schalten Sie über die `ipchains`-Option `-l` das Protokollieren in beide Richtungen ein. Wenn Sie einen Zugriff auf den Service versuchen, erhalten Sie in `/var/log/messages` Meldungen über abgewiesene Pakete (DENY oder REJECT)?
- Wenn Sie von der Firewall auf das Internet zugreifen können, aber nicht vom LAN, dann sehen Sie nach, ob in `/etc/sysconfig/network` bzw. in `/proc/sys/net/ipv4/ip_forward` IP-Forwarding eingeschaltet ist. In ersterer Datei wird Forwarding permanent konfiguriert; entweder per Hand (`FORWARD_IPV4=yes`) oder über das `control-panel` im Abschnitt über Routing. Wenn Sie die Datei in `/etc/sysconfig` verändern, muss das Netzwerk neu gestartet werden, oder Sie aktivieren Forwarding sofort über den Befehl:

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

- Wenn ein Service auf dem LAN funktioniert, aber nicht von außen, schalten Sie die Protokollfunktion für angenommene Pakete auf dem internen Interface ein. Greifen Sie ganz kurz einmal auf den Dienst zu und sehen Sie nach, welche Ports, Adressen, Flags usw. in beide Richtungen verwendet werden. Schalten Sie das Logging schnell wieder aus, bevor `/var/log/messages` überläuft.
- Wenn Sie mit einem Service überhaupt nicht zurechtkommen, setzen Sie ganz am Anfang des Firewall-Skripts vorübergehend ein Regelpaar ein, das *alle* Pa-

kete in beiden Richtungen akzeptiert und ins `syslog` schreibt. Funktioniert es jetzt? Wenn ja, finden Sie die benutzten Ports in `/var/log/messages`.

5.2 Anzeigen der Firewall-Regeln

Sehen Sie sich die definierten Firewall-Regeln an und überprüfen Sie, ob alles in der richtigen Reihenfolge vorhanden ist. Die `ipchains`-Option `-L` (`>list<`) zeigt die momentan im Kernel gespeicherten Regeln einer Chain an, und zwar in der Reihenfolge, in der sie mit Paketen verglichen werden.

Der List-Befehl hat folgendes Format:

```
ipchains -L input
ipchains -L output
ipchains -L forward
```

Im Gegensatz zur Neudefinition von Regeln kann das Anzeigen der bestehenden Firewall-Einstellungen gefahrlos von der Kommandozeile aus erfolgen. Die Ergebnisse erscheinen auf Ihrem Terminal; sie können Sie natürlich auch in eine Datei umleiten.

In den nächsten Beispielen enthält die `input`-Chain vier Beispielregeln, anhand derer ich die verfügbaren Anzeigeformate demonstriere und erkläre, was die einzelnen Felder bedeuten. Durch die unterschiedlichen Formate lassen sich die gleichen vier Regeln in unterschiedlicher Genauigkeit und Lesbarkeit anzeigen. Diese Formate sind für die verschiedenen Chains – `input`, `output` und `forward` – identisch.

5.2.1 ipchains -L input

Wenn Sie sich auf die Voreinstellungen für die Ausgabe verlassen, sehen die vier Beispielregeln so aus:

Hinweis

Zur besseren Lesbarkeit enthalten die Beispiele in diesem Kapitel Zeilennummern, die natürlich nicht Teil der Original-Ausgabe sind.

```
> ipchains -L input
```

```
1. Chain input (policy DENY):
2. target  prot  opt    source          destination      ports
3. DENY    all   ----1- my.host.domain  anywhere         n/a
4. ACCEPT  icmp  ----- anywhere        my.host.domain  echo-reply
5. ACCEPT  all   ----- anywhere        anywhere         n/a
6. ACCEPT  tcp   !y---- anywhere        my.host.domain  www ->
1024:65535
```

Zeile 1 gibt die Chain an, für die das Listing gilt – die `input`-Chain. Die Policy ist `DENY`, Pakete werden also kommentarlos abgewiesen, wenn keine Regel passt.

Zeile 2 enthält die Spaltenüberschriften:

- `target` bezieht sich darauf, was mit einem passenden Paket geschieht. Mögliche Werte sind `ACCEPT`, `DENY` und `REJECT`.
- `prot` steht für »Protokoll«: `all`, `tcp`, `udp` oder `icmp`.
- `opt` ist eine Abkürzung für »Optionen« und steht für die Flags. Am häufigsten sind gar keine Flags gesetzt. Sonst können u.a. folgende Flags auftreten: `l` (Protokollierung), `y` (SYN-Flag muss gesetzt sein), `!y` (ACK-Flag muss gesetzt sein).
- `source` ist die Absenderadresse des Paketes.
- `destination` ist die Empfängeradresse.
- Das Feld `ports` enthält die Absender- und Empfänger-Ports, den ICMP-Nachrichtentyp oder `n/a`, wenn keine entsprechenden Optionen für diese Regel gesetzt sind.

Zeile 3 ist die Regel gegen gefälschte Absenderadressen: sie weist ankommende Pakete von Ihrer eigenen IP-Adresse zurück und zeichnet sie auf.

Zeile 4 akzeptiert ankommende `ping`-Antworten auf Ihre abgehenden `pings`.

Zeile 5 veranschaulicht, dass die einfache Auflistung mit `-l` ohne zusätzliche Argumente einige wichtige Feinheiten verschweigt. Die Regel scheint alle ankommenden Pakete – `tcp`, `udp` und `icmp` – von überall zuzulassen. Das fehlende Detail ist in diesem Fall das Interface, `lo`. Es handelt sich um die Regel, die alle Pakete vom Loopback-Interface annimmt.

Zeile 6 akzeptiert Pakete von fremden Webservern, die Sie kontaktiert haben. Das Protokoll ist `tcp`. In den eingehenden Antworten vom Server muss das ACK-Bit gesetzt sein (`!y`). Der Absender-Port des Webserverns ist 80 oder `www`. Ihr Browser als Client benutzt einen der unprivilegierten Ports im Bereich von 1024 bis 65535.

5.2.2 `ipchains -L input -n`

Wenn Sie die Option `-n` (für »numerisch«) angeben, werden alle Felder als numerische Werte und nicht als symbolische Namen gedruckt. Dadurch sparen Sie viel Zeit, wenn in den Regeln viele IP-Adressen vorkommen, die ohne diese Option alle im DNS nachgeschlagen werden müssten. Außerdem sind Portbereiche in der Form `23:79` viel leichter lesbar, als wenn da `telnet:finger` stünde.

`ipchains -L input -n` gibt die gleichen vier Beispielregeln wie folgt aus:

```
> ipchains -L input -n
```

```
1. Chain input (policy DENY):
2. target      prot  opt      source      destination  ports
3. DENY        all   ----!-   192.168.10.30 0.0.0.0/0    n/a
4. ACCEPT      icmp  -----  0.0.0.0/0     192.168.10.30 0 -> *
5. ACCEPT      all   -----  0.0.0.0/0     0.0.0.0/0    n/a
6. ACCEPT      tcp   !y----   0.0.0.0/0     192.168.10.30 80 ->
                                           1024:65535
```

Zeile 1 gibt die Chain an, für die das Listing gilt – die input-Chain. Die Policy ist DENY, Pakete werden also kommentarlos abgewiesen, wenn keine Regel passt.

Zeile 2 enthält die Spaltenüberschriften:

- target bezieht sich darauf, was mit einem passenden Paket geschieht. Mögliche Werte sind ACCEPT, DENY und REJECT.
- prot steht für »Protokoll«: all, tcp, udp oder icmp.
- opt ist eine Abkürzung für »Optionen« und steht für die Flags. Am häufigsten sind gar keine Flags gesetzt. Sonst können u.a. folgende Flags auftreten: ! (Protokollierung), y (SYN-Flag muss gesetzt sein), !y (ACK-Flag muss gesetzt sein).
- source ist die Absenderadresse des Paketes.
- destination ist die Empfängeradresse.
- Das Feld ports enthält die Absender- und Empfänger-Ports, den ICMP-Nachrichtentyp oder n/a, wenn keine entsprechenden Optionen für diese Regel gesetzt sind.

Zeile 3 ist die Regel gegen gefälschte Absenderadressen: sie weist ankommende Pakete von Ihrer eigenen IP-Adresse (im Beispiel 192.168.10.30) zurück und zeichnet sie auf.

Zeile 4 akzeptiert ankommende ping-Antworten auf Ihre abgehenden pings.

Zeile 5 veranschaulicht, dass die einfache Auflistung mit -L ohne zusätzliche Argumente einige wichtige Feinheiten verschweigt. Die Regel scheint alle ankommenden Pakete – tcp, udp und icmp – von überall zuzulassen. Das fehlende Detail ist in diesem Fall das Interface, lo. Es handelt sich um die Regel, die alle Pakete vom Loopback-Interface annimmt.

Zeile 6 akzeptiert Pakete von fremden Webservern, die Sie kontaktiert haben. Das Protokoll ist tcp. In den eingehenden Antworten vom Server muss das ACK-Bit gesetzt sein (!y). Der Absender-Port des Webserver ist 80 oder www. Ihr Browser als Client benutzt einen der unprivilegierten Ports im Bereich von 1024 bis 65535.

5.2.3 ipchains -L input -v

Die Option `-v` (für »verbose« oder »ausführlich«) erhöht die Ausführlichkeit der von `ipchains -L` angezeigten Informationen. U.a. wird auch der Name des Interfaces mit ausgegeben. Das ist besonders dann nützlich, wenn ein Computer über mehr als ein Netzwerkinterface verfügt.

Die obigen vier Beispielregeln sehen mit dem Parameter `-v` so aus:

```
> ipchains -L input -v
```

```
1. Chain input (policy DENY: 60018 packets, 4120591 bytes):
2. pkts bytes target prot opt  tosa tosx ifname * source      destination ports
3.    0    0  DENY  all  ----1- 0xFF 0x00 eth0  my.host.domain anywhere    n/a
4.    0    0 ACCEPT icmp ----- 0xFF 0x00 eth0  anywhere    my.host.domain
                                           echo-reply
5. 61004 5987K ACCEPT  all  ----- 0xFF 0x00 lo    anywhere    anywhere    n/a
6. 2332 1597K ACCEPT  tcp  !y---- 0xFF 0x00 eth0  anywhere    my.host.domain
                                           www -> 1024:65535
```

Zeile 1 gibt die Chain an, für die das Listing gilt – die `input-Chain`. Die Policy ist `DENY`. 60018 Pakete mit insgesamt 4120591 Byte haben die `input-Chain` bisher passiert.

Zeile 2 enthält die Spaltenüberschriften:

- `pkts` zeigt die Zahl der Pakete, auf die die jeweilige Regel gepasst hat.
- `bytes` ist die gesamte Größe dieser Pakete in Byte.
- `target` zeigt an, was mit passenden Paketen geschieht – `ACCEPT`, `DENY` oder `REJECT`.
- `prot` steht für »Protokoll«: `all`, `tcp`, `udp` oder `icmp`.
- `opt` ist eine Abkürzung für »Optionen« und steht für die Flags: `keine`, `1` (Protokollierung), `y` (SYN-Flag muss gesetzt sein), `!y` (ACK-Flag muss gesetzt sein).
- `tosa` und `tosx` sind die Type-of-Service-Masken (TOS), `tosa` für `and` und `tosx` für `xor`. Vier der TOS-Bits entsprechen vier verschiedenen Prioritäten für die Paketzustellung: kleinstmögliche Verzögerung, größtmögliche Übertragungsrate, größtmögliche Zuverlässigkeit sowie kleinstmögliche Kosten.

Die hier angegebenen Masken verändern die TOS-Bits von Paketen, auf die die jeweilige Regel passt. Die bereits im Paket enthaltenen Bits werden dabei über logisches *und* mit der `tosa`-Maske verknüpft und anschließend über logisches *exklusiv-oder* mit der `tosx`-Maske. (Wenn eine Regel das Paket nicht akzeptiert, macht eine Veränderung der TOS-Flags natürlich keinen Sinn.)

Hinweis***Die Type-of-Service-Flags (TOS)***

Die TOS-Bits aus dem IP-Header werden kaum benutzt. Bestenfalls lassen sie sich als Tipp oder als Präferenzangabe verstehen. Der Empfänger ignoriert sie vermutlich völlig. Wenn Sie von TOS noch nie etwas gehört haben, dann ignorieren Sie die ganze Geschichte am besten. Normale Anwender verwenden all das ohnehin nicht.

Die TOS-Flags sind aber immerhin ein offizieller Teil des IP-Protokollstandards. In der Erwartung, dass sie einmal benötigt wurden, hat man damals Platz für sie vorgesehen. Stattdessen hat sich IP zu einem ganz einfachen Protokoll entwickelt, das versucht, jedes Paket gleichermaßen optimal zu behandeln.

Die aktuellen Vorschläge für neue Routing-Standards greifen wieder auf die TOS-Bits zurück und wollen sie für »QoS« benutzen – Quality of Service, also unterschiedliche Qualitätsgrade für unterschiedliche Benutzer und/oder Verbindungstypen. Z.B. könnte eine telnet-Sitzung mit niedriger Bandbreite, aber hohen Anforderungen an die Reaktionszeit anders behandelt werden als eine Videoübertragung, die vor allem eine große Bandbreite braucht.

- `ifname` zeigt, für welches Netzwerkinterface – z.B. `eth0`, `eth1`, `lo` oder `ppp0` – diese Regel gilt. Nur Pakete von oder zu genau diesem Interface werden berücksichtigt. Dieses Feld ist in einem LAN sehr wichtig, wenn für die einzelnen Interfaces einer Maschine unterschiedliche Regeln gelten sollen.
- * ist ein Platzhalter für zwei unbenutzte Felder, die ich im Beispiel aus Platzgründen weggelassen habe:
 - `mark` wird momentan nicht benutzt und ist nicht gut dokumentiert.
 - `outsize` ist nicht dokumentiert.

Hinweis***Die unbenutzten Felder mark und outsize***

mark und outsize sind Platzhalter für Funktionalität, die noch nicht existiert. Die Markierung von Paketen (Packet marking) gehört zum Konzept der Quality of Service oder QoS. Die TOS-Bits werden nicht verwendet, und Entwicklungsfirmen und Standard-Komitees wollen sie durch QoS-Bits ersetzen.

- `source` ist die Absenderadresse des Pakets.
- `destination` ist die Empfängeradresse.
- Die Spalte `ports` enthält Absender- und Empfänger-Port, den ICMP-Nachrichtentyp oder `n/a`, wenn die Regel nichts derartiges vorgibt.

Zeile 3 ist die Regel gegen gefälschte Absenderadressen: sie weist ankommende Pakete von Ihrer eigenen IP-Adresse zurück und zeichnet sie auf.

Zeile 4 akzeptiert ankommende ping-Antworten auf Ihre abgehenden pings.

Zeile 5 demonstriert die Ausgabe nützlicher Details durch die Option `-v`. Jetzt ist klar, dass sich diese Regel nur auf das Loopback-Interface `lo` bezieht. Diese Regel akzeptiert also alle Pakete von Loopback.

Zeile 6 akzeptiert Pakete von fremden Webservern, die Sie kontaktiert haben. Das Protokoll ist `tcp`. In den eingehenden Antworten vom Server muss das ACK-Bit gesetzt sein (`!y`). Der Absender-Port des Webserver ist 80 oder `www`. Ihr Browser als Client benutzt einen der unprivilegierten Ports im Bereich von 1024 bis 65535.

5.2.4 ipchains -L input -nv

Wenn Sie die Optionen `-n` (numerisch) und `-v` (verbose) gleichzeitig angeben, werden Ihnen die vier Beispielregeln folgendermaßen angezeigt:

```
> ipchains -L input -nv
```

```
1. Chain input (policy DENY: 60018 packets, 4120591 bytes):
2.  pkts bytes target prot opt  tosa tosx ifname * source      destination  ports
3.    0    0 DENY  all  ----l- 0xFF 0x00 eth0    192.168.10.30 0.0.0.0/0    n/a
4.    0    0 ACCEPT icmp  ----- 0xFF 0x00 eth0    0.0.0.0/0    192.168.10.30 0 -> *
5. 61004 5987K ACCEPT all  ----- 0xFF 0x00 lo      0.0.0.0/0    0.0.0.0/0    n/a
6. 2332 1597K ACCEPT tcp  !y---- 0xFF 0x00 eth0    0.0.0.0/0    192.168.10.30
                                           80 -> 1024:65535
```

Zeile 1 gibt die Chain an, für die das Listing gilt – die `input-Chain`. Die Policy ist `DENY`. 60018 Pakete mit insgesamt 4120591 Byte haben die `input-Chain` bisher passiert.

Zeile 2 enthält die folgenden Spaltenüberschriften:

- `pkts` zeigt die Zahl der Pakete, auf die die jeweilige Regel gepasst hat.
- `bytes` ist die gesamte Größe dieser Pakete in Byte.
- `target` zeigt an, was mit passenden Paketen geschieht – `ACCEPT`, `DENY` oder `REJECT`.
- `prot` steht für »Protokoll«: `all`, `tcp`, `udp` oder `icmp`.
- `opt` ist eine Abkürzung für »Optionen« und steht für die Flags: keine, `l` (Protokollierung), `y` (SYN-Flag muss gesetzt sein), `!y` (ACK-Flag muss gesetzt sein).
- `tosa` und `tosx` sind die Type-of-Service-Masken (TOS), `tosa` für `and` und `tosx` für `xor`. Vier der TOS-Bits entsprechen vier verschiedenen Prioritäten für die Paketzustellung: kleinstmögliche Verzögerung, größtmögliche Übertragungsrate, größtmögliche Zuverlässigkeit sowie kleinstmögliche Kosten. Die im Pa-

ket enthaltenen Bits werden über logisches *und* mit *tos* verknüpft, über logisches *exklusiv-oder* mit *tosx*.

- *ifname* zeigt, für welches Netzwerkinterface – z.B. *eth0*, *eth1*, *lo* oder *ppp0* – diese Regel gilt. Nur Pakete von oder zu genau diesem Interface werden berücksichtigt. Dieses Feld ist in einem LAN sehr wichtig, wenn für die einzelnen Interfaces einer Maschine unterschiedliche Regeln gelten sollen.
- * ist ein Platzhalter für zwei unbenutzte Felder, die ich im Beispiel aus Platzgründen weggelassen habe:
 - *mark* wird momentan nicht benutzt und ist nicht gut dokumentiert.
 - *outsize* ist nicht dokumentiert.
- *source* ist die Absenderadresse des Pakets.
- *destination* ist die Empfängeradresse.
- Die Spalte *ports* enthält Absender- und Empfänger-Port, den ICMP-Nachrichtentyp, bzw. *n/a*, wenn die Regel nichts derartiges vorgibt.

Zeile 3 ist die Regel gegen gefälschte Absenderadressen: sie weist ankommende Pakete von Ihrer eigenen IP-Adresse zurück und zeichnet sie auf.

Zeile 4 akzeptiert ankommende *ping*-Antworten auf Ihre abgehenden *pings*.

Zeile 5 demonstriert die Ausgabe nützlicher Details durch die Option *-v*. Jetzt ist klar, dass sich diese Regel nur auf das Loopback-Interface *lo* bezieht. Diese Regel akzeptiert also alle Pakete von Loopback.

Zeile 6 akzeptiert Pakete von fremden Webservern, die Sie kontaktiert haben. Das Protokoll ist *tcp*. In den eingehenden Antworten vom Server muss das ACK-Bit gesetzt sein (!y). Der Absender-Port des Webserverns ist 80 oder *www*. Ihr Browser als Client benutzt einen der unprivilegierten Ports im Bereich von 1024 bis 65535.

5.3 Die Regeln für die input-, output- und forward-Chains

Nachdem wir uns jetzt angesehen haben, wie eine Firewall-Chain angezeigt wird und mit welchen Optionen Sie das Format dieser Anzeige beeinflussen können, wollen wir jetzt ein paar kurze Regellisten für die *input*-, *output*- und *forward*-Chains genauer untersuchen. Viele der in den Beispielen gezeigten Regeln werden Sie in Ihren eigenen Regeln wiederfinden.

5.3.1 Die input-Chain

Bei einer Policy von *DENY* werden die meisten Regeln vom Typ *ACCEPT* sein. Nachdem die Voreinstellung erst einmal alles blockiert, müssen Sie explizite Ausnahmen definieren, welche Pakete doch angenommen werden sollen. Das folgende Beispiel ist eine repräsentative Auswahl solcher Regeln:

```
> ipchains -L input
```

```
Chain input (policy DENY):
target prot opt      source                destination            ports

1. ACCEPT icmp ----- anywhere             my.host.domain destination-unreachable
2. ACCEPT icmp ----- anywhere             my.host.domain source-quench
3. ACCEPT icmp ----- anywhere             my.host.domain time-exceeded
4. ACCEPT icmp ----- anywhere             my.host.domain parameter-problem
5. ACCEPT udp ----- isp.name.server my.host.domain domain -> domain
6. ACCEPT udp ----- isp.name.server my.host.domain domain -> 1024:65535
7. ACCEPT tcp !y---- isp.name.server my.host.domain domain -> 1024:65535
8. REJECT tcp ----- anywhere             my.host.domain 1024:65535 -> auth
9. ACCEPT tcp !y---- anywhere             my.host.domain auth -> 1024:65535
10. ACCEPT tcp ----- anywhere             my.host.domain 1024:65535 -> www
11. ACCEPT tcp !y---- anywhere             my.host.domain www -> 1024:65535
12. ACCEPT tcp !y---- isp.news.server my.host.domain nntp -> 1024:65535
13. ACCEPT tcp ----- anywhere             my.host.domain 1024:65535 -> smtp
14. ACCEPT tcp !y---- anywhere             my.host.domain smtp -> 1024:65535
```

Die Policy ist hier DENY, d.h. ein Paket, das zu keiner Regel passt, wird stillschweigend verworfen, ohne dass eine Rückmeldung an den Absender erfolgt. Insgesamt gibt es 14 Regeln in dieser Chain:

- Zeile 1: Ankommende ICMP-Fehlermeldungen vom Typ `destination-unreachable` werden von überall angenommen.
- Zeile 2: Ankommende ICMP-Flusskontrollnachrichten vom Typ `source-quench` werden von überall angenommen.
- Zeile 3: Ankommende ICMP-Fehlermeldungen vom Typ `time-exceeded` werden von überall angenommen.
- Zeile 4: Ankommende ICMP-Fehlermeldungen vom Typ `parameter-problem` werden von überall angenommen.
- Zeile 5: Ein UDP-Paket, das als Absender den Nameserver Ihres Internet-Providers hat, wird angenommen, sofern Absender- und Empfänger-Ports beide 53 sind, der `domain`-Port. Dabei handelt es sich um eine Datenübertragung von einem Nameserver zu einem anderen, wenn z.B. Ihr lokaler Nameserver eine Anfrage an den Nameserver Ihres Providers weitergeleitet hat und jetzt die Antwort eintrifft.
- Zeile 6: Ein UDP-Paket, das als Absender den Nameserver Ihres Internet-Providers hat, wird angenommen, sofern der Absender-Port 53 ist (der `domain`-Port) und der Empfänger-Port im unprivilegierten Bereich liegt. Diese Sorte Paket ist eine Antwort des DNS-Servers auf eine Anfrage von einem Client auf Ihrer Maschine.

- Zeile 7: Ein TCP-Paket, das als Absender den Nameserver Ihres Internet-Providers hat, wird angenommen, sofern der Absender-Port 53 (`domain`) ist, das ACK-Bit gesetzt ist und der Empfänger-Port im unprivilegierten Bereich liegt. Dabei handelt es sich um die TCP-Antwort des DNS-Servers auf eine Anfrage von einem Client auf Ihrer Maschine, wenn zuvor eine UDP-Anfrage fehlgeschlagen ist.
- Zeile 8: Alle ankommenden Pakete an den lokalen `identd`-Server (Port `auth` bzw. 113) werden abgewiesen. Eine ICMP-Fehlermeldung vom Typ 3, `destination-unreachable`, geht an den Absender der Anfrage zurück.
- Zeile 9: Alle ankommenden TCP-Pakete vom `auth`-Port 113 an einen unprivilegierten Empfänger-Port auf unserer Seite werden angenommen, sofern das ACK-Bit gesetzt ist. Diese Pakete enthalten die Antwort eines fremden `auth`-Servers auf eine Anfrage eines Ihrer Clients.
- Zeile 10: Ankommende TCP-Pakete von einem unprivilegierten Port auf einer beliebigen Absenderadresse werden angenommen, wenn Sie an den Port 80 Ihres Webservers gerichtet sind. Dabei ist es egal, ob die SYN- oder ACK-Flags gesetzt sind oder nicht.
- Zeile 11: Ankommende TCP-Antworten vom Port 80 auf beliebigen Absenderadressen werden angenommen, wenn in ihnen das ACK-Bit gesetzt ist und sie an einen lokalen unprivilegierten Port gehen. Solche Pakete sind Antworten eines fremden Webservers auf eine Anfrage Ihres Webbrowsers.
- Zeile 12: Ankommende TCP-Antworten vom `nntp`-Port 119 auf dem Newsserver Ihres Internet-Providers werden angenommen, wenn in ihnen das ACK-Bit gesetzt ist und sie an einen lokalen unprivilegierten Port gehen. Hier handelt es sich um die Antwort-Pakete des Newsservers, wenn Sie Usenet-News lesen.
- Zeile 13: Ankommende TCP-Pakete von beliebigen Absenderadressen und unprivilegierten Absender-Ports werden angenommen, wenn sie an den `smtp`-Port 25 Ihres Mailservers gerichtet sind. SYN- und ACK-Flags werden nicht beachtet, sodass jemand anders eine E-Mail an Ihren Server ausliefern kann.
- Zeile 14: Ankommende TCP-Pakete vom `smtp`-Port 25 auf beliebigen Absenderadressen werden angenommen, wenn bei gesetztem ACK-Bit der Empfänger einer Ihrer unprivilegierten Ports ist. Diese Pakete sind die Antworten fremder Mailserver, wenn Sie eine Mail verschicken.

5.3.2 Die output-Chain

Bei einer Policy von `DENY` oder `REJECT` sind die meisten Regeln vom Typ `ACCEPT`. Nachdem die Voreinstellung erst einmal alles blockiert, definieren Sie explizite Ausnahmen, welche Pakete doch erlaubt werden sollen. Das folgende Beispiel ist eine repräsentative Auswahl solcher Regeln:

> ipchains -L output

```
Chain output (policy REJECT):
target prot opt    source      destination  ports

1. ACCEPT icmp ----- my.host.domain anywhere      destination-un-
   reachables
2. ACCEPT icmp ----- my.host.domain anywhere      source-quench
3. ACCEPT icmp ----- my.host.domain isp.address.range time-exceeded
4. ACCEPT icmp ----- my.host.domain anywhere      parameter-pro-
   blem
5. ACCEPT udp ----- my.host.domain isp.name.server domain -> domain
6. ACCEPT udp ----- my.host.domain isp.name.server 1024:65535-> do-
   main
7. ACCEPT tcp ----- my.host.domain isp.name.server 1024:65535-> do-
   main
8. ACCEPT tcp ----- my.host.domain anywhere      1024:65535 ->
   auth
9. ACCEPT tcp !y---- my.host.domain anywhere      www ->
   1024:65535
10. ACCEPT tcp ----- my.host.domain anywhere      1024:65535 ->
   www
11. ACCEPT tcp ----- my.host.domain isp.news.server 1024:65535 ->
   nntp
12. ACCEPT tcp !y---- my.host.domain anywhere      smtp ->
   1024:65535
13. ACCEPT tcp ----- my.host.domain anywhere      1024:65535 ->
   smtp
```

Die Policy für die output-Chain ist REJECT. Wenn also eines Ihrer Programme eine Verbindung aufbauen will, die nicht erlaubt ist, erhält es sofort eine entsprechende Fehlermeldung. Die Chain enthält im Beispiel 13 Regeln:

- Zeile 1: Abgehende ICMP-Fehlermeldungen vom Typ 3 sind überallhin erlaubt. Obwohl dieser Fehler *destination-unreachable* heißt, handelt es sich in Wirklichkeit um eine sehr allgemeine Fehlermeldung. Solche Nachrichten enthalten ein zusätzliches Feld, das die genaue Art des aufgetretenen Fehlers angibt.
- Zeile 2: ICMP-Flusskontrollnachrichten vom Typ *source-quench* dürfen nach überall hin verschickt werden.
- Zeile 3: ICMP-Fehlermeldungen des Typs *time-exceeded* sind nur an Computer Ihres Internet-Providers zulässig. Das bedeutet, dass Sie nur auf *tracero* des Ihres Providers reagieren.
- Zeile 4: ICMP-Fehlermeldungen vom Typ *parameter-problem* dürfen nach überall hin verschickt werden.

- Zeile 5: UDP-Pakete mit dem Nameserver Ihres Internet-Providers als Empfänger werden angenommen, wenn sowohl Absender- als auch Empfänger-Port der `domain`-Port 53 sind. Bei solchen Paketen handelt es sich um eine Datenübertragung von einem Nameserver zum anderen, z.B. wenn Ihr lokaler Nameserver eine Anfrage an den Server Ihres Providers weiterleitet, weil die benötigten Informationen nicht im Cache stehen.
- Zeile 6: UDP-Pakete mit dem Nameserver Ihres Internet-Providers als Empfänger werden angenommen, wenn der Absender-Port im unprivilegierten Bereich liegt und der Empfänger-Port der `domain`-Port 53 ist. Dies ist eine UDP-Anfrage eines DNS-Clients auf Ihrem Computer an den fremden Server.
- Zeile 7: TCP-Pakete mit dem Nameserver Ihres Internet-Providers als Empfänger-Adresse und beliebig gesetzten SYN- und ACK-Flags sind erlaubt, sofern der Absender-Port im unprivilegierten Bereich liegt und der Empfänger-Port 53 ist (`domain`). So eine TCP-Anfrage eines DNS-Clients auf Ihrem System erfolgt, wenn eine vorherige Anfrage über UDP fehlschlägt.
- Zeile 8: Alle abgehenden TCP-Pakete von unprivilegierten Ports an den `auth`-Port 113 sind erlaubt. Dabei stellt ein Client auf Ihrem Computer eine `auth`-Anfrage an einen fremden `identd`-Server.
- Zeile 9: Abgehende TCP-Antworten an unprivilegierte Ports auf beliebigen Empfänger-Adressen sind von Ihrem Webserver auf Port 80 gestattet. Das ACK-Bit muss gesetzt sein, weil es sich bei solchen Paketen um Antworten auf eingehende Anfragen handelt.
- Zeile 10: Abgehende TCP-Anfragen an den Port 80 auf beliebigen Ziel-IPs sind von lokalen unprivilegierten Ports erlaubt. Dabei handelt es sich um Anfragen und bestehende Verbindungen von Ihrem Web-Browser zu einem fremden Web-Server.
- Zeile 11: Abgehende TCP-Anfragen an den `nntp`-Port 119 auf dem Newsserver Ihres Internet-Providers sind von lokalen unprivilegierten Ports erlaubt. Solche Pakete werden versandt, wenn Sie Usenet-News lesen.
- Zeile 12: Abgehende TCP-Pakete vom `smtp`-Port 25 Ihres lokalen Mailservers sind erlaubt, wenn das ACK-Bit gesetzt ist und der Empfänger-Port im unprivilegierten Bereich liegt. Diese Pakete sind Antworten an fremde Mail-Clients, die eine E-Mail bei Ihrem SMTP-Server abliefern.
- Zeile 13: Abgehende TCP-Pakete von lokalen unprivilegierten Ports an den `smtp`-Port 25 auf einem fremden Rechner werden ebenfalls akzeptiert; sie entstehen, wenn Sie eine E-Mail versenden.

5.3.3 Die forward-Chain

Die forward-Chain liegt in der Mitte zwischen Eingabe und Ausgabe. Ein ankommendes Paket muss zunächst durch die input-Chain akzeptiert werden. Ist das der Fall und ist die Empfängeradresse des Paketes eine andere als die Adresse des Netzwerkinterfaces, auf dem das Paket angekommen ist, so gelangt das Paket in die forward-Chain. Wenn eine der Weiterleitungsregeln passt, wird es in die output-Chain des nächsten Netzwerkinterfaces eingespeist. Sofern auch diese es akzeptiert, wird das Paket an den nächsten Computer weitergesandt.

Zur besseren Veranschaulichung gilt die als Nächstes gezeigte Firewall-Regel nur für TCP-Pakete des internen Netzwerkinterfaces. UDP- und ICMP-Pakete werden hier nicht weitergeleitet.

Hinweis

ICMP-Masquerading unter Linux

Bis vor kurzem wurden nur ICMP-Fehlermeldungen durch Masquerading korrekt behandelt, sofern nicht ein spezieller ICMP-Masquerading-Dienst explizit in den Kernel hineinkompiliert wurde. Red Hat Version 6.0 aktiviert ICMP-Masquerading schon in der Voreinstellung, d.h. interne Computer können fremde Rechner anpingen oder traceroutes durchführen, ohne dass man vorher den Kernel neu übersetzen müsste.

Dieser Abschnitt beruht auf einer Beispielregel für die Weiterleitung. Ein kleines, privates Netzwerk braucht aber auch keine ausgefeilten Weiterleitungsregeln.

```
ipchains -A forward -i $EXTERNAL_INTERFACE -p tcp -s $INTERNAL_LAN_
                                         ADDRESSES -j MASQ
```

```
> ipchains -L forward -v
```

```
Chain forward (policy REJECT: 0 packets, 0 bytes):
 pkts bytes target prot opt  tos  tosx ifname source destination ports
  80  4130 MASQ  tcp  ----- 0xFF 0x00 eth0  choke anywhere any->any
```

Die Policy ist REJECT, d.h. ohne ausdrückliche Regeln werden keine Pakete weitergeleitet.

In dieser Situation hilft uns die Option `-v` wieder. Die Regel besagt, dass Pakete aus dem internen LAN (choke) an das externe Interface eth0 weitergeleitet werden sollen. Allerdings erfolgt diese Weiterleitung nur dann, wenn die Empfängeradresse die eines fremden Computers ist.

Mit `ipchains` kann man das Interface des internen LANs – also das Interface, über das ein Paket ankommt – nicht explizit angeben. Es ist vielmehr implizit über die Absenderadresse des Pakets bestimmt. Ein Paket mit einer Absenderadresse aus dem LAN, das auf dem internen Netzwerkinterface ankommt, gehört nicht ins

gleiche Netzwerk wie das externe Netzwerkinterface. Wenn die Empfängeradresse nicht die interne Netzwerkadresse ist, muss sie notwendig eine fremde Adresse sein, zumindest aus der Sicht der internen Interfaces.

Beachten Sie, dass diese Pakete vom externen Interface zurückgewiesen würden, wenn Masquerading nicht eingeschaltet wäre. Die Masquerading-Regeln kommen zum Zuge, wenn das Paket die `input`-Warteschlange des internen Interfaces verlässt, noch bevor es in die `output`-Schlange des externen Interfaces gelangt. Wenn Ihre Firewall alle abgehenden Pakete mit privaten Absenderadressen ablehnt, werden Pakete aus Ihrem LAN also am externen Interface verworfen, sofern die Absenderadresse nicht durch Masquerading umgeschrieben wurde.

Der nächste wichtige Punkt ist, dass diese Regel nur für TCP-Pakete gilt. Weder UDP-Pakete noch ICMP-Kontrollnachrichten werden weitergeleitet. Wie schon gesagt, selbst wenn UDP- oder ICMP-Pakete prinzipiell die Firewall verlassen dürften, werden sie von internen Maschinen nicht weitergeleitet, wenn sie nicht in einer Forward- oder Masquerading-Regel angegeben sind.

Auf einer kleinen Site kann man ebensogut die folgende allgemeine Regel benutzen, um allen internen Verkehr an fremde Adressen durch Masquerading zu behandeln, nicht nur TCP-Pakete. Alle Pakete, die nach den `output`-Regeln für das externe Interface die Maschine verlassen dürften, werden dann auch von anderen internen Maschinen weitergeleitet:

```
ipchains -A forward -i $EXTERNAL_INTERFACE -s $INTERNAL_LAN_ADDRESSES -j  
MASQ
```

5.4 Die Firewall-Regeln mit Einzelpaketen testen

Sie können für einzelne Pakete prüfen, ob sie die Firewall-Regeln passieren würden, indem Sie die Option `-C` verwenden. `ipchains` meldet dann auf der Standardausgabe, was mit dem Paket auf Basis der aktuell gültigen Firewall-Regeln passieren würde: ob es angenommen (ACCEPT), mit (REJECT) oder ohne Fehlermeldung abgewiesen (DENY), durch Masquerading umgeschrieben wird (MASQ) oder Ähnliches.

Die Option `-C` (anstelle von `-I` für »Regel am Anfang einfügen« oder `-A` für »Regel ans Ende anhängen«) teilt `ipchains` mit, dass Sie ein Paket beschreiben und wissen möchten, was die momentanen Regeln mit solch einem Paket tun würden.

Ob Sie eine tatsächliche neue Regel oder nur ein Testpaket beschreiben, macht einige wichtige Unterschiede in der `ipchains`-Syntax aus. Die Option `-C` sieht Ihnen absolut nichts nach. Diese Unterschiede können recht verwirrend sein, weil vollkommen normale und erlaubte Regeldefinitionen mit `-C` plötzlich illegal werden.

`ipchains` kennt bei der Verwendung von `-C` keinerlei Default-Werte, sondern Sie stellen wirklich alles über Kommandozeilenoptionen ein. Die Beschreibung des

Testpaketes muss ein Interface über die Option `-i` enthalten, dazu je einen Absender- und Empfänger-Port (Portbereiche sind nicht erlaubt). In den folgenden Beispielen habe ich daher `$UNPRIVPORTS` durch einen einzelnen Port aus diesem Bereich – 5000 – ersetzt. Weil Absender- und Empfänger-Ports benötigt werden, müssen Sie auch eine Absender- und eine Empfängeradresse angeben. Der Negationsoperand `!` ist nicht gestattet. Die Option `-y` können Sie verwenden, nicht aber `! -y`. Immerhin sind für Adressen und Ports sowohl numerische als auch symbolische Namen akzeptabel.

Die folgenden Beispiele für `ipchains -C` gehen davon aus, dass Sie eine Firewall wie aus Kapitel 3 installiert haben.

Ein im nächsten Befehl beschriebenes Paket wird verworfen, selbst wenn Sie einen Webserver betreiben: Die empfohlene Regelsammlung gegen Adressfälschung erlaubt keine ankommenden Pakete von Ihrer eigenen Adresse.

```
> ipchains -C input -i eth0 -p tcp -y \  
    -s meine.IP.Adresse 5000 \  
    -d meine.IP.Adresse 80
```

denied

Ein Paket, auf das die folgende Beschreibung passt, wird von einem Computer mit Webserver angenommen – es ist Teil einer laufenden Verbindung zwischen einem fremden Web-Client und Ihrem Server:

```
> ipchains -C output -i eth0 -p tcp \  
    -s meine.IP.Adresse 80 \  
    -d any/0 5000
```

accepted

Abgehende Pakete entsprechend der folgenden Beschreibung werden ebenfalls verworfen, aber mit einer ICMP-Fehlermeldung. Die Firewallbeispiele aus unserem Buch werfen ankommende Pakete kommentarlos (`DENY`), schicken zu abgehenden Paketen aber eine Fehlermeldung (werfen sie mit `REJECT`). Die Option `-y` steht für das SYN-Flag und bedeutet, dass die Verbindung von Ihrem Webserver initiiert wird. Ihr Server darf aber nicht aktiv eine Verbindung zu einem fremden Client aufbauen.

```
> ipchains -C output -i eth0 -p tcp -y \  
    -s meine.IP.Adresse 80 \  
    -d any/0 5000
```

rejected

Das nächste Paketbeispiel wird durch Masquerading umgeschrieben. Alle Pakete von Computern im LAN an anderweitig nicht auflösbare Adressen (d.h. fremde Adressen) werden maskiert und über das externe Interface weitergeleitet.

```
> ipchains -C forward -i eth0 -p tcp \  
    -s meine.LAN.IP.Adresse 5000 \  
    -d any/0 80
```

masqueraded

Dieses letzte Beispiel zeigt einen weiteren Syntax-Unterschied zwischen dem normalen `ipchains`-Gebrauch und dem Prüfmodus über `-C`: `-j MASQ` ist nicht erlaubt. Lediglich die `forward`-Chain wird angegeben.

5.5 Suche nach offenen Ports

Das beste Tool für die Suche nach offenen Ports ist `ipchains -L` zur Anzeige der Firewall-Regeln. »Offen« sind alle Ports, bei denen die Firewall-Regel `ACCEPT` enthält. Die Red Hat-Distribution enthält in der Version 6.0 noch keine weiteren Utilities – außer `netstat` –, mit denen Sie nach offenen Ports suchen könnten. Allerdings finden Sie auf dem Internet eine Fülle an entsprechenden Programmen.

`netstat` ist recht vielseitig. Im nächsten Abschnitt verwenden wir es für die Suche nach offenen Ports und überprüfen so die TCP- und UDP-Ports aus unserer Firewalldefinition. In Kapitel 6 wird `netstat` für etwas andere Zwecke benutzt.

Anschließend stelle ich noch zwei weitere Portscanner vor, `strobe` und `nmap`.

5.5.1 netstat -a [-n -p -A inet]

`netstat` kann viele verschiedene Arten von Statusinformationen anzeigen. Zahlreiche Kommandozeilenoptionen wählen aus, welche davon tatsächlich gerade erwünscht sind. Die folgenden Optionen sind für die Suche nach offenen Ports nützlich, für die Anzeige, ob sie gerade benutzt werden und von wem, sowie für die Auflistung der Programme und Prozesse, die auf diesen Ports auf Anfragen warten:

- `-a` zeigt alle Ports an, ob sie jetzt gerade aktiv benutzt werden oder ob nur ein lokaler Server auf eine neue Anfrage wartet.
- `-n` stellt für Hostnamen und Portbezeichnungen das numerische Format ein. Ohne diese Option werden Rechnernamen und Ports symbolisch angezeigt, oder zumindest soviel davon, wie in 80 Zeilen Terminalbreite passt. `-n` erspart Ihnen manchmal eine lange Wartezeit auf die DNS-Namensauflösung. Ohne `-n` ist das Ergebnis etwas besser lesbar.
- `-p` bewirkt, dass `netstat` auch die Programmnamen ausgibt, die den entsprechenden Sockets zugeordnet sind. Nur `root` darf diese Option benutzen.
- `-A inet` bezieht sich auf die Adressfamilie und zeigt nur die Ports an, die mit einer Netzwerkkarte assoziiert sind. Sockets aus der lokalen Unix-Adressfamilie einschließlich lokaler netzwerkbasierter Verbindungen werden ignoriert, z.B. von laufenden X-Windows-Programmen.

Hinweis

Sockettypen: TCP/IP und UNIX

Linux ist ein BSD-Unix-Derivat. BSD führte Sockets 1986 mit der Version 4.3 ein. Die wichtigsten Sockettypen sind die Internet-Domain (AF_INET) und die UNIX-Domain (AF_UNIX). AF_INET verwendet TCP/IP über das Netzwerk; AF_UNIX ist ein rein lokaler Sockettyp. Diese UNIX-Domain-Sockets werden bei der Kommunikation zwischen zwei Prozessen auf dem gleichen Computer eingesetzt. Sie sind bei lokalen Sockets deutlich effizienter als TCP/IP; über das Netzwerk wird bei ihnen nichts übertragen.

Der folgende `netstat`-Befehl bezieht nur Internet-Domain-Sockets mit ein. Alle Ports, denen ein Netzwerkservice zugeordnet ist, werden mit Programmname und Prozess-ID aufgeführt.

```
> netstat -a -p -A inet
```

```
1. Active Internet connections (servers and established)
2. Proto Recv-Q Send-Q Local Address   Foreign Address State    PID/
   Program name

3. tcp      0    143 internal:telnet  macintosh:62360 ESTABLISHED
   15392/in.telnetd
4. tcp      0      0 *:smtp          *:.*             LISTEN   3674/sendmail: acce
5. tcp      0      0 my.host.domain:www *:.*             LISTEN   638/httpd
6. tcp      0      0 internal:domain *:.*             LISTEN   588/named
7. tcp      0      0 localhost:domain *:.*             LISTEN   588/named
8. tcp      0      0 *:auth          *:.*             LISTEN   574/inetd
9. tcp      0      0 *:pop-3         *:.*             LISTEN   574/inetd
10. tcp     0      0 *:telnet        *:.*             LISTEN   574/inetd
11. tcp     0      0 *:ftp           *:.*             LISTEN   574/inetd
12. udp     0      0 *:domain        *:.*             588/named
13. udp     0      0 internal:domain *:.*             588/named
14. udp     0      0 localhost:domain *:.*             588/named
```

Zeile 1 stellt klar, dass lokale Server sowie bestehende Internet-Verbindungen aufgeführt sind. Diese Auswahl ist durch die `netstat`-Option `-A inet` bedingt.

Zeile 2 enthält die folgenden Spaltenüberschriften:

- `Proto` steht für das Transportprotokoll: TCP oder UDP.
- `Recv-Q` gibt an, wieviele Bytes bereits vom fremden Rechner empfangen, aber noch nicht an das lokale Programm ausgeliefert wurden.
- `Send-Q` gibt an, wieviele Bytes bereits vom lokalen Programm gesendet wurden, deren Empfang der fremde Rechner noch nicht bestätigt hat.

- **Local Address:** Das lokale Socket mit Netzwerkinterface und Port.
- **Foreign Address:** Das fremde Socket mit Netzwerkinterface und Port.
- **State** bezeichnet bei TCP-Verbindungen den Zustand des Sockets: ESTABLISHED bzw. für bestehende Verbindungen, LISTEN für einen Server, der auf Anfragen wartet, sowie eine Reihe von Zwischenstadien für Verbindungsauf- und -abbau.
- **PID/Program name** bezieht sich auf den Prozess, dem das lokale Socket gehört.

Zeile 3 zeigt eine bestehende telnet-Verbindung über das interne Netzwerkinterface zu dem Computer macintosh. Über diese Verbindung wurde der netstat-Befehl eingegeben.

Zeile 4 zeigt sendmail, das auf dem SMTP-Port aller Interfaces auf ankommende Mail wartet – einschließlich dem externen Interface ins Internet, dem internen LAN-Interface und dem Loopback-Interface für den lokalen Rechner.

Zeile 5 zeigt einen lokalen Webserver, der auf dem externen Internet-Interface auf Verbindungen wartet.

Zeile 6 zeigt einen Nameserver, der auf dem internen LAN-Interface auf DNS-Anfragen über das TCP-Protokoll wartet.

Zeile 7 zeigt noch einmal den Nameserver, der auch auf dem Loopback-Interface auf DNS-Anfragen über TCP wartet.

Zeile 8 zeigt, dass inetd auf dem auth-Port (für identd) auf Verbindungen wartet.

Zeile 9 zeigt, dass inetd auf dem pop-3-Port (für popd) auf Verbindungen wartet. Sobald jemand eine Anfrage schickt, startet inetd einen popd-Server, der sie beantwortet. inetd bedient hier alle Interfaces. Sowohl die Firewall als auch die Sicherheitsmechanismen auf den höheren Ebenen – tcp_wrapper und popd-Konfiguration – begrenzen die tatsächlich erlaubten Verbindungen auf die Computer im LAN.

Zeile 10 zeigt, dass inetd auf dem telnet-Port (für telnetd) auf Verbindungen wartet. Sobald jemand eine Anfrage schickt, startet inetd einen telnetd-Server, der sie beantwortet. inetd bedient hier alle Interfaces. Sowohl die Firewall als auch die Sicherheitsmechanismen auf den höheren Ebenen (tcp_wrapper) begrenzen die tatsächlich erlaubten Verbindungen auf die Computer im LAN.

Zeile 11 zeigt, dass inetd auf dem ftp-Port (für ftpd) auf Verbindungen wartet. Sobald jemand eine Anfrage schickt, startet inetd einen ftpd-Server, der sie beantwortet. inetd bedient hier alle Interfaces. Sowohl die Firewall als auch die Sicherheitsmechanismen auf den höheren Ebenen (tcp_wrapper und ftpd-Konfiguration) begrenzen die tatsächlich erlaubten Verbindungen auf die Computer im LAN.

Zeilen 12 bis 14 zeigen den Nameserver, der auf allen Interfaces auf UDP-Verbindungen von anderen Nameservern sowie auf Anfragen von Clients wartet.

Hinweis

Anzeigekonventionen von netstat

In der Ausgabe von netstat sind lokale und fremde Adressen in der Form Adresse:Port aufgeführt.

*In der Spalte Local Address bezieht sich die angegebene Adresse (bzw. auch der Hostname, falls -n nicht mit angegeben war) auf eines Ihrer Netzwerkinterfaces. Wenn die Adresse als Sternchen * angezeigt wird, hat der Server alle Netzwerkinterfaces belegt, nicht nur eines. Der Port ist einfach der numerische oder symbolische Port des Servers.*

*Bei den fremden Adressen (Spalte Foreign Address) ist die Adresse der Name bzw. die IP-Adresse des Kommunikationspartners, bzw. *, wenn ein Server auf Verbindungen wartet. Der Port bezieht sich in dieser Spalte auf das fremde Ende der Verbindung.*

Bei Servern, die auf Verbindungen über das TCP-Protokoll warten, steht in der State-Spalte LISTEN. Ist das verwendete Protokoll hingegen UDP, so ist der Tabelleneintrag leer. UDP kennt keine Verbindungszustände; netstat unterscheidet hier einfach zwischen TCP, bei dem es die Verbindungszustände anzeigt, und UDP, das sie eben nicht kennt.

Die nächsten zwei Abschnitte beschreiben zwei Tools, die Sie aus dem Internet beziehen können: `strobe` und `nmap`.

5.5.2 strobe

`strobe` ist ein simpler TCP-Portscanner. Er zeigt Ihnen an, welche Ports auf Ihren Netzwerkinterfaces offen sind. `strobe` erhalten Sie von <http://metalab.unc.edu/pub/Linux/system/network/admin/>.

Das folgende Beispiel zeigt, welche TCP-Ports `strobe` auf einem System gefunden hat. In der Voreinstellung zeigt `strobe` auch den Namen des überprüften Computers sowie die zugehörigen Portbeschreibungen aus `/etc/services` an. Diese Liste ist nicht unbedingt vollständig: Wenn eine Firewall installiert ist, können durchaus noch weitere Server auf der Maschine laufen, die aber durch die Firewall gesperrt sind.

```
> strobe firewall
```

```
strobe 1.02 (c) 1995 Julian Assange -Proff- (proff@suburbia.apana.org.au).
firewall  ssh          22/tcp          # SSH Remote Login Proto-
                                         col
firewall  smtp          25/tcp mail
```

```
firewall    domain    53/tcp nameserver    # name-domain server
firewall    www       80/tcp http         # WorldWideWeb HTTP
firewall    auth     113/tcp authentication tap ident
```

5.5.3 nmap

nmap ist ein neueres Sicherheitstool, das viele der heute benutzten Scantechniken beherrscht. Sie sollten Ihre Systeme mit *nmap* abchecken – andere werden das ganz gewiss tun. *nmap* finden Sie z.B. unter <http://metalab.unc.edu/pub/Linux/system/network/admin/>.

Im folgenden Beispiel berichtet *nmap* über den Zustand aller TCP- und UDP-Ports. Weil keine ausführliche Ausgabe angefordert wurde, tauchen hier nur die offenen Ports auf, auf denen also Server laufen. Die Anzeige von *nmap* umfasst neben dem gescannten Hostnamen und der IP-Adresse die einzelnen Ports mit ihrem jeweiligen Zustand (offen oder geschlossen), dem Transportprotokoll und dem symbolischen Portnamen aus */etc/services*. Weil in unserem Beispiel der Rechner *sebastion* auf dem internen Netzwerk liegt, sehen wir auch *telnet*- und *X11*-Ports für den Zugriff über das LAN:

```
> nmap -sT -sU sebastion
```

```
WARNING: -sU is now UDP scan - for TCP FIN scan use -sF
```

```
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on sebastion.firewall.lan (192.168.1.2):
```

Port	State	Protocol	Service
21	open	tcp	ftp
22	open	tcp	ssh
23	open	tcp	telnet
25	open	tcp	smtp
53	open	udp	domain
53	open	tcp	domain
113	open	tcp	auth
123	open	udp	ntp
6000	open	tcp	X11

```
nmap run completed - 1 IP address (1 host up) scanned in 3 seconds
```

5.6 Fehlersuche für ssh – ein Beispiel aus der Praxis

Vor einer ganzen Weile legte ein Freund mir nahe, wie wichtig es sei, statt *telnet* doch lieber *ssh* zu verwenden. Ich hatte damals die himmlischen Freuden meiner Firewall-Bibel noch nicht entdeckt (D. Brent Chapman und Elizabeth D. Zwicky: *Building Internet Firewalls*. O'Reilly & Associates, 1995), ebenso wenig die RFCs (Requests for Comment) von www.ietf.cnri.reston.va.us/rfc.html. Wenn es für

eine Firewall-Regel kein Beispiel aus den Linux-HOWTOs gab, musste ich das Protokoll über Versuch und Irrtum kennen lernen, indem ich die verfügbaren Tools benutzte und die Protokollierung einschaltete. `ssh` gehörte zu meinen ersten Übungsstückchen.

Ich hatte `ssh` also heruntergeladen, übersetzt und installiert. Daheim und bei einem meiner Internet-Provider hatte ich die kryptografischen Schlüssel für die Authentifizierung installiert. Abgehende `ssh`-Verbindungen zum Provider funktionierten ohne Probleme. Ankommende `ssh`-Verbindungen vom Provider ebenso. Alles war wunderbar.

Schnell merkte ich, dass ankommende `ssh`-Verbindungen aus meiner Arbeitsstelle nicht möglich waren. Wenn ich von der Arbeit `ssh` nach Hause versuchte, erhielt ich die Aufforderung zur Passwortheingabe, danach die Begrüßungsmeldung meines Rechners und einen Shell-Prompt. Danach war Schluss. Die Tastatur reagierte nicht mehr. Wenn ich stattdessen von der Arbeit eine `ssh`-Sitzung zu meinem Internet-Provider aufbaute und mich von dort aus daheim anmeldete, funktionierte alles wie erwartet. Ich konnte mich vom ISP aus daheim einloggen und von der Arbeit aus am ISP, aber nicht direkt von der Arbeit aus daheim.

Ich stellte also eine `ssh`-Verbindung zu meinem Internet-Provider her und von dort eine nach Hause. Dann untersuchte ich mit `ps` und `netstat`, wo der Unterschied zwischen der Verbindung vom Provider und der von der Arbeit lag.

Ich kannte das `ssh` damals nicht und verstand nicht, was ich sah. Tatsächlich lieferte ich eine komplette Missinterpretation der Geschehnisse, aber es gelang mir trotzdem, eine brauchbare Firewall-Regel aufzustellen.

Mein erster Gedanke war, dass der lokale `sshd`-Server vielleicht abstürzte. `ps -ax` zeigte drei Kopien des `sshd`-Servers: der Master-Server, der Server für die laufende Verbindung vom Provider sowie der für die Verbindung von der Arbeit, für die abgestürzte Verbindung also. Okay – die Server waren alle da.

Im nächsten Schritt sah ich mir mit `netstat -a -A inet` die Verbindungszustände an. Der Master-Server wartete wie vermutet auf Port 22 auf neue Verbindungen. Der Server für die Verbindung vom Provider benutzte den Port 22, während der zugehörige Client einen unprivilegierten Port verwendete – ebenfalls genau wie erwartet. Der Server für die Verbindung von der Arbeit benutzte Port 22, aber der zugehörige Client den privilegierten Port 1023! Das war verwirrend. Der Client in der Arbeit lief auf einem privilegierten Port, der vom Internet-Provider auf einem unprivilegierten.

Also was jetzt? In `/etc/services` erscheint `ssh` als ein ganz normaler TCP-Dienst, der auf Port 22 läuft. Wenn das so wäre, müsste das folgende Firewall-Regelpaar für ankommende `ssh`-Verbindungen funktionieren:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 22 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 22 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

Genau diese Regeln waren für die Verbindung vom Provider ja auch genug. Die Regeln waren schon richtig, aber die Verbindung aus der Arbeit blieb nach dem Login hängen. Die beiden angeführten Regeln erlaubten mir also auch aus der Arbeit den Login mit Begrüßungsmeldung und Shell-Prompt – aber sie reichten offensichtlich nicht aus.

Ich wollte die Sache noch einmal prüfen und schaltete das Logging für alle ankommenden und abgehenden Pakete zu und von Port 22 ein, egal, ob sie angenommen oder abgelehnt wurden. Dabei nahm meine Firewall alle abgehenden Pakete von Port 22 an, sowohl zu unprivilegierten wie auch zu privilegierten Ports. Ankommende Pakete wurden von unprivilegierten Ports akzeptiert, von privilegierten (also z.B. auch von Port 1023) hingegen nicht. Das erklärte am Ende, warum der `ssh`-Login aus der Arbeit bis zur Begrüßungsmeldung und zum Shell-Prompt kam.

Zu diesem Zeitpunkt waren meine Firewall-Regeln ziemlich primitiv. Die Firewall-Policy für ankommende Pakete war `DENY`, die für abgehende Pakete `ACCEPT`. Im Vergleich zu heute ließ ich also viel mehr Pakete nach außen passieren. Die Regeln für ein typisches TCP-Protokoll waren für die Verbindung vom Provider völlig ausreichend. Was die Verbindung von der Arbeit angeht, war die Kombination aus `DENY` für `input` und `ACCEPT` für `output` verwirrend. Meine Firewall erlaubte die abgehenden Pakete von meinem Server an den privilegierten Port, nicht aber die ankommenden Pakete vom zugehörigen Client. Der Server konnte daher die Begrüßungsmeldung und den Shell-Prompt schicken, der Client konnte aber nicht mehr darauf antworten.

Weil ich nie mehr als ein oder zwei ankommende `ssh`-Verbindungen auf einmal sah, dachte ich, dass `ssh` – für die Verbindung zu meiner Arbeitsstelle – die Verbindung zwischen einem unprivilegierten Port und 22 herstellte und dann für den eigentlichen Datenaustausch auf eine neue Verbindung zwischen Port 1022 oder 1023 und Port 22 umstellte. Natürlich kann sich der Server den Port auf Client-seite nicht aussuchen; der Client wählt seinen Port selbst, ob unprivilegiert oder privilegiert.

Erst viel später las ich im Quellcode zu `ssh` nach. In Abhängigkeit von der lokalen Konfiguration fängt der `ssh`-Client beim Port 1023 an und sucht dann bis hinunter zu Nummer 513 nach dem ersten freien Port, dem er die Verbindung zuweist. Werden andere Einstellungen benutzt, verwendet der Client einen unprivilegierten Port. Der Server akzeptiert die Verbindungen in beiden Fällen. Wir brauchen also noch zwei zusätzliche Regeln:

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE 1022:1023 \  
-d $IPADDR 22 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 22 \  
-d $ANYWHERE 1022:1023 -j ACCEPT
```

In diesem Fall war die Protokollierung aller Pakete an und von Port 22 – egal, ob angenommen oder abgewiesen – schon genug, um das Problem einzugrenzen und durch entsprechende Regeln zu beheben. Erst viel später verstand ich, wofür diese Regeln eigentlich gut waren.

5.7 Zusammenfassung

Dieses Kapitel stellte die `ipchains`-Befehle zum Anzeigen von Regeln vor, die Informationen über Ports und Netzwerkdämonen, die `netstat` liefert, dazu noch ein paar ergänzende Tools, mit denen man die korrekte Installation und Reihenfolge von Firewall-Regeln überprüfen kann. Auch die `ipchains`-Option, einen bestimmten Pakettyp probeweise durch die installierten Regeln laufen zu lassen und sich das Ergebnis anzusehen, wurde demonstriert.

Dieses Kapitel betont die Firewall-Regeln und die durch sie geschützten Ports. Das nächste Kapitel konzentriert sich darauf, welche Server-Software diese Ports eigentlich benutzt, und untersucht darüber hinaus die Informationen zu Firewall, System und Servern, die in den Logdateien abgelegt werden.



Teil III

Systemsicherheit und Systemüberwachung

- Kapitel 6 Läuft das System wie erwartet?
- Kapitel 7 Ergänzende Maßnahmen durch die Unix-Systemadministration
- Kapitel 8 Entdecken von Eindringlingen und Melden der Vorfälle

Kapitel 6

Läuft das System wie erwartet?



- 6.1 Überprüfen der Netzwerkschnittstellen mit ifconfig
- 6.2 Überprüfen der Netzwerkverbindung mit ping
- 6.3 Überprüfen der Netzwerkprozesse mit netstat
- 6.4 Überprüfen aller laufenden Prozesse mit ps ax
- 6.5 Die Protokolldateien Ihres Systems
- 6.6 Zusammenfassung

Kapitel 5 hatte gezeigt, wie Sie mit *ipchains* als diagnostischem Tool die Firewall-Regeln und die durch sie geschützten Ports überprüfen – um sicherzugehen, dass die Regeln aktiv sind und richtig funktionieren. In diesem Kapitel geht es vor allem darum, welche Server-Software diese Ports benutzt. Es dürfen nur die Programme und Dienste aktiv sein, von denen Sie das auch wollen. Nachdem Sie die laufenden Programme und Ports geprüft haben, sehen Sie sich an, was Ihnen diese Software in den Status- und Fehlermeldungen aus den Logdateien mitteilt. In diesem Kapitel werden zusätzliche administrative Hilfsmittel und Protokollinformationen vorgestellt.

Sie können sich niemals absolut sicher sein, dass Ihr Unix-System wirklich korrekt läuft. Es gibt immer nur eine gewisse Wahrscheinlichkeit dafür, wenn nämlich der Rechner genau das tut, was Sie von ihm erwarten. Unix-Systeme sind viel zu komplex, die Konfiguration ist zu vielfältig, als dass es absolute Sicherheit geben könnte. Die Protokolldateien des Systems helfen bei der Prüfung, ob alles so ist wie gewollt, und welche Vorkommnisse vom Erwarteten abweichen.

6.1 Überprüfen der Netzwerkschnittstellen mit *ifconfig*

Die Hauptaufgabe von *ifconfig* ist die Konfiguration und Aktivierung der Netzwerkschnittstellen. Es wird von einem der Boot-Skripten für das Netzwerk aufgerufen, die während des Bootvorgangs unter der Kontrolle des Runlevel-Managers stehen. Später eignet sich *ifconfig* auch als Hilfsmittel bei der Fehlersuche, wobei es den Zustand der Netzwerkschnittstellen anzeigen kann.

Wenn Sie *ifconfig* alleine aufrufen, ohne Optionen, erhalten Sie eine Übersicht aller aktiven Netzwerkinterfaces. Die Option *-a* schließt alle Interfaces ein, ob aktiv oder nicht. Wenn alle existierenden Interfaces aktiviert sind, sehen beide Übersichten identisch aus.

Wenn ein Interface entgegen Ihrer Erwartung inaktiv ist, deutet das normalerweise auf irgendein Konfigurationsproblem hin. Überprüfen Sie die Einstellungen unter Red Hat mit *control-panel* oder *linuxconf*, unter SuSE mit *YaST*.

Die folgende Übersicht stammt von einem Computer mit nur einer Netzwerkkarte. *ifconfig* berichtet über den Zustand der physikalischen Schnittstelle *eth0* sowie des Loopback-Interfaces *lo*:

```
> ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 00:A0:CC:40:9B:A8
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:266027 errors:0 dropped:0 overruns:0 frame:0
          TX packets:202290 errors:0 dropped:0 overruns:0 carrier:0
          collisions:17805 txqueuelen:100
          Interrupt:9 Base address:0xec00
```



```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:3924  Metric:1
        RX packets:51997 errors:0 dropped:0 overruns:0 frame:0
        TX packets:51997 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
```

Falls es Schwierigkeiten gibt, wollen Sie zuerst einmal wissen, ob das Interface noch aktiv ist. Die dritte Zeile jedes Blocks enthält die entsprechende Statusmeldung. Im Beispiel ist z.B. eth0 aktiv oder UP.

Wie Sie sehen, liefert `ifconfig` noch viele andere Informationen, die Sie später vielleicht einmal brauchen, darunter die MAC-Hardware-Adresse der Ethernetkarte (`Hwaddr`), die IP-Adresse (`inetaddr`), die Broadcast-Adresse (`Bcast`) und die Netmask (`Mask`). Daneben erscheinen in der Ausgabe auch weniger hilfreiche Infos wie z.B. die maximale Framegröße (`MTU`) sowie die Zahl der bislang empfangenen (`RX`) und gesendeten (`TX`) Pakete.

Hinweis

Maximum Transmission Unit MTU und Fragmentierung

Die MTU lassen Sie am besten bei der Voreinstellung von 1500. Die meisten Netzwerke sind heute Ethernets, und die MTU ist im Ethernet 1500 Byte. Bei anderen Protokollen der Sicherungsschicht (z.B. ATM oder Token Ring) werden höhere Einstellungen für die MTU empfohlen. Das garantiert Ihnen aber eine Fragmentierung Ihrer Pakete, irgendwo unterwegs, wenn das Paket eine Netzwerkgrenze überschreitet. Beim TCP-Verbindungs Aufbau wird die MTU ausgehandelt.

Eine höhere MTU spart Bandbreite ein: Zu jedem Paket muss ja auch ein Header versandt werden, und bei einer größeren MTU passen mehr Daten in ein Paket, kommt das System also mit weniger Paketen aus. Bandbreite ist auf den Backbones der heutigen Netze aber ein viel geringeres Problem als Fragmentierung. Die für die korrekte Behandlung fragmentierter Pakete notwendige Rechenzeit ist ein sehr wesentlicher Faktor in der Gesamt-Performance. Mit vielen kleinen Paketen erhalten Sie in der Regel eine bessere Performance als mit wenigen großen Paketen, die unterwegs fragmentiert werden.

6.2 Überprüfen der Netzwerkverbindung mit ping

Für grundlegende Connectivity-Tests ist `ping` ein optimales Werkzeug. Wenn das externe Netzwerkkarte aktiv ist, aber Sie keine Verbindung zu einem anderen Rechner herstellen können, zeigt `ping` Ihnen, ob Pakete über das Interface laufen. Natürlich müssen die Firewall-Regeln `ping` zulassen. Eine fehlende Antwort auf

ping beweist noch nicht, dass der betreffende Rechner nicht erreichbar ist. Möglicherweise reagiert er bewusst nicht auf ICMP-echo-requests. Wenn Sie aber eine Antwort auf Ihr ping erhalten, wissen Sie mit Sicherheit, dass Pakete übertragen werden und der andere Rechner reagiert.

Hinweis***Auf ping-Anfragen reagieren?***

Auch wenn ein Computer läuft, muss er nicht unbedingt auf ping-Anfragen aus dem Internet reagieren. So grundlegend und einfach ping ist, so alt ist auch seine Geschichte in Denial-of-Service-Angriffen. Aus diesem Grund reagieren die Firewall-Beispiele aus diesem Buch nur auf pings von den Geräten des Internet-Providers, der evtl. interne Netzwerkanalysen betreiben möchte.

Wenn Sie nur einen Hostnamen oder eine IP-Adresse als Argument angeben, schickt ping Pakete in einer Endlosschleife, bis Sie den Prozess per Signal anhalten. Dann fasst ping die Ergebnisse zusammen:

```
> ping smtp.mein.isp.domain
```

```
PING smtp.mein.isp.domain (10.10.22.85): 56 data bytes
64 bytes from 10.10.22.85: icmp_seq=0 ttl=253 time=4.2 ms
64 bytes from 10.10.22.85: icmp_seq=1 ttl=253 time=4.4 ms
64 bytes from 10.10.22.85: icmp_seq=2 ttl=253 time=4.1 ms
64 bytes from 10.10.22.85: icmp_seq=3 ttl=253 time=5.4 ms
64 bytes from 10.10.22.85: icmp_seq=4 ttl=253 time=3.9 ms
```

```
> ^C
```

```
--- smtp.mein.isp.domain ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 3.9/4.4/5.4 ms
```

Wenn Sie ein System anpingen können, heißt das nicht in jedem Fall, dass die von ihm angebotenen Dienste laufen und Ihnen zur Verfügung stehen. Es bedeutet lediglich, dass ein ping-Paket über das Netzwerk und wieder zurück geschickt worden ist. Wenn ein bestimmter Netzwerkdienst nicht funktioniert oder eine Maschine nicht reagiert, wissen Sie damit zumindest, dass die Netzwerkverbindung funktioniert.

Aus unserem Beispiel entnehmen Sie, dass der Computer mit dem Mailserver Ihres Internet-Providers eingeschaltet ist. Sie wissen noch nicht, ob die Mailserver-Software läuft. Wenn Sie also keine Mail verschicken, aber den Mailserver anpingen können, ist der entsprechende Computer zumindest online. Möglicherweise läuft das Mailserver-Programm nicht.

6.3 Überprüfen der Netzwerkprozesse mit netstat

In Kapitel 5 hatten wir uns mit `netstat -a -p -A inet` angesehen, welche Programme auf welchen Netzwerkschnittstellen, welchen Protokollen (TCP oder UDP) und welchen Ports laufen. Die Option `-A inet` hatte die Ergebnisse auf Dienste und Ports begrenzt, die mit Netzwerkkommunikation zu tun hatten. Ohne die Option `-A` berichtet `netstat` sowohl über INET- als auch über UNIX-Domain-Sockets.

`netstat -a -p -A inet` zeigt die entsprechenden Informationen für die aktiven Unix-Domain-Sockets an, also für die Sockets der lokalen Dienste. Sie sollten jede Zeile der folgenden Auflistung erklären können:

```
> netstat -a -p -A unix
```

```
1. Active UNIX domain sockets (servers and established)
2. Proto RefCnt Flags   Type   State      I-Node PID/Program name Path
3.  unix  1      [ ]    STREAM CONNECTED  938    588/named      @0000007b
4.  unix  0      [ ACC ] STREAM LISTENING 500119 521/syslogd    /dev/log
5.  unix  1      [ ]    STREAM CONNECTED  864    532/klogd      @00000075
6.  unix  0      [ ACC ] STREAM LISTENING 1083    661/xfs        /tmp/.font-
    unix/fs-1
7.  unix  0      [ ACC ] STREAM LISTENING 939     588/named      /var/run/
    ndc
8.  unix  1      [ ]    STREAM CONNECTED 492327 3674/sendmail  @00000722
9.  unix  0      [ ]    STREAM CONNECTED 129     1/init        @00000016
10. unix  1      [ ]    STREAM CONNECTED 1349    893/sshd      @0000008a
11. unix  1      [ ]    STREAM CONNECTED 549784 521/syslogd    /dev/log
12. unix  1      [ ]    STREAM CONNECTED 492328 521/syslogd    /dev/log
13. unix  1      [ ]    STREAM CONNECTED 1350    521/syslogd    /dev/log
14. unix  1      [ ]    STREAM CONNECTED 953     521/syslogd    /dev/log
15. unix  1      [ ]    STREAM CONNECTED 865     521/syslogd    /dev/log
```

Zeile 1 schränkt den Report auf wartende und aktive Unix-Domain-Sockets von lokalen Servern ein.

Zeile 2 enthält die folgenden Spaltenüberschriften:

- **Proto** bezieht sich auf das Transportprotokoll, über das der jeweilige Dienst läuft. Hier steht immer `unix`.
- **RefCnt** (»Reference Count«) gibt an, wie viele Prozesse mit diesem Socket verbunden sind.
- **Flags** kann `ACC` enthalten oder leer bleiben. `ACC` (»Accept«) bedeutet, dass ein Prozess auf eine ankommende Verbindung wartet.
- **Type** enthält die benutzte Socket-Art. Mögliche Werte sind `STREAM` für eine Verbindung analog TCP. Ein `DGRAM`-Socket entspräche wie UDP einem verbindungslosen Protokoll.

- *State* ist der Zustand der Verbindung: bestehende Verbindung (ESTABLISHED), Warten auf Verbindung (LISTEN) sowie eine Reihe von Zwischenstadien des Verbindungsauf- und -abbaus (z.B. SYN SENT, SYN RECV, FIN WAIT, FIN SENT).
- Die *I-Node-Nummer* bezeichnet das Dateisystemobjekt, mit dem das Programm die Verbindung zum jeweiligen Socket hergestellt hat. Ein *inode* identifiziert unter Linux eine Datei.
- Die Spalte *PID/Program name* enthält Nummer und Name des Prozesses, zu dem das Socket gehört.
- *Path* ist der Pfad zu dem Objekt, über das die Verbindung zum Socket hergestellt wurde.

Zeile 3 zeigt den Nameserver *named*, der aktuell mit einem zweiten internen Nameserver verbunden ist.

Die Zeilen 4 und 11–15 zeigen den Protokolldämon *syslogd*, der momentan sechs Sockets benutzt. Auf dem Computer, von dem dieses Beispiel stammt, erzeugt das *syslog* sechs verschiedene Protokolldateien. Aus der *netstat*-Anzeige geht allerdings nicht hervor, welches Socket zu welcher Logdatei gehört. Die Konfiguration des *syslogs* wird zu einem späteren Zeitpunkt besprochen.

Zeile 6 zeigt den Protokolldämon für den Kernel, *klogd*, der über */proc/kmsg* auf Nachrichten aus den Kernel-Nachrichtenspeichern wartet. *klogd* arbeitet eng mit *syslogd* zusammen.

Zeile 6 zeigt *xfs*, der lokale Anfragen nach X-Windows-Fonts beantwortet.

Zeile 7 zeigt den Nameserver *named*.

Zeile 8 zeigt *sendmail*, das für den Empfang und Versand von E-Mail bereit ist.

Zeile 9 zeigt *init*, die Mutter aller Unix-Prozesse, die über dieses Socket Anfragen für den Neustart von Terminal-Sitzungen und für Änderungen des Runlevels entgegennimmt.

sshd aus Zeile 10 nimmt lokale *ssh*-Sitzungen an und startet jeweils einen eigenen Server.

Das System läuft in diesem Beispiel genau wie erwartet. *init* muss, *syslogd* und *klogd* sollten ständig laufen. Die übrigen Server – *named*, *xfs*, *sendmail* und *sshd* sind optional, auf diesem Computer aber beabsichtigt.

6.4 Überprüfen aller laufenden Prozesse mit *ps ax*

ps liefert eine Übersicht über den Prozessstatus. Die Option *a* wählt alle Prozesse mit zugeordnetem Terminal aus, typischerweise Programme Ihrer User, die im Vordergrund laufen. *x* selektiert zusätzlich alle Prozesse ohne Terminal, i.d.R. Dämonen, die automatisch gestartet werden und ständig im Hintergrund aktiv sind. Zusammen sorgen die beiden Optionen also für die Anzeige aller Prozesse

einschließlic Prozess-ID, zugeordnetem Terminal, Zustand, verbrauchter Rechenzeit und Programmname. Wenn Sie zusätzlich die Option *u* angeben, erhalten Sie darüber hinaus noch Infos im Zusammenhang mit den Benutzern, z.B. den Loginnamen.

`ps ax` zeigt Ihnen alle Programme, die gerade auf Ihrem Computer laufen. Wie bei `netstat` sollten Sie von jedem laufenden Prozess wissen, was er tut und warum es ihn gibt. Abgesehen von ein paar Systemdämonen – `init`, `kflushd`, `kpiod`, `kswapd`, `mdrecoveryd` und `mingetty` – müssten Sie alle weiteren Dämonen explizit über den Runlevel-Manager, in `/etc/inetd.conf` oder `/etc/rc.d/rc.local` eingeschaltet haben. Alle Programme, die darüber hinaus noch laufen, müssen eindeutig als Benutzerprozesse erkennbar sein. Vor allem darf `ps -ax` keine Prozesse anzeigen, die Sie nicht erwartet hätten.

Die folgende `ps -ax`-Ausgabe ist relativ typisch:

```
> ps -ax
```

1.	PID	TTY	STAT	TIME	COMMAND
2.	1	?	S	0:03	init
3.	2	?	SW	0:02	[kflushd]
4.	3	?	SW	0:00	[kpiod]
5.	4	?	SW	0:01	[kswapd]
6.	5	?	SW<	0:00	[mdrecoveryd]
7.	202	?	S	0:00	/sbin/dhccpd
					-c /etc/sysconfig/network-scripts/ifdhcp
8.	521	?	S	0:03	syslogd -m 0
9.	532	?	S	0:00	klogd
10.	546	?	S	0:00	/usr/sbin/atd
11.	560	?	S	0:01	cron
12.	574	?	S	0:01	inetd
13.	588	?	S	0:10	named
14.	638	?	S	0:02	httpd
15.	661	?	S	0:01	xfs
16.	893	?	S	0:07	/usr/local/sbin/sshd
17.	928	tty2	S	0:00	/sbin/mingetty tty2
18.	930	?	S	0:04	update (bdfush)
19.	1428	tty1	S	0:00	/sbin/mingetty tty1
20.	3674	?	S	0:00	sendmail: accepting connections on port 25
21.	17531	?	S	0:00	in.telnetd -n
22.	17532	pts/1	S	0:00	login -- bob
23.	17533	pts/1	S	0:00	-ksh
24.	17542	pts/1	R	0:00	ps -ax

Zeile 1 enthält die Spaltenüberschriften:

- PID ist die eindeutige Prozesskennung.
- TTY ist das dem Prozess zugeordnete Terminal, sofern eines existiert.

- **STAT** ist der Zustand des Prozesses. Im Beispiel läuft nur *ps* momentan (Runnable). Die anderen Prozesse schlafen (Sleeping), d.h. sie warten auf ein Ereignis, auf das sie reagieren sollen.
- **TIME** ist die gesamte Rechenzeit, die der Prozess bis jetzt verbraucht hat.
- **COMMAND** ist der Befehlsname des Prozesses.

Zeile 2 zeigt *init*, die Mutter aller Prozesse. Sie läuft immer. Die meisten Unix-Varianten stürzen ab, falls *init* nicht mehr arbeitet.

Zeile 3 zeigt *kflushd*: dieses ist kein Prozess, sondern ein Thread im Kernel, der die veränderten Puffer für das Dateisystem regelmäßig auf die Festplatte zurückschreibt.

Zeile 4 zeigt *kpiod*. Dabei handelt es sich ebenfalls um einen Kernelthread, nicht um einen Prozess. *kpiod* verwaltet das Paging.

Zeile 5 zeigt *kswapd*, ein weiterer Kernelthread, der bei Bedarf Speicheranteile aus dem Hauptspeicher auf Festplatte schreibt, um für andere Prozesse Platz zu schaffen.

Zeile 6 zeigt *mdrecoveryd*, noch ein Kernelthread, der die Zusammenfassung mehrerer Platten zu einer einzelnen Einheit (RAID) organisiert.

Zeile 7 zeigt *dhcpcd*, den DHCP-Client, der dynamische Adresszuweisungen von Ihrem DHCP-Server entgegennimmt.

Zeile 8 zeigt den Protokolldämonen *syslogd*, der Meldungen von Programmen annimmt und auf die einzelnen Logdateien, die Konsole, angeschlossene Terminals und so weiter verteilt.

Zeile 9 zeigt den Kernel-Protokolldämon *klogd*. Er überträgt Kernelnachrichten in Zusammenarbeit mit *syslogd* in die entsprechenden Logfiles und auf die Konsole.

Zeile 10 zeigt *atd*, der Benutzerprogramme zu vorher festgelegten Zeiten aufruft. *atd* ist die Entsprechung von *crond*, aber für Benutzer.

Zeile 11 zeigt *crond*, der zu genau bestimmten Zeiten System- und Verwaltungsprogramme aufruft.

Zeile 12 zeigt *inetd*, den übergeordneten Server für Netzwerkdienste. Er wartet an Stelle anderer Dienste auf Verbindungen, sodass nicht ständig unzählige separate Dämonen laufen müssen, ohne dass tatsächlich aktive Verbindungen bestehen.

Zeile 13 zeigt *named*, den DNS-Nameserver.

Zeile 14 zeigt *httpd*, den Apache-Webserver. Dieser Server wird nicht über *inetd* verwaltet, sodass der Prozess ständig im Hintergrund bleibt.

Zeile 15 zeigt *xfs*, den Fontserver von X-Windows.

Zeile 16 zeigt `sshd`, den SSH-Server. Auch dieser wird in der Regel nicht von `inetd` gestartet und ist deshalb permanent vorhanden.

Zeile 17 zeigt `mingetty` auf `tty2`. `mingetty` wird über `/etc/inittab` gestartet und nimmt Logins von einem Terminal entgegen. `tty2` ist eine virtuelle Konsole.

Zeile 18 zeigt `update (bdfush)`. `update` ist die Userspace-Entsprechung zu `kflushd` und schreibt modifizierte Puffer regelmäßig auf die Festplatte.

Zeile 19 zeigt `mingetty` auf `tty1`. `mingetty` wird über `/etc/inittab` gestartet und nimmt Logins von einem Terminal entgegen. `tty1` ist die physikalische Konsole.

Zeile 20 zeigt den Mailserver `sendmail`.

Zeile 21 zeigt `in.telnetd`, der von `inetd` für meine `telnet`-Verbindung aufgerufen wurde, über die ich den `ps`-Befehl eingab.

Zeile 22 zeigt meine `login`-Sitzung.

Zeile 23 zeigt meine Shell, die `ksh`.

Zeile 24 zeigt den `ps`-Prozess, der die Beispielausgabe erzeugt hat.

Wenn Sie mit `ps` einen unbekannten oder unerwarteten Prozess entdecken, deutet das möglicherweise darauf hin, dass sich jemand unerlaubten Zugang zu Ihrem Rechner verschafft hat. Kapitel 8 diskutiert, was in so einem Fall zu tun ist.

6.5 Die Protokolldateien Ihres Systems

Der Protokolldämon `syslogd` zeichnet Geschehnisse auf Ihrem System auf. Bei der Standardkonfiguration von Red Hat landen die meisten Aufzeichnungen in `/var/log/messages`. Viele Programme greifen für Protokolle einfach auf das `syslog` zurück, aber einige – z.B. der Apache-Webserver – verwalten lieber ihre eigenen Logdateien.

6.5.1 Was wird wohin geschrieben?

Normalerweise stehen die meisten Logdateien im Verzeichnis `/var/log`. Welche Dateien im Einzelnen angelegt werden, und was sie enthalten, definieren Sie in der Konfigurationsdatei `/etc/syslog.conf`. Die verschiedenen Linux-Distributionen verwenden hier unterschiedliche Ansätze. Praktisch immer existiert eine Datei `/var/log/messages`. Bei Red Hat 6.0 werden insgesamt vier Dateien angelegt: `messages`, `secure`, `maillog` und `spooler`:

- `/var/log/messages` erhält die meisten Nachrichten und kann sogar die einzige existierende Datei sein. Eine Kopie aller Konsolenmeldungen wird hierher geschrieben, dazu alle Betriebssystemmeldungen des Kernels und viele Meldungen von Programmen, die den Systemaufruf `syslog()` verwenden, z.B. `named`, `sendmail` oder `login`.

- Wenn `root` oder ein anderer Benutzer sich einloggt oder jemand mit `su` seine Identität wechselt, wird das in `/var/log/secure` aufgezeichnet. Daneben werden hier Verbindungen von anderen Computern sowie fehlgeschlagene Logins festgehalten.
- Protokolle über ankommende und abgehende E-Mail und den Status des Mail-servers finden Sie in `/var/log/maillog`.
- `/var/log/spooler` enthält Protokolle von `uucp`- und Newsserver (`innd`); damit wird es in der Regel wohl leer bleiben.

Hinweis

Definition abweichender Logdateien

Sie können Meldungen in andere Logdateien umleiten oder kopieren und sie so nach Thema oder Wichtigkeit sortieren.

6.5.2 Konfiguration des syslog

Nicht alle Meldungen sind gleich interessant oder wichtig. Über die Datei `/etc/syslog.conf` beschreiben Sie Ihre eigenen Bedürfnisse. In dieser Konfigurations-datei passen Sie die Protokollvorgänge an Ihre Vorstellungen an.

Protokollmeldungen lassen sich nach dem Subsystem unterscheiden, das sie generiert. In den Manual-Pages heißen diese Kategorien *Facilities* (siehe Tabelle 6.1).

Facility	Nachrichtenkategorie
auth oder security	Sicherheit und Autorisierung
authpriv	private Meldungen zu Sicherheit und Autorisierung
cron	Meldungen des cron-Dämons
daemon	Meldungen von Systemdämonen
ftp	Meldungen des ftp-Servers
kern	Kernel-Meldungen
lpr	Subsystem »Drucken«
mail	Subsystem »E-Mail«
news	Subsystem »News«
syslog	Meldungen von syslogd selbst
user	Meldungen von Benutzerprogrammen
uucp	Subsystem »UUCP«

Tab. 6.1: Kategorien (»Facilities«) für das syslog

Innerhalb jeder Facility werden die einzelnen Meldungen nach *Prioritäten* unterteilt. Tabelle 6.2 zeigt die definierten Prioritäten, nach zunehmender Bedeutung geordnet.

Priorität	Nachrichtentyp
debug	Meldungen für die Fehlersuche
info	Statusmeldungen mit rein informativem Charakter
notice	Normale, aber wichtige Zustände
warning oder warn	Warnmeldungen
error oder err	Fehlermeldungen
crit	Gefährliche Zustände
alert	Sofortiges Handeln notwendig
emerg oder panic	System unbrauchbar

Tab. 6.2: Prioritäten für das syslog

Jeder Eintrag von `syslog.conf` beschreibt für eine gegebene Facility und eine gegebene Priorität, wohin die entsprechenden Meldungen geschrieben werden. Die Priorität ist dabei ein Mindestwert, d.h. alle Meldungen mit mindestens der angegebenen Priorität sind gemeint. Wenn Sie z.B. als Priorität `error` angeben, gilt die jeweilige Regel für Nachrichten mit genau dieser Priorität ebenso wie für alle mit höherer Priorität.

Protokolle können auf Geräte (z.B. auf die Konsole) oder in Dateien geschrieben oder an andere Computer gesendet werden.

Im folgenden Beispiel werden alle Kernelmeldungen auf die Konsole geschrieben und zusätzlich an die Datei `/var/log/messages` angehängt. Eine Meldung darf durchaus an mehrere verschiedene Empfänger verschickt werden.

```
kern.* /dev/console
kern.* /var/log/messages
```

Die folgende Zeile schreibt Panikmeldungen überallhin, einschließlich der Datei `/var/log/messages`, der Konsole und auf alle Benutzerterminals:

```
*.emerg *
```

Die folgenden beiden Einträge schreiben Informationen zur `root`-Authentifizierung und zu Verbindungen zu fremden Rechnern nach `/var/log/secure`, Meldungen über Benutzerautorisierung nach `/var/log/auth`. Weil die Priorität `info` ist, werden Debugging-Meldungen (mit der Priorität `debug`) nicht aufgezeichnet.

```
authpriv.info /var/log/secure
auth.info /var/log/auth
```

Die folgenden Zeilen schreiben allgemeine Informationen von Dämonen nach `/var/log/daemon`, Meldungen über E-Mail nach `/var/log/maillog`:

```
daemon.notice                /var/log/daemon
mail.info                    /var/log/maillog
```

Dämonenmeldungen der Prioritäten `debug` und `info` sowie Mail-Meldungen der Priorität `debug` werden verworfen – mir ist das so am liebsten. `named` und die regelmäßige Mailkontrolle produzieren regelmäßig uninteressante Nachrichten.

Der letzte Eintrag schließlich lenkt Meldungen aller Kategorien nach `/var/log/messages` um, sofern die Priorität mindestens `info` ist. Die Facilities `auth`, `authpriv`, `daemon` und `mail` werden allerdings ignoriert; wir lassen sie ja bereits in eigene Protokolldateien schreiben.

```
*.info;auth,authpriv,daemon,mail.none    /var/log/messages
```

Hinweis

Tipps zu den Dateien in `/var/log`

`syslogd` legt keine Dateien neu an, sondern hängt Daten immer nur an bereits bestehende Files an. Wenn eine Datei noch nicht existiert, erstellen Sie sie mit `touch` und vergewissern Sie sich, dass sie `root` gehört. Aus Sicherheitsgründen sind die Logdateien für einfache Benutzer oft nicht lesbar. Insbesondere darf nur `root` das Sicherheitsprotokoll `/var/log/secure` lesen.

Hinweis

Weitere Informationen zur `syslog`-Konfiguration

Eine vollständigere Beschreibung der Möglichkeiten der `syslog`-Konfiguration sowie Beispieleinstellungen finden Sie in den Manual-Pages zu `syslog.conf(5)` und `syslogd(8)`.

6.5.3 Was bedeuten die Meldungen der Firewall?

Bevor Sie überhaupt Firewall-Logs erhalten, müssen Sie das Firewall-Logging im Kernel einschalten. Bei Red Hat 6.0 ist das schon beim mitgelieferten Kernel der Fall; bei früheren Versionen müssen Sie den Kernel gegebenenfalls neu übersetzen.

Wenn in einer Firewall-Regel die Option `-l` gesetzt ist, werden alle passenden Pakete als `kern.info`-Meldungen über das `syslog` aufgezeichnet. Die meisten Felder aus dem IP-Header werden dabei mit angegeben.

Die Firewall-Protokolle landen normalerweise in `/var/log/messages`. Wenn Sie möchten, legen Sie dafür eine eigene Protokolldatei an. Erstellen Sie eine leere Datei und hängen Sie eine entsprechende Zeile an `/etc/syslog.conf` an:

```
kern.info                                /var/log/fwlog
```

Alle Kernmeldungen erscheinen weiterhin auch in `/var/log/messages`. Nach abgeschlossenem Bootvorgang erzeugt der Kernel kaum noch Info-Meldungen außer den Firewall-Logs.

Beispielsweise würde die folgende Regel, die den Zugriff auf den RPC-Portmapper (Port 111) verbietet, etwa so eine Nachricht in `/var/log/messages` generieren:

```
ipchains -A input -p udp -i $EXTERNAL_INTERFACE \
-d $IPADDR 111 -j DENY -l
```

(1) (2) (3) (4) (5) (6) (7) (8)
Jun 9 14:07:01 firewall kernel: Packet log: input DENY eth0 PROTO=17

(9) (10) (11) (12)
10.10.22.85:14386 192.168.10.30:111

(13) (14) (15) (16) (17)
L=316 S=0x00 I=14393 F=0x0000 T=52

Um uns die Diskussion zu erleichtern, habe ich die einzelnen Felder der Nachricht durchnummeriert.

- Feld 1 ist das Datum, Jun 9.
- Feld 2 ist die Zeit, 14:07:01.
- Feld 3 ist der Hostname des Computers, firewall.
- Feld 4 ist die Facility, von der die Meldung stammt, in diesem Fall kernel. Die Logroutine von IPFW hängt noch einen Hinweis Packet log an.
- Feld 5 ist die Firewall-Chain der Regel, hier input. Sie erinnern sich: die eingebauten Chains sind input, output und forward.
- Feld 6 gibt an, was mit dem Paket geschah: DENY. Hier könnte stattdessen auch ACCEPT, REJECT, MASQ, REDIRECT oder RETURN stehen.
- Feld 7 ist das Netzwerkinterface, über das das Paket empfangen oder gesendet wurde, hier eth0.
- Feld 8 ist der Protokolltyp für die Nachricht, hier PROTO=17. Mögliche Werte sind u.a. 6 (TCP), 17 (UDP) oder 1 (ICMP). Es existieren noch weitere Protokolltypen; eine Liste finden Sie z.B. in der Datei `/etc/protocols`.
- Feld 9 ist die Absenderadresse des Paketes, 10.10.22.85.

- Feld 10 ist der Absender-Port des Paketes, 14386.
- Feld 11 ist die Empfängeradresse des Paketes, 192.168.10.30.
- Feld 12 ist der Empfänger-Port des Paketes, 111.

Die restlichen Felder sind nicht allzu interessant:

- Feld 13 enthält die Gesamtlänge des Paketes in Byte: $L=316$. Diese Zahl umfasst sowohl den Header als auch die Nutzdaten.
- Feld 14 ist das Type-of-Service-Feld (TOS): $S=0 \times 00$.
- Feld 15 ist die Datagramm-Kennung des Paketes: $I=14393$. Dabei handelt es sich entweder um die Paket-ID oder um das Segment, zu dem ein TCP-Fragment gehört.
- Feld 16 ist der Byte-Offset des Fragments: $F=0 \times 0000$. Wenn ein Paket ein TCP-Fragment enthält, gibt der Offset an, an welcher Stelle im rekonstruierten Paket dieses Fragment steht.
- Feld 17 ist die Lebenszeit des Paketes (Time-to-Live TTL): $T=52$. Diese Zahl gibt an, wie viele »Hops« oder Sprünge von einem Computer zum nächsten das Paket noch durchführen darf, bevor es wegen Überalterung zurückgeschickt wird.

Bei der Beurteilung des Protokolls sind die interessantesten Felder:

```
Jun  9 14:07:01 input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:111
```

D.h. das abgewiesene Paket ist ein UDP-Paket, das über das externe Interface `eth0` von einem unprivilegierten Port des Rechners 10.10.22.85 ankommt. Es richtet sich an den Port 111 (`sunrpc/portmap`) auf unserer Maschine. (Eine solche oder ähnliche Nachricht werden Sie recht häufig zu Gesicht bekommen, weil `portmap` ein häufig angegriffener Dienst ist.)

6.5.4 Häufig gescannte Ports

Wenn Sie die von Ihrer Firewall abgewiesenen Pakete protokollieren, werden Sie feststellen, dass nur eine kleine Untermenge der 65536 möglichen Ports gescannt wird. (Die neueren Stealth-Scanner hinterlassen keine Spuren in Ihren Logdateien, selbst wenn die Protokollierung eingeschaltet ist.) Oft testet jemand nur einen einzelnen Port, für den er sich besonders interessiert.

Hinweis

Bewertung der Aggressivität

Meine Bewertung der Aggressivität aus Tabelle 6.3 ist nur eine allgemeine Abschätzung. Jede Probe kann natürlich auch aus unschuldiger Neugier erfolgt sein. Solche subjektiven Einschätzungen beziehen sich auf die potenzielle Gefahr für Ihr System, wenn der Zugriff auf diesen Port erlaubt wäre, auf die Wahrscheinlichkeit, dass eine einzelne Probe dieses Ports feindlicher Natur ist, sowie darauf, wie oft der Port in der Vergangenheit an Angriffen beteiligt war. Letztlich müssen Sie die Bedeutung eines Ports für Ihr System von Fall zu Fall selbst beurteilen. Trotz allem ist es ein riesiger Unterschied, ob jemand ping oder traceroute benutzt oder nach einem Web- oder ftp-Server sucht, oder ob er systematisch nach einem offenen pop-, imap- oder portmap-Dämon forscht.

Tabelle 6.3 zeigt eine Auswahl der am häufigsten gescannten Ports. Die Services, die sich hinter den aufgeführten Ports verbergen, sind inhärent unsichere LAN-Services, sind Services mit bekannten Sicherheitslücken, sind Ziele bestimmter spezifischer Angriffe oder gehören zum typischen Erscheinungsbild bestimmter Angriffs-Tools. Die Ports aus Tabelle 6.3 werden in den CERT-Ratgebern am meisten diskutiert, und auch ich selbst habe in den letzten paar Jahren am häufigsten Scans auf sie gesehen. Cheswick und Bellovin warnen in ihrem Buch *Firewalls and Internet Security—Repelling the Wily Hacker* vor vielen dieser Ports.

Service	Port	Protokoll	Aggressivität	Kommentar
(reserviert)	0	TCP/UDP	hoch	als Absender und Empfänger gleichermaßen ungültig
	0-5	TCP	hoch	Signatur von <code>sscan</code>
echo	7	TCP/UDP	hoch	UDP-Angriff
systat	11	TCP	hoch	Information über laufende Prozesse (<code>ps</code>)
netstat	15	TCP	hoch	Abfrage des Netzwerkstatus: offene Verbindungen, Routingtabellen usw.
chargen	19	TCP/UDP	hoch	UDP-Angriff
ftp	21, 20	TCP	niedrig bis hoch	ftp-Service
ssh	22	TCP	mittel bis hoch	ssh-Service
ssh	22	UDP	niedrig	eine alte Version von PC Anywhere

Tab. 6.3: Häufig gescannte Ports

Service	Port	Protokoll	Aggressivität	Kommentar
telnet	23	TCP	hoch	telnet-Service
smtp	25	TCP	hoch	jemand sucht nach einem Spam-Relay oder nach einer alten Sicherheits-lücke
domain	53	TCP	hoch	TCP-Zone-Transfer oder Fälschung von DNS-Informationen
bootps	67	UDP	niedrig	vermutlich nur ein Fehler
tftpd	69	UDP	mittel bis hoch	unsichere Alternative zu ftp
finger	79	TCP	niedrig	Informationen über Ihre Benutzer
link	87	TCP	hoch	tty-link, von Angreifern häufig benutzt
pop-3	110, 109	TCP	hoch	einer der drei am häufigsten angegriffenen Ports
sunrpc	111	TCP/UDP	hoch	am häufigsten angegriffener Port
nntp	119	TCP	mittel bis hoch	öffentlicher Newsfeed oder Versenden von Spam
ntp	123	UDP	niedrig	Zeitsynchronisation über das Netz – akzeptabel, aber unhöflich
netbios-ns	137	TCP/UDP	niedrig bis hoch	für Unix harmlos
netbios-dgm	138	TCP/UDP	niedrig bis hoch	für Unix harmlos
netbios-ssn	139	TCP	niedrig bis hoch	für Unix harmlos
imap	143	TCP	hoch	einer der drei am häufigsten angegriffenen Ports
NeWS	144	TCP	hoch	ein Window-Manager
snmp	161, 162	UDP	mittel	Netzwerk-Administration und -Abfrage über das Internet
xdmcp	177	UDP	hoch	Login-Manager von X-Windows
exec	512	TCP	hoch	(nur für das Intranet)
biff	512	UDP	hoch	(nur für das Intranet)

Tab. 6.3: Häufig gescannte Ports (Forts.)

Service	Port	Protokoll	Aggressivität	Kommentar
login	513	TCP	hoch	(nur für das Intranet)
who	513	UDP	hoch	(nur für das Intranet)
shell	514	TCP	hoch	(nur für das Intranet)
syslog	514	UDP	hoch	(nur für das Intranet)
printer	515	TCP	hoch	(nur für das Intranet)
talk	517	UDP	mittel	(nur für das Intranet)
ntalk	518	UDP	mittel	(nur für das Intranet)
route	520	UDP	hoch	Routing-Tabellen
uucp	540	TCP	mittel	UUCP-Service
mount	635	UDP	hoch	Sicherheitslücke in mountd
socks	1080	TCP	hoch	Spam-Relay oder Sicherheitslücke im Proxy-Server
SQL	1114	TCP	hoch	Signatur von sscan
openwin	2000	TCP	hoch	OpenWindows
NFS	2049	TCP/UDP	hoch	Fernzugriff auf Dateien
pcanywhere	5632	UDP	niedrig	PC Anywhere
X11	6000+n	TCP	hoch	X-Windows
NetBus	12345, 12346, 20034	TCP	hoch	für Unix harmlos
BackOrifice	31337	UDP	hoch	für Unix harmlos
tracert	33434- 33523	UDP	niedrig	ankommendes tracert
ping	8	ICMP	niedrig bis hoch	ankommendes ping
redirect	5	ICMP	hoch	Redirect-Bombe
tracert	11	ICMP	niedrig	abgehende Antwort auf tracert
Test auf Unix	0	TCP/UDP	hoch	Broadcast an Empfänger 0.0.0.0

Tab. 6.3: Häufig gescannte Ports (Forts.)

Typische Beispiele für Portscans

Die Protokollmeldungen zu den am häufigsten gescannten Ports sind hier aufgelistet. Wenn Sie sich mit der Ausgabe einer Firewall bereits auskennen, überspringen Sie diesen Abschnitt. Wenn Sie bislang noch keine Firewall-Logs gesehen

haben, lernen Sie hier, was Ihnen später oft begegnen wird. (Die folgenden Zeilen sind leicht gekürzt, damit keine Zeilenumbrüche notwendig waren.)

- 22/UDP – PC Anywhere (alte Version):

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:22
```

- 23/TCP – telnet:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:23
```

- 25/TCP – smtp:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:25
```

- 79/TCP – finger:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:79
```

- 110/TCP – pop-3:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:110
```

- 111/UDP – sunrpc:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:111
```

- 119/TCP – nntp:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:119
```

- 123/UDP – ntp:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:123
```

- 143/TCP – imap:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:143
```

- 161/UDP – snmp:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:161
```

- 520/UDP – route:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:520
```

- 635/UDP – mount:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:635
```

- 1080/TCP – socks:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:1080
```


- 2049/UDP – nfs:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:2049
```

- 5632/UDP – PC Anywhere:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:5632
```

- 12345/TCP – NetBus:

```
input DENY eth0 PROTO=6 10.10.22.85:14386 192.168.10.30:12345
```

- 31337/UDP – BackOrifice:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:31337
```

- 33434:33523/UDP – traceroute:

```
input DENY eth0 PROTO=17 10.10.22.85:14386 192.168.10.30:33434
```

- 8/ICMP – ping echo-request:

```
input DENY eth0 PROTO=1 10.10.22.85:8 192.168.10.30:0
```

6.5.5 Pakete zur automatisierten Protokollanalyse

Programme zur Protokollanalyse überprüfen den Inhalt der System-Logs; bei anomalen Einträgen erstatten sie Meldung oder setzen eine vorher definierte Maßnahme um. Diese Tools laufen entweder ständig im Hintergrund, werden regelmäßig durch `crond` aufgerufen oder werden manuell gestartet. Sie erkennen potenzielle Sicherheitsprobleme und benachrichtigen Sie, wenn fragwürdige Protokolleinträge auftreten.

Ein Log-Analyse-Tool namens `swatch` ist Teil der Red Hat-Distribution. Weitere, ähnliche Programme sind über das World Wide Web erhältlich. Drei Pakete, die besonders nützlich oder besonders leicht erhältlich sind, sollen hier kurz vorgestellt werden: `autobuse`, `logcheck` und `swatch`. Alle drei sind ziemlich frei konfigurierbar und können Sie bei unerwarteten Ereignissen per E-Mail benachrichtigen.

autobuse

`autobuse` überprüft neue Logeinträge regelmäßig auf häufige Proben. Fragwürdige Ergebnisse erhalten Sie per E-Mail. `autobuse` ist Copyright 1998 Grant Taylor; Sie erhalten es von <http://www.picante.com/~gtaylor/download/autobuse/>.

logcheck

`logcheck` durchsucht die neuen Protokolleinträge regelmäßig nach Sicherheitslücken und unerwarteten Vorgängen. Alles, was nicht ganz astrein erscheint, wird Ihnen per E-Mail vorgelegt. `logcheck` wurde einem ähnlichen Paket aus dem »TIS

Gauntlet«-Firewallpaket nachempfunden. Bereits in der Voreinstellung kann es viele Muster in Logeinträgen erkennen. (Ich benutze `logcheck` selber – es ist ein tolles Paket.) Der Autor von `logcheck` ist Craig H. Rowland <crowland@psionic.com>. Das Paket finden Sie bei www.psionic.com.

swatch

»swatch« steht für »simple watcher« (»einfacher Aufpasser«) und ist ein einfach konfigurierbarer Logfilter. Es überprüft Protokolle und führt, je nach den darin gefundenen Mustern, selektiv ein oder mehrere benutzerdefinierte Maßnahmen durch. Es lässt sich periodisch von `crond` aufrufen, läuft auf Wunsch aber auch ständig im Hintergrund und überwacht das `syslog` dann in Echtzeit. `swatch` ist in Red Hat bereits enthalten.

6.6 Zusammenfassung

In diesem Kapitel haben wir uns darauf konzentriert, dass die ausgewählten Programme und Dienste laufen, und angesehen, welche Ports sie benutzen. Praktische Diagnoseprogramme in diesem Zusammenhang sind `ifconfig`, `ping`, `netstat` und `ps`. Serverprogramme und auch der Kernel selbst schreiben Status- und Fehlermeldungen in das `syslog`, dessen Dateien in `/var/log` stehen. Die Konfigurationsdatei des `syslog`-Dämons `/etc/syslog.conf` wurde vorgestellt, und es wurde besprochen, was für Informationen eine Protokollmeldung der Firewall enthält. Die am häufigsten gescannten Ports, die in den Firewallprotokollen oft auftauchen, wurden beschrieben. Schließlich wurden einige Logüberwachungs- und -analyseprogramme besprochen, die mit der Linux-Distribution mitgeliefert werden oder aus dem Internet erhältlich sind. Diese Protokoll-Überwacher können Sie benachrichtigen oder anderweitig aktiv werden, wenn sie verdächtige Muster in den Protokollen finden.

Nachdem Sie jetzt wissen, welche Dienste laufen, welche Ports sie benutzen, und was die Protokollnachrichten der Firewall bedeuten, beschreibt Kapitel 7 Sicherheitsmaßnahmen auf höheren Ebenen. Eine Paketfilter-Firewall ist schließlich keine vollständige Sicherheitslösung. Auf der Ebene der Systemadministration lassen sich alle Dienste durch entsprechendes Tuning ihrer eigenen Konfigurationsdateien noch sicherer gestalten. Ein Paketfilter konzentriert sich darauf, welche Verbindungen zu welchen Ports gestattet sind. Auf Anwendungsebene geht es darum, den Zugriff auf die Programme dahinter noch selektiver nur für bestimmte Hosts und User zu erlauben.

Kapitel 7

Ergänzende Maßnahmen durch die Unix- Systemadministration



- 7.1 Authentifizierung: Prüfung der Identität
- 7.2 Autorisation: Festlegung von Zugriffsrechten
- 7.3 Serverspezifische Konfiguration
- 7.4 SOCKS: eine Proxy-Firewall auf Anwendungsebene
- 7.5 Diverse Systemaccounts in /etc/passwd und /etc/group
- 7.6 Die PATH-Variable
- 7.7 /etc/issue.net
- 7.8 Protokollierung auf andere Rechner
- 7.9 Halten Sie Ihre Software auf dem Laufenden!
- 7.10 Zusammenfassung

In diesem Kapitel lassen wir die grundlegende Ebene der Firewall hinter uns und sehen uns einige der Sicherheitsmaßnahmen an, die Ihnen auf höheren Ebenen zur Verfügung stehen. Eine Firewall ist keine vollständige Sicherheitslösung: Auf der Ebene der Systemadministration können Sie jeden Dienst noch weiter schützen, indem Sie die Einstellungen in seinen eigenen Konfigurationsdateien sorgfältig vornehmen. Ein Paketfilter legt auf der Basis von IP-Adresse und Port fest, welche Verbindungen erlaubt sind und welche nicht. Auf der Anwendungsebene geht es uns mehr darum, welche spezifischen Netzwerke, Hosts und Benutzer Zugriff zu den jeweiligen Programmen erhalten, und welche Dateien sie darüber benutzen dürfen.

7.1 Authentifizierung: Prüfung der Identität

Die IP-Netzwerkschicht kennt keine Authentifizierung. Die einzige Information, die zur Identifizierung benutzt werden könnte, ist die Absenderadresse aus dem Header des IP-Paketes. Wie Sie bereits gesehen haben, lassen sich solche Absenderadressen aber leicht fälschen. Die eigentliche Authentifizierung – die Überprüfung, dass der Absender wirklich der ist, der zu sein er vorgibt – erfolgt auf höheren Ebenen. Im TCP/IP-Referenzmodell liegt sie in der Zuständigkeit der Anwendungsschicht.

Die Benutzerauthentifizierung erfolgt unter Unix letztlich über geheime Passwörter, die ausschließlich der Benutzer kennt, dem der jeweilige Account gehört. Auf diesem Mechanismus aufbauend existieren weitere LAN-Funktionalitäten, die z.B. den Ursprungshost eines fremden Benutzers mit in die Entscheidung einbeziehen oder zentralisierte Benutzerdatenbanken benutzen. Die Benutzerauthentifizierung ist eine der Grundlagen von Unix-Systemsicherheit. Wie bei allen Systemen in der Entwicklung waren auch beim Unix-Passwortmechanismus immer wieder Erweiterungen und Korrekturen notwendig. In diesem Abschnitt geht es um zwei dieser Erweiterungen: Shadow-Passwörter und MD5-Hashes. Zwei der beliebtesten Authentifizierungsmechanismen in LANs, die auf dem Passwortmechanismus aufbauen, werden besprochen: die *rhost*-Authentifizierung sowie NIS. Bei beiden werden wir sehen, wie sie funktionieren und warum sie in der Zusammenarbeit mit dem Internet inhärent unsicher sind.

7.1.1 Shadow-Passwörter

Normalerweise werden Unix-Passwörter verschlüsselt in der Datei `/etc/passwd` gespeichert. Diese Datei ist für jedermann – jeden Benutzer und jedes Programm auf dem System – lesbar, weil die darin enthaltenen Informationen über Benutzeraccounts, z.B. Benutzer- und Gruppenkennung, Passwort, Benutzername, Arbeitsplatz und Telefonnummer, Heimatverzeichnis und Lieblings-Shell, von vielen Programmen benötigt werden.

Hinweis***Fernzugriff auf die Passwort-Datei***

Die Datei `/etc/passwd` ist frei lesbar und damit – unter bestimmten Umständen – auch für Benutzer fremder Systeme zugänglich, z.B. über einen falsch konfigurierten FTP-Server, ein von außen erreichbares, über NFS gemountetes Root-Dateisystem oder Zugang zu den NIS-Datenbanken sind Beispiele, ebenso Zugriffe von gefälschten IP-Adressen über den `rhost-Authentifizierungsmechanismus` oder ein Einbruch über den `sendmail`-Port. Jede dieser Gefahrenstellen wird in diesem Kapitel noch näher besprochen.

Da das File für jedermann lesbar ist, kann ein Angreifer Brute-Force-Methoden versuchen und die Passwörter knacken, indem er einfach alle möglichen Kombinationen von acht Buchstaben ausprobiert und das verschlüsselte Ergebnis mit dem verschlüsselt gespeicherten Passwort vergleicht. Früher war diese Methode so aufwändig, dass sie auf einem normalen Computer völlig undenkbar war. Heute sind die Computer aber viel leistungsfähiger. Man kann ein Programm zum Knacken von Passwörtern ständig im Hintergrund laufen lassen und so ziemlich schnell die echten Passwörter herausfinden.

Shadow-Passwörter sind ein Ansatz, diese Schwäche zu umgehen. Dabei liegen die verschlüsselten Passwörter in einer Shadow-Passwortdatei, die eigens für diesen Zweck angelegt wird. Alle »normalen« Informationen über die Benutzeraccounts sind weiterhin in `/etc/password` gespeichert und für alle lesbar. Das eigentliche verschlüsselte Passwort hingegen liegt in `/etc/shadow` und ist damit nur Programmen zugänglich, die `root`-Rechte besitzen.

Ein gewisses Maß an Unterstützung für Shadow-Passwörter ist bei Red Hat Linux schon lange gegeben. Seit der Version 6.0 sind Shadow-Passwörter nahtlos in die grafische Oberfläche für die Installation integriert. Sie können sie aktivieren, indem Sie einfach die entsprechende Option anklicken. Bei früheren Linux-Distributionen müssen Sie die Passwörter manuell über das Programm `pwconv` konvertieren.

7.1.2 MD5-Hashes für Passwörter

Shadow-Passwörter versuchen, die potenzielle Sicherheitslücke einer frei lesbaren Passwortdatei zu umgehen. MD5-Hashes wollen auf einer etwas allgemeineren Ebene das Problem beheben, dass Passwörter mit nur acht Zeichen Länge heute relativ leicht geknackt werden können.

Passwörter, deren MD5-Hash gespeichert ist, dürfen bis zu 256 Zeichen lang sein. Wenn man von einem beliebigen Objekt einen MD5-Hash anfertigt, erhält man einen 128-Bit-Wert, von dem man annimmt, er sei unmöglich berechenbar – zumindest so unmöglich, wie man es vor 10 oder 15 Jahren von den alten, acht Zeichen langen, DES-verschlüsselten Passwörtern gedacht hat.

Red Hat unterstützt ab Version 6.0 die MD5-Bibliothek. Das Installationsprogramm sowie die Konfigurationsprogramme `linuxconf` und `control-panel` kennen sich mit MD5-Passwörtern aus und aktivieren sie, wenn Sie die entsprechende Option anklicken.

7.1.3 Die rhost-Authentifizierung von Berkeley: `hosts.equiv` und `.rhosts`

Hostbasierte Authentifizierung geht von der Prämisse aus, dass ein Benutzer, der sich einmal gegenüber einem Computer authentifiziert hat, automatisch auch auf allen anderen Computern des lokalen Netzwerkes zugriffsberechtigt wird. Für ein ganzes System definiert der Administrator diese Rechte, indem er die entsprechenden vertrauenswürdigen Hosts in die Datei `/etc/hosts.equiv` einträgt. Einzelne Benutzer können ebenfalls solche Zugriffsrechte einschalten, indem sie die entsprechenden Hosts in ihre eigene Datei `~/.rhosts` aufnehmen.

Die `rhost`-Authentifizierung wurde in BSD 4.2 eingeführt. In Verbindung mit den Kommandos `rlogin` und `rsh` konnte ein Benutzer damit bequem auf alle Rechner in seinem LAN zugreifen, sobald er sich irgendwo eingeloggt hatte. In solchen vertrauenswürdigen Umgebungen wählte man – sowohl als Administrator wie auch als einfacher Benutzer – oft den einfachsten Weg: Man erlaubte den Zugriff nicht nur von wenigen ausgewählten Rechnern, sondern von `+`, d.h. allen Rechnern, auf denen man selbst einen Account hatte. Besonders für den `root`-Account war das praktisch: der Systemadministrator durfte auf allen Computern arbeiten.

Ganz offensichtlich eignet sich `rhost` nur für LANs, nicht für das Internet. Allen Benutzern lokaler Maschinen vertraut man, einfach wegen ihrer Hostnamen. Im Internet lassen sich sowohl die IP-Absenderadressen als auch die DNS-Datenbanken fälschen: deshalb die ständigen Warnungen in diesem Buch, dass die »r«-Services von Berkeley auf einer Firewall gesperrt sein müssen.

Die `rhost`-Authentifizierung ist in einem LAN mit mehreren Benutzern auf mehreren Maschinen äußerst bequem. Auf einer Maschine, die aus dem Internet erreichbar ist, muss sie aber deaktiviert bleiben. Insbesondere muss diese Authentifizierungsmethode auf Firewalls ausgeschaltet werden, indem man alle `.rhost`-Dateien löscht und den Zugang zu den `r`-Befehlen sperrt.

7.1.4 Gemeinsamer Zugang zu zentralen Authentifizierungsdatenbanken: der Network Information Service (NIS)

Der *Network Information Service* (NIS) erlaubt in einem LAN die zentrale Verwaltung von Daten über Benutzer und Computer. NIS ist Teil der Red Hat-Distribution und kann bei der Installation zusammen mit Shadow-Passwörtern und der MD5-Verschlüsselung eingeschaltet werden. Wie bei der `rhost`-Authentifizierung und den `r`-Befehlen gilt auch hier, dass der Mechanismus auf einer Firewall ausgeschaltet bleiben sollte.

Clientcomputer fragen Daten zu Hosts, Benutzern, Passwörtern und Gruppenzugehörigkeit von einem NIS-Server ab. Die Clients können auch andere Clients nach einem NIS-Server fragen. Clients dürfen auf Clients und Server in anderen Domains zugreifen. Ohne (einstellbare) Einschränkungen der erlaubten IP-Adressen antwortet der NIS-Server jedem, der seinen Domainnamen kennt. Benutzer können die verschlüsselten Passwörter sehen und die entsprechenden Informationen auch über das Netz ändern.

NIS ist ein UDP-basierter Dienst, auf den man über den `portmap`-Dämon zugreift. Wie bei allen `portmap`-Dienstern bleibt NIS auch dann zugänglich, wenn die Firewall den Zugriff auf den `portmap`-Port sperrt: Ein Portscan liefert schnell den Port, den der NIS-Server benutzt.

Aus der Perspektive eines Sicherheitsfachmanns verkörpert NIS so ungefähr das Schlimmste, was passieren kann: Passwortdateien für die Benutzerauthentifizierung und `/etc/hosts`-Dateien für die Computerauthentifizierung sind beide gleichermaßen aus dem Internet lesbar. Zudem kann ein Paketfilter den Service überhaupt nicht schützen, wenn man nicht generell den Zugriff auf alle unprivilegierten UDP-Ports verbieten will.

7.2 Autorisation: Festlegung von Zugriffsrechten

Wenn die Identität des Benutzers überprüft ist, der Authentifizierungsprozess also sichergestellt hat, dass der Benutzer wirklich der ist, der zu sein er vorgibt, geht es als Nächstes darum, auf welche Ressourcen des Computers er zugreifen darf. Das beinhaltet folgende systemweiten Fragen: Wer oder was darf `root`-Privilegien ausüben? Wer oder was darf die Identität eines fremden Accounts annehmen? Welche fremden Systeme dürfen auf welche lokalen Netzwerkdienste zugreifen? Wer oder was darf welche Dateien und Verzeichnisse lesen, be- oder überschreiben oder ausführen?

7.2.1 Zugriff auf den `root`-Account

Programme mit `root`-Rechten dürfen *per definitionem* alles tun und auf alles zugreifen. Ganz offensichtlich muss der Zugang zu diesen Privilegien stark eingeschränkt werden. Folglich darf `root` sich nicht von einem anderen Computer aus einloggen, und ebenso sollten andere Dienste – z.B. `ftp` – Benutzern auf fremden Computern keinen Zugang zum `root`-Account gewähren.

Die Datei `/etc/securetty` kontrolliert, von wo aus ein `root`-Login erlaubt wird. Normalerweise bleibt das auf die physikalische Konsole selbst und alle virtuellen Konsolen des gleichen Terminals beschränkt. Pseudoterminals, `tty`-Geräte, haben in `/etc/securetty` nichts verloren.

7.2.2 Verwendung von `su` einschränken

Mit dem Befehl `su` wechselt man zu einem anderen Benutzeraccount und nimmt damit die Identität eines anderen Benutzers und einer anderen Gruppe an. Obwohl `su` das Passwort des Zielaccounts abfragt, wollen manche Systemadministratoren trotzdem die Verwendung dieses Befehls einschränken.

Unter Linux kontrolliert man über die Mitgliedschaft in Benutzergruppen, wer welche Programme ausführen darf. Die `wheel`-Gruppe dient traditionell der Zugriffskontrolle für `su` und andere Administrationsprogramme. Sie sehen hier einen mehrschichtigen Ansatz der Zugriffskontrolle, sozusagen eine Art »Sicherheit durch Tiefe«: Ein Benutzer muss das Zielpasswort kennen, bevor er `su` verwenden darf. Die meisten administrativen Programme nützen ihm aber ohnehin nichts, weil er auf die zugehörigen Systemressourcen nicht schreiben darf. Indem man die Ausführbarkeit dieser Programme nochmals einschränkt, erhält man eine zusätzliche Sicherheitsschicht.

Wenn Sie nur ausgewählten Benutzern die Verwendung von `su` erlauben wollen, tragen Sie diese Benutzer zunächst in `/etc/group` in die `wheel`-Gruppe ein. Anschließend ändern Sie die Gruppenzugehörigkeit und Zugriffsrechte für das `su`-Programm.

1. Nehmen Sie die entsprechenden Benutzer in die `wheel`-Gruppe auf, indem Sie die Datei `/etc/group` bearbeiten.
2. Ändern Sie die Gruppenzugehörigkeit von `su` auf `wheel`:

```
chgrp wheel /bin/su
```

3. Ändern Sie die Zugriffsrechte von `su`: Setzen Sie das `uid`-Bit (sonst funktioniert `su` nicht mehr), geben Sie dem Eigentümer (`root`) die Rechte für Lesen, Schreiben und Ausführen, der Gruppe (`wheel`) die für Lesen und Ausführen, und verbieten Sie alle anderen Zugriffe:

```
chmod 4750 /bin/su
```

7.2.3 Die `tcp_wrapper`

Mit Hilfe des TCP-Wrappers `tcpd` definieren Sie Zugriffskontrollen für lokale Dienste, wobei Sie explizit fremden Netzwerken, Systemen oder Benutzern den Zugriff erlauben oder verbieten können. Damit definieren Sie ganz genau, welche fremden Systeme welche Netzwerkdienste Ihres Systems benutzen dürfen.

`tcpd` ist ein Wrapper-Programm. Das heißt, dass `inetd` so umkonfiguriert wird, dass es anstelle des eigentlichen Serverprogrammes `tcpd` aufruft. `tcpd` prüft die Autorisation und startet dann den eigentlichen Service-Dämon. Weil `tcpd` ein Filter ist, der nur ganz kurz zwischen `inetd` und dem angeforderten Server läuft, muss es nicht ständig im Hintergrund aktiv sein. Dieser ganze Autorisationsme-

chanismus wird meistens kurz `tcp_wrappers` genannt. Die `tcp_wrapper` führen sowohl eine Rechnerauthentifizierung als auch eine Prüfung der Autorisation durch.

Bei Red Hat Linux ist `/etc/inetd.conf` bereits so konfiguriert, dass alle Services, die damit zurechtkommen, die `tcp_wrapper` verwenden. Dazu gehören vor allem `ftp`, `telnet`, `pop-3`, `imap`, `finger` und `identd` bzw. `auth`.

Die TCP-Wrapper funktionieren nicht für alle Dienste. Grundsätzlich muss der Dienst unter der Obhut von `inetd` stehen, d.h. erst wenn eine TCP-Verbindung oder ein UDP-Datagramm ankommt, ruft `inetd` tatsächlich das Serverprogramm auf. Es ist zwar prinzipiell denkbar, Server wie den Apache-Webserver oder den SSH-Server über `inetd` auszuführen, aber derartige Dienste bleiben normalerweise besser ständig als Dämonen im Hintergrund. Bei Apache geschieht das ausschließlich aus Performance-Gründen; bei `sshd` könnte es im Falle einer `inetd`-Verbindung auf einem langsamen System mit starker Verschlüsselung sogar zu einem Timeout kommen. Zum Ausgleich lässt sich `sshd` so konfigurieren, dass es selbstständig die Inhalte der `hosts_access`-Dateien berücksichtigt.

Die durch die `tcp_wrapper` mögliche Autorisation von TCP-Diensten ist wesentlich besser als die von UDP-basierten Diensten.

Bei TCP-Diensten erfragt `tcpd` vom Client-System den Benutzernamen, sofern `identd` unterstützt wird. Die so erhaltene Information wird neben Hostname und beanspruchtem Dienst aufgezeichnet. Ein gewisser Schutz vor gefälschten IP-Adressen und DNS-Hostnamen wird dadurch realisiert, dass `tcpd` einen *Reverse-Lookup* durchführt. Dabei schlägt `tcpd` zunächst im DNS den Hostnamen nach, der zu der IP-Adresse des Clients gehört. Anschließend erfragt es die IP-Adresse zu eben diesem Hostnamen. Sie muss identisch sein mit der Client-IP. Vom Absender geroutete Pakete werden grundsätzlich nicht akzeptiert.

Bei UDP-Diensten ist nur eine grundlegende Zugriffskontrolle nach Hostname oder Adresse möglich, wobei aber auch hier ein Reverse-Lookup zum Schutz vor gefälschten IP-Adressen und Hostnamen erfolgt. Die Zugriffskontrolle ist jedoch nicht so stringent wie für TCP-Dienste: `inetd` startet den Server (auf Wunsch) mit einer Option `wait`, d.h. dass der Server nach dem Austausch des letzten Datagramms noch für ein paar Minuten warten soll, bevor er sich beendet. Wenn innerhalb dieser Zeit eine neue Anfrage eintrifft, muss nicht eigens ein weiterer Server aufgerufen werden. Wenn allerdings von irgendwoher eine neue Anfrage eintrifft, während der alte Server noch wartet, können die `tcp_wrapper` hierfür keine Prüfungen durchführen.

Wie bei den Firewallregeln gilt auch für die `tcp_wrapper`-Regeln, dass immer die erste passende Regel gilt. Per Voreinstellung ist der Zugriff erlaubt. Die Liste für erlaubte Clients wird vor der Liste für nicht erlaubte Clients abgearbeitet. Im Einzelnen geht `tcpd` die Zugriffslisten in der folgenden Reihenfolge durch:

1. Wenn eine Anfrage auf einen Eintrag in `/etc/hosts.allow` passt, wird der Zugriff erlaubt.

2. Wenn eine Anfrage auf keinen Eintrag in `/etc/hosts.allow`, aber auf einen in `/etc/hosts.deny` passt, wird der Zugriff verweigert.
3. Wenn überhaupt keine Regel zutrifft, wird der Zugriff erlaubt.

Bei der Einrichtung von `hosts.allow` und `hosts.deny` sollte das gleiche Ziel wie beim Firewall-Design gelten: Per Voreinstellung werden alle Pakete abgewiesen, wobei für die erlaubten Dienste explizite Ausnahmeregeln gelten.

Die wichtigsten Felder jeder Regel sind der Name des Serverprogramms sowie eine Liste erlaubter Clients. Alle Felder sind durch Doppelpunkte voneinander getrennt.

Platzhalter sind erlaubt. Die beiden am häufigsten eingesetzten Platzhalter sind `ALL` und `LOCAL`.

`ALL` erklärt sich von selbst: Im Feld für den Servernamen erlaubt es alle Services; im Feld für den Client erlaubt es alle Clients.

`LOCAL` bezieht sich auf das Loopback-Interface sowie auf alle unqualifizierten Hostnamen, d.h. alle Hosts ohne Punkt im Namen, oder Hostnamen ohne Domainnamen. Solche unqualifizierten Hostnamen stehen für die Maschinen in der gleichen Domain wie der eigene Computer. Beachten Sie aber, dass `tcpd` nur den ersten Namen in `/etc/hosts` berücksichtigt. Es ignoriert die optionalen Alias-Felder. Bei den meisten Privatinstallationen ohne eigenen Domainnamen hat die Firewall selbst einen vom Internet-Provider zugewiesenen Domainnamen. Die Computer aus dem internen LAN sind damit nicht mehr `LOCAL`, sondern gehören zu einer internen (privaten) Domain.

Der Platzhalter `PARANOID` steht für alle Hosts, deren Name nicht zu ihrer IP-Adresse passt. Normalerweise wird `tcpd` so übersetzt, dass diese `PARANOID`-Prüfung immer eingeschaltet ist, d.h. solche »unpassenden« Hosts werden bereits abgewiesen, bevor sich `tcpd` die Zugriffskontrolllisten auch nur ansieht. Falls Sie `tcpd` neu übersetzen und `PARANOID` dabei ausschalten, reaktivieren Sie die `PARANOID`-Prüfung mittels dieses Platzhalters gezielt für einzelne Dienste.

Zugriffslisten können auch Muster überprüfen. Die häufigsten Muster sind ein Punkt mit einem Domainnamen sowie eine Netzwerkadresse, die auf einen Punkt endet. Beide Muster gelten für alle Hosts in der entsprechenden Domain bzw. im angegebenen Netzwerk.

Hier ein Beispiel für `/etc/hosts.allow`:

1. `ALL: LOCAL .internal.lan`
2. `in.ftpd: friend@trusted.host.net`
3. `sshd: 10.30.27.`
4. `ipop3d: 10.30.27.45 EXCEPT PARANOID`

Zeile 1 erlaubt Clients vom eigenen Computer und aus dem internen LAN den Zugriff auf alle durch `tcp_wrapper` geschützten Dienste.

Zeile 2 erlaubt zusätzlich einem bestimmten Useraccount den Zugriff auf den FTP-Server. Die Angabe eines Benutzernamens ist nur möglich, wenn das fremde System eine `ident`-Anfrage zulässt. Damit funktioniert sie nicht, wenn es sich beim angegebenen Computer um einen PC oder einen Macintosh handelt.

Zeile 3 erlaubt zusätzlich jedem Host aus dem Netzwerk 10.30.27.0 den Zugriff auf den SSH-Server.

Zeile 4 erlaubt zusätzlich einer bestimmten IP-Adresse den Zugriff auf den POP-Server. Wenn `tcpd` ohne ständig eingeschaltete `PARANOID`-Funktionalität rekompiliert wurde, schaltet der Term `EXCEPT PARANOID` eben diese Funktionalität wieder ein. Zeile 4 besagt also, dass Verbindungen zum POP-Server von 10.30.27.45 erlaubt sind, aber nur, wenn IP-Adresse und Hostname nach einem Reverse-Lookup gültig erscheinen.

Hier ein Beispiel für `/etc/hosts.deny`:

```
ALL: ALL
```

Alle durch `tcp_wrapper` geschützten Dienste sind für alle Clients gesperrt. Weil die Regeln aus `hosts.deny` erst nach denen aus `hosts.allow` angewandt werden, setzt die angegebene Beispieldatei in der Praxis eine Voreinstellung, die immer dann gilt, wenn keine Ausnahme aus `hosts.allow` zutrifft.

Weitere Informationen zu den `tcp_wrappern` finden Sie u.a. in den folgenden Manual-Pages: `tcpd(8)`, `hosts_access(5)`, `hosts_options(5)`, `tcpdchk(8)`, `tcpdmatch(8)`, `inetd(8)` und `inetd.conf`.

Hinweis

Die tcp_wrapper und RPC

RPC-basierte Dienste können nicht über `tcpd` gestartet werden, aber der `portmap`-Dämon für RPC berücksichtigt ebenfalls die Zugriffslisten aus `/etc/hosts.allow` und `/etc/hosts.deny`. Der Servicename für die Listen ist immer `portmap`, egal wie der `portmap`-Dämon tatsächlich heißt. (Er wird manchmal auch `rpcbind` genannt.)

Die Zugriffskontrolllisten sind für RPC allerdings nur ein erster Teil einer Lösung, weil jeder einfache Portscan die offenen RPC-Ports anzeigt. Ein Angreifer kann den `portmap`-Dämon und die damit verbundene Zugriffskontrolle also leicht umgehen.

Weitere Informationen zur Verwendung von `hosts.allow` und `hosts.deny` im Zusammenhang mit `portmap` erhalten Sie in der Manual-Page `portmap(8)`.

7.2.4 Zugriffsrechte für Dateien und Verzeichnisse

Ganz eng zur Authentifizierung von Benutzern und Autorisierung für bestimmte Dienste gehört das Konzept, wer Zugriff auf Objekte im Dateisystem hat. Datei-bezogene Zugriffsrechte werden unter Unix jeweils für den Eigentümer und die Gruppe der Datei sowie für alle anderen Benutzer vergeben, und zwar getrennt nach Lesen, Schreiben und Ausführen. Ganz offensichtlich ist das Konzept des Zugriffsrechtes, wenn es darum geht, wer welche Datei beschreiben darf. Etwas raffinierter wird es, wenn es um Dienste geht, die für den Benutzer Systemprivilegien beanspruchen. Unter Unix können Sie die Zugriffsrechte der Server selbst ebenso einschränken wie den Teil des Dateisystems, den der Server sieht.

Dateien, die jeder überschreiben darf

Es ist bei keiner einzigen Datei notwendig, dass jeder auf sie schreiben dürfte. Wenn mehrere Benutzer in eine Datei schreiben sollen, fasst man sie am besten in einer gemeinsamen Gruppe zusammen, die den gemeinsamen Zugriff auf diese Datei erlaubt.

Der folgende Befehl zeigt Ihnen alle Dateien auf Ihrem Computer an, die für jedermann beschreibbar sind:

```
find / -perm -0002 -fstype ext2 -type f -print
```

Je nachdem, welche Teile der Linux-Distribution Sie installiert haben, wird dieser Befehl möglicherweise einige Dateien anzeigen. Spiele-Highscores unter `/var/lib/games` sind oft offen beschreibbar – eine bekannte Sicherheitslücke aktueller Red Hat-Distributionen. Von Zeit zu Zeit findet man auch beschreibbare Quellcodes oder Dokumentationsdateien. Dabei handelt es sich um Fehler auf Seiten der Entwickler.

Verzeichnisse, in die jeder schreiben darf

Nur sehr wenige Verzeichnisse benötigen ein Schreibrecht für alle. Im Idealfall sollte das nur bei `/tmp` der Fall sein. In der Praxis existieren auch darüber hinaus noch ein paar global beschreibbare Verzeichnisse für LAN-Dienste wie SAMBA. Der folgende Befehl gibt alle solchen Verzeichnisse aus:

```
find / -perm -0002 -fstype ext2 -type d -print
```

Je nachdem, welche Anteile der Linux-Distribution bei Ihnen installiert sind, gibt dieser Befehl mehr oder weniger Verzeichnisse aus. Mit der Ausnahme einiger weniger Verzeichnisse unterhalb von `/var` sollte aber nichts angezeigt werden.

Programme mit `setuid`- und `setgid`-Flags

Programme mit `setuid`- und `setgid`-Flags laufen unter der effektiven Benutzer- oder Gruppenkennung eines anderen Accounts. Das Programm greift für den aufrufenden Benutzer mit elevierten Privilegien auf Systemressourcen zu.

Beispielsweise greifen Benutzerprogramme wie `login`, `passwd` oder `su` auf privilegierte Benutzerkonto-Funktionen zu, die nur der `root`-Account verwenden darf. `sendmail` schreibt in die Mailboxdateien, die den einzelnen Benutzern gehören und normalerweise nur durch den entsprechenden User selbst sowie durch die Gruppe `mail` beschrieben werden dürfen. Die Berkeley-r-Programme (`rcp`, `rsh` und `rlogin`) bauen von privilegierten Ports Verbindungen auf, was nur mit `root`-Rechten möglich ist.

Programme mit `setuid`-Flags blicken auf eine lange Geschichte von Sicherheitsproblemen zurück. Ein erfolgreicher Eindringling wird als Erstes ein trojanisches Pferd installieren, bei dem das `setuid`-Flag gesetzt ist und das ihm somit `root`-Rechte verschafft.

Einer der gefährlichsten Fehler ist die Installation eines Shellskripts mit solch einem Flag. Ein Shellskript ist eine ausführbare Datei, die für einen Menschen leicht lesbar ist. Mit ein wenig Entschlossenheit findet man meistens eine Möglichkeit, das Skript zu verändern oder zu kopieren und die Befehle darin umzuschreiben. Die heute üblichen Verzeichnisrechte machen das allerdings viel schwieriger als es in der Jugendzeit von Unix war.

Software zur Überprüfung der Systemintegrität, wie sie z.B. in Kapitel 8 besprochen wird, sucht regelmäßig nach Programmdateien, bei denen unerwartet eines der `setuid`- oder `setgid`-Flags gesetzt ist. Mit folgendem Befehl können Sie solch eine Suche auch manuell durchführen:

```
find / \( -perm -4000 -o -perm -2000 \) -fstype ext2 -type f -print
```

Durch chroot eingeschränkte Services

Wenn öffentlich zugängliche Services aus dem Dateisystem lesen oder hineinschreiben, können Sie deren Zugriff auf einen bestimmten Verzeichnisbaum beschränken. `chroot` modifiziert die Perspektive eines Prozesses, indem es ein bestimmtes Verzeichnis für diesen Prozess als Root-Verzeichnis definiert. Alles oberhalb dieses Verzeichnisses ist für den Prozess unsichtbar und unerreichbar. Solch ein begrenztes »virtuelles« Dateisystem bietet über die Server-spezifischen Zugangsbeschränkungen hinaus einen zusätzlichen Schutz.

Wenn Sie Server in `chroot`-Umgebungen betreiben, müssen Sie evtl. Teile des Dateisystems kopieren, die außerhalb der `chroot`-Umgebung liegen. Z.B. verwendet `ftp` das `ls`-Programm, um den Benutzern den Inhalt des aktuellen `ftp`-Verzeichnisses anzuzeigen. Solche Programme liegen außerhalb des `chroot`-Verzeichnisses und müssen daher hineinkopiert werden, damit sie dem eingeschlossenen Prozess zugänglich sind. Ebenso müssen evtl. Konfigurationsdateien, Bibliotheken und Logdateien in die `chroot`-Struktur kopiert werden.

Weitere Informationen zu `chroot` finden Sie in der Manual-Page zu `chroot(1)`.

7.3 Serverspezifische Konfiguration

Ob mit oder ohne Firewall, den bestmöglichen Schutz erhalten Sie, indem Sie nur die unbedingt nötigen Services aktivieren. Wenn Sie einen Dienst nicht brauchen, schalten Sie ihn aus. Wenn Sie auf einen bestimmten Dienst aber nicht verzichten können oder wollen, dann sollten Sie bei seiner Konfiguration sehr sorgfältig vorgehen.

In diesem Abschnitt untersuchen wir die Konfiguration einzelner Server vom Standpunkt eines Sicherheitsexperten.

7.3.1 telnet-Konfiguration

telnet sollten Sie anderen Sites nicht anbieten; ebenso wenig sollten Sie es selbst beim Zugriff auf andere Computer verwenden, wenn es nicht unbedingt nötig ist. ssh ist telnet bei weitem vorzuziehen.

telnetd wird in `/etc/inetd.conf` aktiviert. Bei Red Hat ist telnetd normalerweise durch die `tcp_wrapper` geschützt. Schränken Sie den Zugriff auf Ihren telnetd-Server durch entsprechende Einträge in der Firewall-Konfiguration sowie in `/etc/hosts.allow` ein.

Wenn eine telnet-Sitzung beginnt, wird auf dem Terminal des Clients noch vor dem Login-Prompt der Inhalt von `/etc/issue.net` ausgegeben. Diese Datei enthält Systeminformationen, die Sie möglicherweise nicht an jeden weitergeben wollen, der neugierig eine Verbindung zu Ihrem Telnet-Port herstellt. Wie man `/etc/issue.net` verändert, wird weiter unten besprochen.

Was telnet im Besonderen angeht, müssen Sie nicht unbedingt den Inhalt von `issue.net` modifizieren. Stattdessen lässt sich der Server auch so konfigurieren, dass er diese Datei ignoriert und einfach nur einen Login-Prompt ausgibt. Dazu starten Sie ihn mit der Option `-h`, indem Sie seinen Eintrag in `/etc/inetd.conf` wie folgt editieren:

```
telnet stream tcp      nowait root    /usr/sbin/tcpd  in.telnetd -h
```

7.3.2 ssh-Konfiguration

ssh ist ein Ersatz für Programme wie telnet und rlogin. Alle Sitzungen sind dabei ab dem ersten Verbindungsaufbau verschlüsselt, noch bevor der Benutzer ein Passwort eingibt. ssh ist für nichtkommerziellen Gebrauch kostenlos verfügbar; es existiert aber auch ein reichhaltiger ausgestattetes kommerzielles Produkt. Informationen zu ssh erhalten Sie von der Firma SSH Communications Security aus Finnland, im Web unter www.ssh.fi. Die nichtkommerzielle Version erhalten Sie in Form von Quellcode, den Sie zunächst übersetzen müssen. Die offizielle Bezugsquelle für den Quellcode ist [ftp://ftp.cs.hut.fi/pub/ssh/](http://ftp.cs.hut.fi/pub/ssh/). Hier erhalten Sie sowohl die ursprüngliche Version 1 wie auch eine neuere Version 2.

Der `sshd`-Dämon läuft normalerweise als selbstständiger Prozess im Hintergrund und wird nicht über `inetd` aufgerufen. SSH Communications Security erklären dazu, dass ansonsten der Austausch der RSA-Schlüssel bei Schlüsselgrößen für den Server-Schlüssel von mehr als 512 Bit (768 Bit ist die Voreinstellung!) länger dauern kann als die Timeout-Wartezeit von `inetd`. Weil die `tcp_wrapper` aber normalerweise nur über `/etc/inetd.conf` eingesetzt werden, enthält `ssh` eingebaute Unterstützung für die `tcp_wrapper`.

Mit den folgenden Befehlen übersetzen und installieren Sie sowohl den ursprünglichen SSH1-Quellcode als auch den neueren SSH2-Code:

```
./configure --with-libwrap  
make all  
make install
```

Bei SSH1 editieren Sie anschließend die Konfigurationsdateien für Server (`/etc/sshd_config`) und Client (`/etc/ssh_config`) entsprechend Ihrer Bedürfnisse. Sammeln Sie RSA-Schlüssel für die Domains, mit deren Hosts Sie Verbindungen herstellen wollen:

```
make-ssh-known-hosts Domainname
```

Damit der `sshd`-Dämon während des Bootvorganges automatisch gestartet wird, bearbeiten Sie abschließend die Datei `/etc/rc.d/rc.local` und hängen die folgende Zeile an:

```
/usr/local/sbin/sshd
```

Haben Sie SSH2 gewählt, so bearbeiten Sie die Konfigurationsdatei für den Server in `/etc/ssh2/sshd2_config` und die für den Client in `/etc/ssh2/ssh2_config`. Die RSA-Schlüssel wurden bereits während des Installationsvorganges erstellt.

Damit der `sshd2`-Dämon beim Booten automatisch aufgerufen wird, hängen Sie an `/etc/rc.d/rc.local` die folgende Zeile an:

```
/usr/local/sbin/sshd2
```

Nähere Informationen zum Einsatz von `ssh` enthalten die Manual-Pages zu `ssh(1)`, `sshd(8)`, `ssh-keygen(1)` und `make-ssh-known-hosts(1)`, bzw. für SSH2 `ssh2(1)`, `sshd2` und `ssh-keygen2`.

7.3.3 SMTP-Konfiguration

In Mailservern wurden immer wieder viele Sicherheitslücken gefunden. Das betrifft sowohl `smtp` und `sendmail` als auch die Server für das Abholen von Mail `pop` und `imap`. Eine sorgfältige Kontrolle, wer auf diese Dienste zugreifen darf, trägt sehr viel zur Absicherung bei. In die Absicherung des aktuellen Mailservers – `sendmail` – ist viel Arbeit investiert worden. Red Hat enthält aktuell die `sendmail`-Version 8.9.3, eines der neuesten Releases.

Trotzdem sind viele der potenziellen Probleme mit `sendmail` an die Konfiguration geknüpft und daran, welche Hosts Mail über den lokalen Server versenden dürfen. Zum Glück ist die voreingestellte `sendmail`-Konfiguration von Red Hat Linux ziemlich sicher. In einer Site mit nur einem System gibt es hier keinen weiteren Handlungsbedarf.

Wenn Sie für ein LAN Mail-Services anbieten, müssen Sie für die Rechner im LAN Mail-Relaying einschalten. Ansonsten akzeptiert der Mailserver keine abgehenden Mails von Ihren eigenen Computern. Relaying aktivieren Sie wahlweise in einer von zwei Dateien, nämlich `/etc/mail/access` oder `/etc/mail/relay-domains`. Ein Beispiel für `/etc/mail/access` in einem kleinen LAN könnte etwa so aussehen:

<code>localhost.localdomain</code>	<code>RELAY</code>
<code>localhost</code>	<code>RELAY</code>
<code>windows.private.lan</code>	<code>RELAY</code>
<code>linux2.private.lan</code>	<code>RELAY</code>
<code>macintosh.private.lan</code>	<code>RELAY</code>

Die meisten Konfigurationsdateien in `/etc/mail` sind Dateipaare: eine ASCII-Datei und eine gehashte Datenbankdatei, die daraus erstellt wird. Wenn Sie eine der ASCII-Dateien bearbeiten, müssen Sie die zugehörige Datenbankdatei aktualisieren. Der folgende Befehl aktualisiert die Hash-Tabelle `/etc/mail/access.db` aus `/etc/mail/access`:

```
makemap hash /etc/mail/access </etc/mail/access
```

Weitere Informationen zur `sendmail`-Konfiguration finden Sie in den Manual-Pages `sendmail(8)`, `aliases(5)` und `newaliases(1)`. Ausführliche Dokumentation zur aktuellen Version liegt unter `/usr/doc/sendmail`. Die allerneuesten Infos sind immer von www.sendmail.org erhältlich.

Hinweis

Einen Mailserver auf Relay-Fähigkeiten prüfen

Es gibt ein Programm namens `rlytest`, das Ihren Mailserver daraufhin überprüft, ob er Mail weiterleitet. Dafür gibt es auch einen öffentlichen Service, und zwar unter <http://maps.vix.com/tsi/ar-test.html>.

`rlytest` erhalten Sie von <http://www.unicom.com/sw/#rlytest>.

Wenn Sie immer die neueste `sendmail`-Version einsetzen und `smrsh` benutzen, dürfen Sie ruhig schlafen. `smrsh` ist eine sichere Shell, die speziell für den Einsatz mit `sendmail` entwickelt wurde.

smrsh

Ihre Benutzer können `sendmail` anweisen, für sie Programme auszuführen. Das `vacation`-Programm ist ein Beispiel; es erzeugt automatische Antworten auf ankommende E-Mails.

`smrsh` ist eine eingeschränkte Shell für `sendmail`. Indem Sie in der `sendmail`-Konfiguration `/bin/sh` durch `/usr/bin/smrsh` ersetzen, dürfen die Benutzer nur noch bestimmte Programme über `sendmail` ausführen lassen. `smrsh` führt nur die Programme aus, die – direkt oder als Link – im Verzeichnis `/etc/smrsh` stehen. Dieses Verzeichnis ist in der Voreinstellung leer. Sie sollten hier vor allem keine Shells, `sed`, `awk`, `perl` oder irgendeinen anderen Interpreter ablegen. Die meisten kleinen Sites kommen gut mit einem leeren Verzeichnis `/etc/smrsh` aus.

Um die Standardshell durch `smrsh` zu ersetzen, bearbeiten Sie `/etc/sendmail.cf` und ändern Sie die Zeile, die mit

```
Mlocal,    P=/bin/sh
```

beginnt, zu

```
Mlocal,    P=/usr/sbin/smrsh,
```

7.3.4 DNS-Konfiguration

DNS, unter Linux als Berkeley Internet Name Domain Service (BIND) zu finden, ist prinzipiell kein besonders sicherer Dienst. Von den potenziellen Problemen mit jedem UDP-basierten Dienst einmal abgesehen, liegen die übrigen Risiken vor allem im Bereich der TCP-Dienste von DNS. Bei den Sicherheitsproblemen stehen die Konfigurationseinstellungen im Vordergrund, die die Beziehungen des Servers mit anderen Servern definieren, sowie die Informationen, die der Server an Clients ausgibt. Die Privatsphäre ist nicht unbedingt gewährleistet, und im Zusammenhang mit DNS-Spoofing oder dem DNS-Cache-Poisoning können Denial-of-Service-Angriffe durchgeführt werden.

Was die Privatsphäre angeht, ist das in erster Linie eine Frage dessen, welche Informationen Sie in Ihrer DNS-Datenbank abspeichern, und wem Sie den Zugriff darauf gestatten. Denial-of-Service-Angriffe hängen davon ab, wer Ihre eigene Datenbank auslesen darf, von wem Sie Zone-Informationen übernehmen, und wer überarbeitete Versionen Ihrer Datenbank einspielen darf.

Weitere Informationen zu DNS und BIND finden Sie in den Man-Pages zu `named(8)`, `resolver(5)` und `hostname(7)`; die offizielle Dokumentation zu BIND 8 liegt in `/usr/doc/bind-8.2/html/index.html`; empfehlenswert ist das Buch *DNS & BIND* von Albitz und Liu (bei O'Reilly); es gibt auch ein DNS-HOWTO.

`/etc/resolv.conf`

Der *Resolver* ist der Client-Anteil des DNS. Dabei handelt es sich aber nicht um ein konkretes ausführbares Programm, sondern um einen Teil der C-Bibliotheken,

die mit jedem Programm verlinkt werden, das Zugang zum Netzwerk benötigt. Der Resolver-Code ist der Teil der Bibliotheken, der in der Praxis die DNS-Anfragen an den Nameserver abschickt. Er wird über die Datei `/etc/resolv.conf` konfiguriert.

Ein System ohne eigenen `named` braucht hierin zwei Direktiven: `domain` und `nameserver`. `domain` ist der lokale Domainname. Dazu können bis zu drei `nameserver`-Direktiven angegeben werden, wobei jede auf einen bestimmten DNS-Host zeigt. Wenn Sie mehrere Nameserver verwenden, können Sie die Option `rotate` setzen, um sie abwechselnd abzufragen, statt jedesmal zuerst den ersten Nameserver aus der Liste anzusprechen. `resolv.conf` könnte beispielsweise den Domainnamen Ihres Providers sowie die IP-Adressen von bis zu drei seiner Nameserver enthalten:

```
domain my.isp.net
nameserver 192.168.47.81
nameserver 192.168.60.7
nameserver 192.168.60.8
option rotate
```

Ein System mit lokal laufendem `named` benötigt nur eine einzige `nameserver`-Direktive, die auf die eigene Maschine zeigt. `resolv.conf` könnte in diesem Fall etwa so aussehen:

```
domain my.isp.net
nameserver 127.0.0.1
```

Auf einem Computer, auf dem Sie einen `named` für Ihr LAN betreiben, könnten Sie die Direktive `domain` durch `search` ersetzen. Als Argumente geben Sie in diesem Fall eine Liste von Domains an, deren Nameserver abgefragt werden sollen. Wieder ein Beispiel für `resolv.conf`:

```
search my.local.lan my.isp.net
nameserver 127.0.0.1
```

Resource-Records für die Master-Dateien von BIND

Wenn `named` läuft, kann er autoritativ für eine *Zone* (einen Teil des Namespace einer Domain) sein. Eine Master-Zone-Datei definiert die Eigenschaften einer Zone, für die der Nameserver autoritativ ist. Die Datei enthält Kontrollinformationen über die Zone sowie Resource-Records, die die Abbildung von IP-Adressen auf Hostnamen für die Hosts der Zone beschreiben. Im Fall eines nicht vernetzten Rechners oder eines Systems, auf dem `named` alle Anfragen weiterleiten soll, wäre `named` autoritativ für den `localhost`.

Die Zone-Dateien liegen unter `/var/named`. Die Dateinamen können beliebig vergeben werden. Das Paket `caching-nameserver` von Red Hat Linux enthält eine Datei `named.local`, in der die obligatorische `localhost`-Zone beschrieben wird. Diese ist praktisch identisch mit dem folgenden Beispiel, einer Zone-Datei für

localhost. Die Datei heißt `/var/named/named.127.0.0` und sollte auf jedem Nameserver vorhanden sein, damit er Anfragen nach sich selbst auflösen kann:

```
1. 0.0.127.in-addr.arpa.  IN  SOA  localhost.  root.localhost. (
2.                        1      ; Seriennummer
3.                        28800   ; Refresh
4.                        14400   ; Retry
5.                        3600000 ; Expire
6.                        604800  ; default_ttl
7.                        )
8.      IN      NS      localhost.
9. 1  IN      PTR      localhost.
```

Die Zeilen eins bis sieben enthalten die Kontrollinformation für die Zone, die Zeilen acht und neun sind Resource-Records.

- In Zeile 1 beginnt ein SOA-Eintrag (Start of Authority) für die Zone. `0.0.127.in-addr.arpa.` ist der Origin der Zone. Weil Origin und Domain identisch sind, könnte der Origin auch als `@` bezeichnet werden. `IN` besagt, dass die Daten innerhalb dieses Eintrags zur Klasse Internet gehören (und nicht z.B. zur Klasse Hesiod). `SOA` besagt, dass es sich um einen SOA-Eintrag handelt. `localhost.` ist der Name der Domain, `root.localhost.` ist die E-Mail-Adresse der Kontaktperson, die für diese Zone-Information zuständig ist. Die öffnende Klammer steht für den Beginn eines Eintrags, der sich über mehrere Zeilen erstreckt.
- Zeile 2 ist die Seriennummer. Wenn es sekundäre Nameserver gäbe, würden diese an einer Änderung der Seriennummer merken, dass sich die Daten für die Zone geändert haben. Sie müssten dann ihre Kopien der Datenbank aktualisieren.
- Zeile 3 gibt an, nach wie vielen Sekunden sekundäre Nameserver die Seriennummer überprüfen sollen. In unserem Beispiel würden etwaige sekundäre Nameserver alle acht Stunden nachsehen.
- Zeile 4 gibt an, wie viele Sekunden ein sekundärer Nameserver abwarten soll, wenn ein Kontaktversuch zur Überprüfung der Seriennummer fehlschlägt. In diesem Beispiel würden die sekundären Server alle vier Stunden einen Kontaktversuch unternehmen.
- Zeile 5 gibt an, nach wie vielen Sekunden ein sekundärer Server die Informationen über die Zone aus seinem Cache löscht, wenn es ihm nach dieser Zeit nicht gelungen ist, mit dem primären Server zu sprechen. In diesem Beispiel löscht der sekundäre Server die Zone-Daten nach 41 Tagen, wenn so lange keine Überprüfung der Seriennummer möglich war.
- Zeile 6 gibt an, wie viele Sekunden ein fremder Server Daten aus unserer Zone zwischenspeichern darf (*Time-to-Live*). Freilich wird kein fremder Server nach

Daten zu `localhost` fragen, aber falls doch, wäre die Time-to-Live in diesem Beispiel eine Woche.

- Zeile 7 enthält die schließende Klammer für den SOA-Eintrag.
- Zeile 8 ist ein Nameserver-Eintrag (NS) und zeigt an, dass `localhost` der Nameserver für diese Domain ist. Bei allen Einträgen nach dem SOA wird `@` (`0.0.127.in-addr.arpa`) als Wert für das erste Feld angenommen.
- Zeile 9 ist ein Pointer- bzw. Zeiger-Eintrag (PTR) und zeigt an, dass der zugehörige symbolische Name für die Adresse `127.0.0.1` `localhost` ist. Die führende 1 in diesem Eintrag steht für die Adresse `127.0.0.1`. Weil der Origin immer im alten Stil als Arpanet-Domain angegeben wird, wobei die IP-Adresse in umgekehrter Reihenfolge am Anfang steht, ist die führende 1 eine Kurzform für `1.0.0.127.in-addr.arpa..`

/etc/named.conf

`named` ist der DNS-Server. Er liefert zu DNS-Anfragen entweder eine Antwort aus dem eigenen Cache, oder er leitet sie an einen anderen Nameserver weiter. `/etc/named.conf` ist seine Konfigurationsdatei.

Dateiname und Format von `/etc/named.conf` sind in der BIND-Version 8.2, die auch in Red Hat 6.0 enthalten ist, neu.

Hinweis

Update von `named.boot` auf `named.conf`

In älteren BIND-Versionen hieß die Konfigurationsdatei `/etc/named.boot`. Ein Perlskript namens `/usr/doc/bind-8.2/named-bootconf/Grot/named-bootconf.pl` konvertiert eine bestehende `named.boot`-Datei ins neue `named.conf`-Format:

```
cd /usr/doc/bind-8.2/named-bootconf/Grot
perl ./named-bootconf.pl </etc/named.boot >/tmp/named.conf
```

In `named.conf` sind viele Konfigurationsdirektiven erlaubt. Eine vollständige Liste mit Beschreibungen finden Sie in `/usr/doc/bind-8.2/html/config.html`. Hier werden nur die in den Beispielen der folgenden Abschnitte verwendeten Direktiven besprochen.

Konfiguration eines lokalen Nameservers, der alle Anfragen weiterleitet

In Kapitel 3 wurde eine DNS-Server-Konfiguration beschrieben, bei der auf der Firewall ein Nameserver für den lokalen Gebrauch lief. Dieser erlaubte anderen Hosts aus dem Internet keine Zugriffe und war für keine öffentliche Domain autoritativ. Er leitete lediglich Anfragen an einen Nameserver des Internetproviders weiter bzw. stellte die entsprechenden Informationen direkt aus seinem Zwischenspeicher zur Verfügung. In diesem Abschnitt finden Sie Beispiele für `/etc/`

resolv.conf, die Datenbank in /var/named/ sowie /etc/named.conf für solch einen Nameserver.

Konfiguration des lokalen Nameservers

Lokale DNS-Clients benutzen localhost als Nameserver. /etc/resolv.conf enthält:

```
domain my.isp.net
nameserver 127.0.0.1
```

Außer der oben bereits besprochenen Zone-Datei für localhost (/var/named/named.127.0.0) werden keine weiteren Zone-Daten benötigt.

Die Konfigurationsdatei für den lokalen named-Server – /etc/named.conf – sieht wie folgt aus:

```
1. options {
2.     directory "/var/named";
3.     forward only;
4. //     forward first;
5.     forwarders {
6.         my.name.server.1;
7.         my.name.server.2;
8.         my.name.server.3;
9.     };
10.
11.     query-source address * port 53;
12.
13.     allow-query {
14.         127.0.0.1;
15. //     192.168.1/24;
16.     };
17.     listen-on port 53 {
18.         127.0.0.1;
19. //     192.168.1.1;
20.     };
21. };
22.
23. zone "0.0.127.in-addr.arpa" {
24.     type master;
25.     notify no;
26.     file "named.127.0.0";
27. };
28.
29. zone "." {
30.     type hint;
31.     file "root.cache";
32. };
33. }
```

Die hier gezeigte `/etc/named.conf` enthält zwei verschiedene Eintragsarten: die `options` sowie die `zone`-Einträge. Erstere definieren Optionen, die für den gesamten Server gelten; zweitere enthalten Einstellungen, die nur für die jeweilige Zone gelten.

Zeilen 1-19 enthalten einen `options`-Eintrag:

- Zeile 1 deklariert den Eintragstyp als `options` und leitet mit einer öffnenden Klammer einen mehrzeiligen Eintrag ein.
- Zeile 2 definiert das Arbeitsverzeichnis des Nameservers: `/var/named`. Hier stehen die Master-Files für die Zonen.
- Zeile 3 weist den Server an, ausschließlich als Forwarding-Nameserver zu arbeiten, also alle Anfragen ausschließlich an die Hosts in der `forwarders`-Direktive weiterzuleiten.
- Zeile 4 enthält eine auskommentierte Alternative zur `forward only`-Direktive. Bei `forward first` leitet der Server Anfragen zunächst an die unter `forwarders` definierten Hosts weiter. Wenn diese keine Auskunft geben können oder nicht antworten, versucht der Server, die Anfrage wie ein regulärer Nameserver selbst aufzulösen.
- In Zeile 5 beginnt mit einer öffnenden Klammer ein mehrzeiliger Eintrag. Die `forwarders`-Direktive enthält eine Liste fremder Nameserver, an die Anfragen weitergeleitet werden.
- Die Zeilen 6 bis 8 enthalten die fremden Nameserver. Sie dürfen bis zu drei Server angeben.
- Mit der schließenden Klammer aus Zeile 9 endet die `forwarders`-Direktive.
- Zeile 10 ist nötig, wenn zwischen dem Nameserver und dem Internet eine Firewall steht. Die Zeile definiert den Absender-Port für Anfragen an andere Server. D.h. der Server wird jetzt den UDP-Port 53 sowohl als Absender- als auch als Empfängerport für Anfragen an andere Server benutzen.
- In Zeile 11 beginnt mit der öffnenden Klammer ein `allow-query`-Eintrag, der sich wieder über mehrere Zeilen erstreckt. Er enthält eine Liste der IP-Adressen, die Anfragen an diesen Server stellen dürfen.
- Zeile 12 legt `localhost` als einen der Hosts fest, die den Server abfragen dürfen.
- Die Zeile 13 ist auskommentiert; sie enthält ein weiteres Netzwerk, von dem Anfragen angenommen werden. Wenn an den Nameserver über ein internes Netzwerkinterface noch ein LAN angeschlossen ist, soll er vermutlich auf Anfragen aus dem LAN reagieren.
- Mit der schließenden Klammer in Zeile 14 endet die `allow-query`-Direktive.

- In Zeile 15 beginnt erneut ein mehrzeiliger Eintrag. Diesmal handelt es sich um die Direktive `listen-on`, die den Port definiert, auf dem der Server auf Clientanfragen wartet. Die Direktive enthält zusätzlich eine Liste lokaler Netzwerkinterfaces.
- Zeile 16 instruiert den Server, dass Anfragen vom `loopback`-Interface erlaubt sind.
- Die auskommentierte Zeile 17 definiert ein weiteres Netzwerkinterface, auf dem der Server auf Anfragen warten soll. Wenn ein LAN über ein internes Netzwerkinterface angeschlossen wäre, würden Sie hier die IP-Adresse des internen Interfaces angeben.
- Zeile 18 beendet mit einer rechten Klammer die `listen-on`-Direktive.
- Die rechte Klammer aus Zeile 19 schließt den `options`-Eintrag ab.
- Mit Zeile 20 beginnt ein mehrzeiliger `zone`-Eintrag für das Loopback-Netz. Domainnamen für Zonen werden im Arpanet-Stil angegeben, d.h. die Netzwerkadresse von `loopback` (127.0.0) erscheint hier in umgekehrter Reihenfolge als `0.0.127.in-addr.arpa`.
- Zeile 21 zeichnet die in diesem Eintrag beschriebenen Zone-Daten als Master-Kopie aus. Dieser Nameserver ist also für diese Zone autoritativ.
- Zeile 22 gibt an, dass bei einer Änderung der Daten dieser Zone keine weiteren Nameserver unterrichtet werden müssen.
- Zeile 23 enthält den Namen der Datenbankdatei für die Zone. Weil `named.127.0.0` ein relativer Pfadname ist, wird er relativ zu `/var/named` interpretiert, der `directory`-Direktive aus dem `options`-Eintrag.
- Zeile 24 beendet den `zone`-Eintrag mit einer schließenden Klammer.
- Zeile 25 leitet einen mehrzeiligen `zone`-Eintrag für den Cache der Root-Domain ein. Dieser enthält die Root-Nameserver des Internets. Ein `forward only`-Nameserver benötigt die Root-Zone überhaupt nicht, weil alle Anfragen ausschließlich an die Hosts aus der `forwarders`-Direktive weitergeleitet werden. Ein `forward first`-Nameserver hingegen kommt ebenso wenig wie ein regulärer Nameserver ohne diese Daten aus; beide beginnen über die hier angegebenen Rechner mit der Suche nach autoritativen Nameservern.
- Zeile 26 deklariert die Zone-Daten dieses Eintrags als *Hint* oder Tipp, d.h. es handelt sich nur um einen Ausgangspunkt für die weitere Suche.
- Zeile 27 enthält den Namen der Datei mit der Zonendatenbank. Weil `root.cache` ein relativer Dateiname ist, wird er relativ zu `/var/named` aus dem `options`-Eintrag interpretiert.
- Zeile 28 schließt den `zone`-Eintrag mit einer rechten Klammer.

Hinweis***Namenskonventionen und Bezugsquellen für root.cache***

Bei Red Hat-Distributionen finden Sie die Datei mit dem Cache der Root-Nameserver in /var/named/named.ca. Für die hier angeführten Beispiele habe ich die Datei entsprechend dem DNS-HOWTO in root.cache umbenannt. Eine Kopie erhalten Sie auch von ftp.rs.internic.net als /domain/named.root.

DNS-Konfiguration der LAN-Clients

Die Client-Maschinen in einem kleinen internen LAN benötigen keine eigenen Nameserver. Stattdessen benutzen sie den Rechner mit dem lokalen Nameserver. /etc/resolv.conf sieht dann wie folgt aus:

```
domain my.private.lan  
nameserver 192.168.1.1
```

Das ist alles, was ein lokaler Client für DNS-Anfragen benötigt.

Falls Sie weitere Nameserver auf Geräten im LAN benötigen, könnten Sie lokale Anfragen an den Master-Server, an die Nameserver Ihres Internet-Providers oder an die anderen lokalen Server weiterleiten.

Der nächste Abschnitt beschreibt eine weitere Konfigurationsmöglichkeit, wenn Sie auf einer internen Maschine einen lokalen Nameserver betreiben. Dabei ist der öffentliche Nameserver nur ein Dummy, der aber behauptet, er sei autoritativ. Der wirkliche autoritative Server für das interne, private LAN ist der interne DNS-Server, der für das Internet unsichtbar bleibt.

Konfiguration einer klassischen Kombination aus öffentlichem und privatem Nameserver

In Kapitel 4 wurde eine DNS-Konfiguration beschrieben, in der auf dem Computer mit der Firewall ein öffentlicher Nameserver läuft und auf einem internen Gerät ein privater. Der öffentliche Server spiegelt vor, er sei für die Domain autoritativ, weiß aber in Wirklichkeit nichts über die Computer im LAN. Seine wahre Aufgabe ist es, Anfragen an externe DNS-Server weiterzuleiten und lokale DNS-Informationen zu verbergen. Der private Server ist tatsächlich autoritativ für die Domain des privaten LANs und stellt den dort gelegenen Maschinen DNS-Dienste zur Verfügung.

In diesem Abschnitt finden Sie /etc/resolv.conf, die Dateien aus /var/named sowie /etc/named.conf, und zwar sowohl für den öffentlichen wie auch für den privaten Nameserver.

Konfiguration des öffentlichen Nameservers

Der öffentliche Nameserver behauptet, er sei für die lokale Domain autoritativ, besitzt in Wirklichkeit aber nur wenig oder gar keine Informationen über das interne LAN. In einem Privathaushalt, wo die Firewall an das Netzwerk eines

Internet-Providers angeschlossen ist, hat es keine Bedeutung, ob der Server autoritativ ist. Es gibt hier wohl kaum einen Grund, Anfragen von fremden Hosts zuzulassen. Anders ist es bei einer kleinen Firma mit mehreren öffentlichen IP-Adressen: Hier wird man möglicherweise ein gewisses Maß an lokaler Information veröffentlichen wollen. Zur Illustration erlaubt unser öffentlicher Server Anfragen von außen.

Weil der öffentliche Server nichts über das LAN weiß, benutzen die lokalen DNS-Clients auf der gleichen Maschine ihn nicht. Stattdessen greifen sie auf den privaten Nameserver zurück, der auf einer internen Maschine läuft. In unserem Beispiel ist diese interne Maschine die Choke-Firewall, die in Kapitel 4 beschrieben wurde. `/etc/resolv.conf` enthält:

```
search my.local.lan my.isp.net
nameserver 192.168.11.2
```

Die oben besprochene Datei für die Zone `localhost (/var/named/named.127.0.0)` wird immer benötigt.

Nehmen wir an, dass die Site einen Adressblock im Bereich 192.168.10.0 und 192.168.11.0 besitzt, dass die externe IP-Adresse der Firewall 192.168.10.30 ist und dass das DMZ-Netzwerk innerhalb von 192.168.11.0 liegt.

Neben der Datei für `localhost` stehen noch zwei weitere Zonendateien in `/var/named`. Die erste ist die öffentliche Zuordnung von IP-Adressen zu symbolischen Namen, `named.public`:

```
1. 10.168.192.in-addr.arpa. IN SOA my.domain.com.
   postmaster.my.domain.com. (
2.                               1999072701 ; Seriennummer
3.                               28800      ; Refresh: acht Stunden
4.                               14400      ; Retry: vier Stunden
5.                               3600000    ; Expire: ca. 41 Tage
6.                               86400     ) ; Minimum

7.      IN      NS      bastion.my.domain.com.
8.      IN      MX      10 bastion.my.domain.com.

9. 30      IN      PTR      bastion.my.domain.com.
```

Die Zeilen eins bis sechs enthalten die Kontrollinformationen über die Zone, Zeilen sieben bis neun die eigentlichen Ressourcen.

- In Zeile 1 beginnt der SOA-Eintrag für die Zone. `10.168.192.in-addr.arpa` ist der Origin der Zone. `IN` bedeutet, dass die Daten in diesem Eintrag sich auf die Datenklasse »Internet« beziehen. `SOA` steht für »Start of Authority«, d.h. die Kontrollinformationen beginnen hier. `my.domain.com.` ist der Domainname, `postmaster.my.domain.com.` deutet auf die E-Mail-Adresse des Verantwortli-

chen für die Zoneninformation hin. Die öffnende Klammer leitet einen mehrzeiligen Eintrag ein.

- Zeile 2 ist eine Seriennummer. Wenn Sie sekundäre Nameserver betreiben, ist eine Änderung der Seriennummer ein Anzeichen für eine Änderung der Zonendaten und veranlasst die sekundären Server dazu, ihre Kopien der Zonendatenbank zu aktualisieren. Im Beispiel folgt die Seriennummer einer häufig angewandten Konvention und besteht aus Jahr, Monat und Tag sowie der Zahl der Änderungen an diesem Datum. 1999072701 bezeichnet also die erste Version vom 27.7.1999.
- Zeile 3 gibt an, nach wie vielen Sekunden sekundäre Nameserver die Seriennummer überprüfen sollen. In unserem Beispiel würden etwaige sekundäre Nameserver alle acht Stunden nachsehen.
- Zeile 4 gibt an, wie viele Sekunden ein sekundärer Nameserver abwarten soll, wenn ein Kontaktversuch zur Überprüfung der Seriennummer fehlschlägt. In diesem Beispiel würden die sekundären Server alle vier Stunden einen Kontaktversuch unternehmen.
- Zeile 5 gibt an, nach wie vielen Sekunden ein sekundärer Server die Informationen über die Zone aus seinem Cache löscht, wenn es ihm nach dieser Zeit noch nicht gelungen ist, mit dem primären Server zu sprechen. In diesem Beispiel löscht der sekundäre Server die Zonendaten nach ungefähr 41 Tagen, wenn so lange keine Überprüfung der Seriennummer möglich war.
- Zeile 6 ist die so genannte *Time-to-Live* und zeigt an, wie lange fremde Server Informationen zwischenspeichern dürfen. In diesem Fall ist das eine Woche.
- Zeile 7 enthält einen Nameserver-Eintrag (NS) und bezeichnet die öffentliche Firewall `bastion.my.domain.com.` als Nameserver für diese Domain.
- Zeile 8 ist ein Mailserver-Eintrag (Mail-Exchanger MX). Die Maschine `bastion.my.domain.com.` nimmt also Mail für diese Domain an oder leitet sie zumindest weiter. Der Wert 10 ist eine Prioritätswertung und hat in unserem Beispiel keine Bedeutung. Wenn mehrere Mailserver angegeben sind, erhält jeder davon eine Priorität zwischen 0 und 65535. Ankommende Mail wird nacheinander an jeden Host aus der Liste zugestellt, beginnend mit der niedrigsten Priorität, bis sie erfolgreich angenommen wird.
- Zeile 9 ist ein Pointer- bzw. Zeiger-Eintrag (PTR) und beschreibt für den Host `bastion.my.domain.com.` die Zuordnung von der IP-Adresse zum symbolischen Namen. Die 30 am Anfang der Zeile steht für die Adresse 192.168.10.30.

Für Reverse-Lookups, bei denen einer IP-Adresse ein Hostname zugeordnet wird, benötigt man noch eine zweite Zone-Datei, `/var/named/named.public.reverse`. Die ersten acht Zeilen einschließlich SOA, NS und MX sind dabei identisch zu der gerade eben beschriebenen Datei. Statt PTR steht diesmal der neue Typ A für Adresse:

Kapitel 7 • Ergänzende Maßnahmen durch die Unix-Systemadministration **347**

```
1. 10.168.192.in-addr.arpa. IN SOA my.domain.com.  
   postmaster.my.domain.com. (  
2.                               1999072701 ; Seriennummer  
3.                               28800      ; Refresh: acht Stunden  
4.                               14400      ; Retry: vier Stunden  
5.                               3600000    ; Expire: ca. 41 Tage  
6.                               86400 )    ; Minimum  
  
7.      IN      NS      bastion.my.domain.com.  
8.      IN      MX      10 bastion.my.domain.com.  
  
9. bastion.my.domain.com. IN      A      192.168.10.30
```

Zeile 9 enthält einen A- bzw. Adress-Eintrag. Dem Hostnamen `bastion.my.domain.com.` ist die Adresse `192.168.10.30` zugeordnet.

Die Konfigurationsdatei des `named-Servers`, `/etc/named.conf`, enthält Folgendes:

```
1. options {  
2.     directory "/var/named";  
3.     forward first;  
4.     forwarders {  
5.         my.name.server.1;  
6.         my.name.server.2;  
7.         my.name.server.3;  
8.     };  
  
9.     query-source address * port 53;  
  
10.    allow-query {  
11.        ! 127/8;  
12.        192.168.11.2;  
13.        ! 192.168.11/24;  
14.        *;  
15.    };  
  
16.    allow-transfer { ! *; };  
17.    allow-update { ! *; };  
  
18.    listen-on port 53 {  
19.        192.168.10.30;  
20.        192.168.11.1;  
21.    };  
22. };  
  
23. zone "0.0.127.in-addr.arpa" {  
24.     type master;  
25.     notify no;
```

```
26.         file "named.127.0.0";
27. };

28. zone "my.domain.com" {
29.     type master;
30.     notify no;
31.     file "named.public.reverse";
32. };

33. zone "10.168.192.in-addr.arpa" {
34.     type master;
35.     notify no;
36.     file "named.public";
37. };

38. zone "." {
39.     type hint;
40.     file "root.cache";
41. };
```

Die Datei `/etc/named.conf` enthält hier zwei Eintragsarten, den `options`-Eintrag sowie die vier `zone`-Einträge. Ersterer enthält Einstellungen, die für den gesamten Server gelten, während die Direktiven innerhalb der `zone`-Einträge nur für diese eine Zone Gültigkeit besitzen.

- Zeile 1 beginnt mit einer öffnenden Klammer einen mehrzeiligen `options`-Eintrag.
- Zeile 2 definiert das Arbeitsverzeichnis des Servers, in dem die Masterdateien für die einzelnen Zonen liegen: `/var/named`.
- Zeile 3 schaltet den Server in den `forward first`-Modus. Dabei leitet er Anfragen, die er aus dem Cache nicht auflösen kann, zunächst an einen der unter `forwarders` angegebenen Hosts weiter. Wenn diese Nameserver die Anfrage ebenfalls nicht auflösen oder überhaupt nicht antworten, findet er die Antwort wie ein regulärer Nameserver selbst heraus.
- Die öffnende Klammer aus Zeile 4 beginnt eine mehrzeilige `forwarders`-Direktive, in der die Nameserver aufgelistet sind, an die wir Anfragen weiterleiten.
- Zeilen 5 bis 7 enthalten diese Nameserver, wobei bis zu drei Server erlaubt sind.
- Zeile 8 beendet die `forwarders`-Direktive mit einer schließenden Klammer.
- Wenn eine Firewall zwischen dem Nameserver und dem Internet steht, wird Zeile 9 benötigt. Sie legt den Absender-Port für Anfragen an andere Nameserver auf 53 fest. D.h. unser Server benutzt jetzt den UDP-Port 53 sowohl als Absender- als auch als Empfänger-Port für Anfragen an fremde DNS-Server.

- Zeile 10 beginnt eine mehrzeilige `allow-query`-Direktive, mit der IP-Adressen festgelegt werden, von denen aus der Server Anfragen entgegennehmen oder abweisen soll. Die Reihenfolge ist dabei wichtig: Die Adressliste wird in der angegebenen Reihenfolge durchsucht; die erste passende Regel gilt.
- Zeile 11 instruiert den Server, dass Anfragen von `localhost` verboten sind. Lokale Anfragen gehen ja an den privaten, internen Nameserver.
- Zeile 12 teilt dem Server mit, dass er Anfragen des internen Nameservers von der IP 192.168.11.2 akzeptieren soll. Wenn der öffentliche Server die nachgefragten Informationen nicht in seinem Cache hat, leitet er die Anfrage an fremde Nameserver weiter.
- Zeile 13 verbietet Anfragen von allen anderen Hosts im internen LAN. Interne Anfragen gehen an den privaten, internen Nameserver.
- Zeile 14 erlaubt Anfragen von überall sonst. Diese pauschale Erlaubnis steht nach der Liste der verbotenen Adressen und gestattet damit Anfragen von allen außer den explizit gesperrten Adressen.
- Zeile 15 beendet die `allow-query`-Direktive mit einer schließenden Klammer.
- Zeile 16 enthält eine einzeilige `allow-transfer`-Direktive, die mit Hilfe der Negation ! alle Zone-Transfers verbietet.
- Zeile 17 enthält eine einzeilige `allow-update`-Direktive, die mit Hilfe der Negation ! alle Zonen-Aktualisierungen verbietet. Das ist zwar ohnehin die Voreinstellung, aber die explizite Angabe der Direktive in der Konfigurationsdatei dokumentiert das erwartete Verhalten und liefert zusätzliche Sicherheit, falls irgendwo ein Fehler bestehen sollte.
- Zeile 18 beginnt mit einer linken Klammer eine mehrzeilige `listen-on`-Direktive. Diese definiert den Port sowie eine Liste von Netzwerkkinterfaces, auf denen der Server auf Anfragen wartet.
- Zeile 19 lässt den Server auf dem externen Interface ins Internet – 192.168.10.30 – auf Anfragen warten.
- Zeile 20 lässt den Server auf dem internen Interface zum LAN – 192.168.11.1 – auf Anfragen warten.
- Zeile 21 schließt mit einer rechten Klammer die `listen-on`-Direktive.
- Zeile 22 beendet mit einer rechten Klammer den `options`-Eintrag.
- Zeile 23 beginnt einen mehrzeiligen `zone`-Eintrag für das `loopback`-Netz. Zonnennamen werden im `Arpanet`-Stil angegeben, sodass die `loopback`-Adresse 127.0.0 hier als `0.0.127.in-addr.arpa` auftaucht.
- Zeile 24 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.

- Zeile 25 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 26 enthält den Namen der Datei mit den Zonendaten. Weil `named.127.0.0` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 27 schließt den `zone`-Eintrag mit einer rechten Klammer.
- Zeile 28 beginnt einen mehrzeiligen `zone`-Eintrag für die öffentliche Domain `my.domain.com`. Die hier beschriebenen Zonendaten gelten für Anfragen, bei denen einem symbolischen Namen eine IP-Adresse zugeordnet werden soll.
- Zeile 29 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 30 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 31 enthält den Namen der Datei mit den Zonendaten. Weil `named.public.reverse` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 32 schließt den `zone`-Eintrag mit einer rechten Klammer.
- Zeile 33 beginnt einen mehrzeiligen `zone`-Eintrag für die öffentliche Domain `10.168.192.in-addr.arpa`. Die Daten gelten für Anfragen, bei denen einer IP-Adresse ein symbolischer Name zugeordnet werden soll.
- Zeile 34 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 35 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 36 enthält den Namen der Datei mit den Zonendaten. Weil `named.public` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 37 schließt den `zone`-Eintrag mit einer rechten Klammer ab.
- Zeile 38 leitet einen mehrzeiligen `zone`-Eintrag für den Cache der Root-Domain ein. Dieser enthält die Root-Nameserver des Internets. Ein `forward first`-Nameserver kommt ebenso wenig wie ein regulärer Nameserver ohne diese Daten aus; beide beginnen über die hier angegebenen Rechner mit der Suche nach autoritativen Nameservern.
- Zeile 39 deklariert die Zone-Daten dieses Eintrags als *Hint* oder Tipp, d.h. es handelt sich nur um einen Ausgangspunkt für die weitere Suche.

- Zeile 40 enthält den Namen der Datei mit den Zonendaten. Weil `root.cache` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 41 schließt den `zone`-Eintrag mit einer rechten Klammer ab.

Konfiguration des privaten Nameservers

Der private Nameserver ist für zwei lokale Domains autoritativ: für das interne, private LAN (`my.local.lan`) sowie für das DMZ-LAN (`my.firewall.lan`) zwischen der Bastion- und der Choke-Firewall. Lokale Clients aus dem LAN und von der Bastion greifen auf den privaten Nameserver zu. In unserem Beispiel ist dieser identisch mit der Choke-Firewall aus Kapitel 4. Seine `/etc/resolv.conf` enthält:

```
search my.local.lan my.firewall.lan
nameserver 127.0.0.1
```

Die Zonendatei für `localhost`, `/var/named/named.127.0.0`, wird immer benötigt.

Im Beispiel liege die Site in den Netzen `192.168.10.0` und `192.168.11.0`. Die externe IP-Adresse der Bastion sei `192.168.10.30`, die interne `192.168.11.1`. Die DMZ liege im Netzwerk `192.168.11.0`. Das externe Interface der Choke sei `192.168.11.2`.

Zwei weitere Zonendatei-Paare liegen in `/var/named`. Das erste Paar enthält die Namen und IP-Adressen der DMZ, das zweite die Namen und IPs des privaten LANs.

Die Zuordnung von IP-Adressen der LANs zu den entsprechenden symbolischen Namen ist in `/var/named/named.dmz` gespeichert. Diese Datei enthält:

```
1. 11.168.192.in-addr.arpa. IN SOA my.dmz.lan.
   postmaster.my.dmz.lan. (
2.                               1999072701 ; Seriennummer
3.                               28800      ; Refresh: acht Stunden
4.                               14400      ; Retry: vier Stunden
5.                               3600000    ; Expire: ca. 41 Tage
6.                               86400     ) ; Minimum

7.      IN      NS      choke.my.dmz.lan.
8.      IN      MX      10 bastion.my.dmz.lan.

9. 1      IN      PTR      bastion.my.dmz.lan.
10. 2     IN      PTR      choke.my.dmz.lan.
```

Die Zeilen 1–6 enthalten die Kontrollinformationen der Zone, Zeilen 7–10 beschreiben Ressourcen.

- Zeile 1 beginnt einen SOA-Eintrag für die Zone. `11.168.192.in-addr.arpa` ist der Origin. `IN` besagt, dass es sich um einen Eintrag aus der Datenklasse `Internet` handelt. `SOA` steht für `Start of Authority`, also den Beginn der Kontrollinformationen. `my.dmz.lan.` ist der Domainname und `postmaster.my.dmz.lan.` deutet auf die E-Mail-Adresse des Verantwortlichen hin. Die linke Klammer beginnt einen mehrzeiligen Eintrag.
- Zeile 2 enthält die Seriennummer. Wenn Sie sekundäre Nameserver betreiben, ist eine Änderung der Seriennummer ein Anzeichen für eine Änderung der Zonendaten und veranlasst die sekundären Server zu einer Aktualisierung ihrer Kopien der Zonendatenbank. Im Beispiel folgt die Seriennummer einer häufig angewandten Konvention und besteht aus Jahr, Monat und Tag sowie der Zahl der Änderungen an diesem Datum. `1999072701` bezeichnet also die erste Version vom 27.7.1999.
- Zeile 3 gibt an, nach wie vielen Sekunden sekundäre Nameserver die Seriennummer überprüfen sollen. In unserem Beispiel würden etwaige sekundäre Nameserver alle acht Stunden nachsehen.
- Zeile 4 gibt an, wie viele Sekunden ein sekundärer Nameserver abwarten soll, wenn ein Kontaktversuch zur Überprüfung der Seriennummer fehlschlägt. In diesem Beispiel würden die sekundären Server alle vier Stunden einen Kontaktversuch unternehmen.
- Zeile 5 gibt an, nach wie vielen Sekunden ein sekundärer Server die Informationen über die Zone aus seinem Cache löscht, wenn es ihm nach dieser Zeit noch nicht gelungen ist, mit dem primären Server zu sprechen. In diesem Beispiel löscht der sekundäre Server die Zonendaten nach ungefähr 41 Tagen, wenn so lange keine Überprüfung der Seriennummer möglich war.
- Zeile 6 ist die so genannte *Time-to-Live* und zeigt an, wie lange fremde Server Informationen zwischenspeichern dürfen. In diesem Fall ist das eine Woche.
- Zeile 7 enthält einen Nameserver-Eintrag (`NS`) und bezeichnet die interne Firewall `choke.my.dmz.lan.` als Nameserver für diese Domain.
- Zeile 8 ist ein Mailserver-Eintrag (Mail-Exchanger `MX`). Die Maschine `bastion.my.dmz.lan.` nimmt also Mail für diese Domain an oder leitet sie zumindest weiter. Der Wert 10 ist eine Prioritätswertung und hat in unserem Beispiel keine Bedeutung. Wenn mehrere Mailserver angegeben sind, erhält jeder davon eine Priorität zwischen 0 und 65535. Ankommende Mail wird nacheinander an jeden Host aus der Liste zugestellt, beginnend mit der niedrigsten Priorität, bis sie erfolgreich angenommen wird.
- Zeile 9 ist ein Pointer- bzw. Zeiger-Eintrag (`PTR`) und beschreibt für den Host `bastion.my.dmz.lan.` die Zuordnung von der IP-Adresse zum symbolischen Namen. Die 1 am Anfang der Zeile steht für die Adresse 192.168.11.1.

- Zeile 10 ist ebenfalls ein PTR-Eintrag und beschreibt den Host `choke.my.dmz.lan`. Die 2 am Anfang der Zeile steht für die Adresse 192.168.11.2.

Die Übersetzung von DMZ-Hostnamen zu IP-Adressen ist in der Datei `/var/named/named.dmz.reverse` beschrieben. Die ersten acht Zeilen einschließlich SOA-, NS- und MX-Einträgen sind identisch mit der gerade eben besprochenen Datei. Statt PTR stehen diesmal aber A-Einträge.

```
1. 11.168.192.in-addr.arpa. IN SOA my.dmz.lan.
   postmaster.my.dmz.lan. (
2.                               1999072701 ; Seriennummer
3.                               28800      ; Refresh: acht Stunden
4.                               14400      ; Retry: vier Stunden
5.                               3600000    ; Expire: ca. 41 Tage
6.                               86400     ) ; Minimum

7.      IN      NS      choke.my.dmz.lan.
8.      IN      MX      10 bastion.my.dmz.lan.

9. bastion.my.dmz.lan.      IN      A      192.168.11.1
10. choke.my.dmz.lan.      IN      A      192.168.11.2
```

Zeile 9 ist ein Adresseintrag (A) und ordnet dem Host `bastion.my.dmz.lan` die IP-Adresse 192.168.11.1 zu.

Zeile 10 ist ebenfalls ein Adresseintrag und ordnet dem Host `choke.my.dmz.lan` die IP 192.168.11.2 zu.

Die Zuordnung von Adressen zu Namen für das interne, private LAN ist in `/var/named/named.local.lan` gespeichert:

```
1. 1.168.192.in-addr.arpa. IN SOA my.local.lan.
   postmaster.my.local.lan. (
2.                               1999072701 ; Seriennummer
3.                               28800      ; Refresh: acht Stunden
4.                               14400      ; Retry: vier Stunden
5.                               3600000    ; Expire: ca. 41 Tage
6.                               86400     ) ; Minimum

7.      IN      NS      choke.my.local.lan.
8.      IN      MX      10 bastion.my.dmz.lan.

9. 1      IN      PTR      choke.my.local.lan.
10. 2     IN      PTR      macintosh.my.local.lan.
11. 3     IN      PTR      bsd.my.local.lan.
```

Die Zeilen eins bis acht sind wieder praktisch identisch mit dem oben gezeigten Beispiel `/var/named/named.dmz`. Sie definieren die Kontrollinformationen für die Zone sowie einen DNS- und einen SMTP-Server.

- Zeile 9 ist ein Pointer bzw. Zeiger (PTR) und definiert für die IP-Adresse 1 (kurz für 192.168.1.1) den zugehörigen Hostnamen `choke.my.local.lan.`.
- Zeile 10 ist ebenfalls ein PTR und ordnet der IP-Adresse 2 bzw. 192.168.1.2 den Hostnamen `macintosh.my.local.lan.` zu.
- Der PTR aus Zeile 11 ordnet der IP-Adresse 3 bzw. 192.168.1.3 den Hostnamen `bsd.my.local.lan.` zu.

Die Zuordnung von Namen zu Adressen für das interne, private LAN steht in `/var/named/named.local.lan.reverse`. Die ersten acht Zeilen mit SOA-, NS- und MX-Einträgen kennen Sie bereits in- und auswendig. Statt PTR- stehen hier wieder A-Einträge:

```

1. 1.168.192.in-addr.arpa. IN SOA my.local.lan.
   postmaster.my.local.lan. (
2.                               1999072701 ; Seriennummer
3.                               28800      ; Refresh: acht Stunden
4.                               14400      ; Retry: vier Stunden
5.                               3600000    ; Expire: ca. 41 Tage
6.                               86400     ) ; Minimum

7.      IN      NS      choke.my.local.lan.
8.      IN      MX      10 bastion.my.dmz.lan.

9. choke.my.local.lan.      IN      A      192.168.1.1
10. macintosh.my.local.lan. IN      A      192.168.1.2
11. bsd.my.local.lan.      IN      A      192.168.1.3

```

- Zeile 9 ist ein A- bzw. Adresseintrag, der dem Host `choke.my.local.lan.` die IP-Adresse 192.168.1.1 zuordnet.
- Zeile 10 ist ein A- bzw. Adresseintrag, der dem Host `macintosh.my.local.lan.` die IP-Adresse 192.168.1.2 zuordnet.
- Zeile 11 ist ein A- bzw. Adresseintrag, der dem Host `bsd.my.local.lan.` die IP-Adresse 192.168.1.3 zuordnet.

`/etc/named.conf`, die Konfigurationsdatei für den lokalen named-Server, sieht so aus:

```

1. options {
2.     directory "/var/named";
3.     forward only;
4.     forwarders {
5.         192.168.11.1;

```

```
6.         };

7.         query-source address * port 53;

8.         allow-query {
9.             127/8;
10.            192.168.11.1;
11.            ! 192.168.11/24;
12.            192.168.1/24;
13.        };

14.        allow-transfer { ! *; };
15.        allow-update { ! *; };

16.        listen-on port 53 { *; };
17.    };

18. zone "0.0.127.in-addr.arpa" {
19.     type master;
20.     notify no;
21.     file "named.127.0.0";
22. };

23. zone "my.dmz.lan" {
24.     type master;
25.     notify no;
26.     file "named.dmz.reverse";
27. };

28. zone "11.168.192.in-addr.arpa" {
29.     type master;
30.     notify no;
31.     file "named.dmz";
32. };

33. zone "my.local.lan" {
34.     type master;
35.     notify no;
36.     file "named.local.lan.reverse";
37. };

38. zone "1.168.192.in-addr.arpa" {
39.     type master;
40.     notify no;
41.     file "named.local.lan";
42. };
```

```
43. zone "." {  
44.     type hint;  
45.     file "root.cache";  
46. };
```

Die Datei `/etc/named.conf` enthält hier zwei Eintragsarten, den `options`-Eintrag sowie die vier `zone`-Einträge. Ersterer enthält Einstellungen, die für den gesamten Server gelten, während die Direktiven innerhalb der `zone`-Einträge nur für diese eine Zone Gültigkeit besitzen.

- Zeile 1 beginnt mit einer öffnenden Klammer einen mehrzeiligen `options`-Eintrag.
- Zeile 2 definiert das Arbeitsverzeichnis des Servers, in dem die Masterdateien für die einzelnen Zonen liegen: `/var/named`.
- Zeile 3 schaltet den Server in den `forward only`-Modus, d.h. Anfragen, die er nicht aus dem eigenen Cache beantworten kann, wird er ausschließlich an die unter `forwarders` definierten Hosts weiterleiten. Die Choke soll alle Anfragen an den Nameserver auf der Bastion weiterleiten.
- Die öffnende Klammer aus Zeile 4 beginnt eine mehrzeilige `forwarders`-Direktive, in der die Nameserver aufgelistet sind, an die wir Anfragen weiterleiten.
- Zeile 5 definiert die Bastion als einzigen solchen Server.
- Zeile 6 beendet die `forwarders`-Direktive mit einer schließenden Klammer.
- Wenn eine Firewall zwischen dem Nameserver und dem Internet steht, wird Zeile 7 benötigt. Sie legt den Absender-Port für Anfragen an andere Nameserver auf 53 fest. D.h. unser Server benutzt jetzt den UDP-Port 53 sowohl als Absender- als auch als Empfänger-Port für Anfragen an andere DNS-Server.
- Zeile 8 beginnt eine mehrzeilige `allow-query`-Direktive, mit der IP-Adressen festgelegt werden, von denen aus der Server Anfragen entgegennehmen oder abweisen soll. Die Reihenfolge ist dabei wichtig: Die Adressliste wird in der angegebenen Reihenfolge durchsucht; die erste passende Regel gilt.
- Zeile 9 instruiert den Server, dass Anfragen von `localhost` erlaubt sind.
- Zeile 10 erlaubt Anfragen von Clients auf dem öffentlichen Nameserver, 192.168.11.1. Wenn der private Server die nachgefragten Daten nicht in seinem Cache hat, wird er die Anfrage an den öffentlichen Server weiterleiten, der seinerseits evtl. noch einmal eine Weiterleitung an einen fremden Server durchführt.
- Zeile 11 verbietet Anfragen von anderen Hosts aus der DMZ. Für dieses Beispiel gehe ich davon aus, dass Hosts aus der DMZ den öffentlichen, externen Nameserver benutzen.

- Zeile 12 erlaubt Anfragen von allen Computern aus dem internen, privaten LAN.
- Zeile 13 beendet die `allow-query`-Direktive mit einer schließenden Klammer.
- Zeile 14 enthält eine einzeilige `allow-transfer`-Direktive, die mit Hilfe der Negation `!` alle Zone-Transfers verbietet.
- Zeile 15 enthält eine einzeilige `allow-update`-Direktive, die mit Hilfe der Negation `!` alle Zonen-Aktualisierungen verbietet. Das ist zwar ohnehin die Voreinstellung, aber die explizite Angabe der Direktive in der Konfigurationsdatei dokumentiert das erwartete Verhalten und liefert zusätzliche Sicherheit, falls irgendwo ein Fehler bestehen sollte.
- Zeile 16 enthält eine einzeilige `listen-on`-Direktive. Diese definiert den Port sowie eine Liste von Netzwerkinterfaces, auf denen der Server auf Anfragen von Clients wartet. Der interne Nameserver soll Anfragen von allen Interfaces entgegennehmen.
- Zeile 17 beendet mit einer rechten Klammer den `options`-Eintrag.
- Zeile 18 beginnt einen mehrzeiligen `zone`-Eintrag für das `loopback`-Netz. Zonnennamen werden im `Arpanet`-Stil angegeben, sodass die `loopback`-Adresse `127.0.0` hier als `0.0.127.in-addr.arpa` auftaucht.
- Zeile 19 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 20 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 21 enthält den Namen der Datei mit den Zonendaten. Weil `named.127.0.0` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 22 schließt den `zone`-Eintrag mit einer rechten Klammer.
- Zeile 23 beginnt einen mehrzeiligen `zone`-Eintrag für die interne DMZ-Domain `my.dnz.lan`. Die hier beschriebenen Zonendaten gelten für Anfragen, bei denen einem symbolischen Namen eine IP-Adresse zugeordnet werden soll.
- Zeile 24 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 25 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 26 enthält den Namen der Datei mit den Zonendaten. Weil `named.dnz.reverse` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 27 schließt den `zone`-Eintrag mit einer rechten Klammer.

- Zeile 28 beginnt einen mehrzeiligen `zone`-Eintrag für die interne DMZ-Domain `11.168.192.in-addr.arpa`. Die hier beschriebenen Zonendaten gelten für Anfragen, bei denen einer IP-Adresse ein symbolischer Name zugeordnet werden soll.
- Zeile 29 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 30 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 31 enthält den Namen der Datei mit den Zonendaten. Weil `named.dnz` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 32 schließt den `zone`-Eintrag mit einer rechten Klammer.
- Zeile 33 beginnt einen mehrzeiligen `zone`-Eintrag für die interne lokale Domain `my.local.lan`. Die hier beschriebenen Zonendaten gelten für Anfragen, bei denen einem symbolischen Namen eine IP-Adresse zugeordnet werden soll.
- Zeile 34 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 35 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 36 enthält den Namen der Datei mit den Zonendaten. Weil `named.local.lan.reverse` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 37 schließt den `zone`-Eintrag mit einer rechten Klammer.
- Zeile 38 beginnt einen mehrzeiligen `zone`-Eintrag für die interne lokale Domain `1.168.192.in-addr.arpa`. Die hier beschriebenen Zonendaten gelten für Anfragen, bei denen einer IP-Adresse ein symbolischer Name zugeordnet werden soll.
- Zeile 39 deklariert die enthaltenen Zonendaten als Master-Kopie, d.h. der Server ist für diese Zone autoritativ.
- Zeile 40 weist darauf hin, dass bei einer Änderung der Zonendaten keine anderen Server benachrichtigt werden müssen.
- Zeile 41 enthält den Namen der Datei mit den Zonendaten. Weil `named.local.lan` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 42 schließt den `zone`-Eintrag mit einer rechten Klammer.

- Zeile 43 leitet einen mehrzeiligen `zone`-Eintrag für den Cache der Root-Domain ein. Dieser enthält die Root-Nameserver des Internets. Ein `forward only`-Nameserver benötigt die Root-Zone genau genommen nicht, weil alle Anfragen ausschließlich an die Hosts aus der `forwarders`-Direktive weitergeleitet werden. Ein `forward first`-Nameserver hingegen kommt ebenso wenig wie ein regulärer Nameserver ohne diese Daten aus; beide beginnen über die hier angegebenen Rechner mit der Suche nach autoritativen Nameservern.
- Zeile 44 deklariert die Zone-Daten dieses Eintrags als *Hint* oder Tipp, d.h. es handelt sich nur um einen Ausgangspunkt für die weitere Suche.
- Zeile 45 enthält den Namen der Datei mit den Zonendaten. Weil `root.cache` ein relativer Pfad ist, wird die Datei in `/var/named` (der `directory`-Einstellung aus dem `options`-Eintrag) gesucht.
- Zeile 46 schließt den `zone`-Eintrag mit einer rechten Klammer ab.

Zugriffskontrolle für die Zonendaten

Wenn man einmal von der Möglichkeit des Cache-Poisoning mit gefälschten IP-Adressen absieht – dabei schmuggelt jemand fehlerhafte Informationen in Ihren DNS-Cache –, sind ankommende Client-Anfragen von fremden Rechnern kein ernsthaftes Sicherheitsrisiko. Anders ist das, wenn Ihre lokalen DNS-Datenbanken private Informationen enthalten – beispielsweise nutzen einige große Firmen den DNS für die Ablage von Name, Telefonnummer und Adresse aller Mitarbeiter. Wie oben gezeigt, lassen sich die erlaubten DNS-Clients mit der Option `allow-query` eingrenzen.

Ein größeres Sicherheitsrisiko besteht im Auslesen der gesamten Zonendatenbank über eine TCP-Verbindung, wodurch jemand Ihre gesamte LAN-Topologie einsehen kann. Wie bereits erwähnt, ist das in der Voreinstellung erlaubt! Sie sollten den Zugriff auf die komplette Zonendatenbank nur Ihren sekundären Nameservern erlauben, sofern solche existieren. Verwenden Sie dazu die Direktive `allow-transfer`.

Das größte Risiko liegt in Schreibzugriffen auf die Zonendaten, was normalerweise verboten ist. Der einzige gültige Schreibzugriff erfolgt, wenn ein primärer Nameserver die Daten der sekundären Server aktualisiert. Das erlaubt man mit der `allow-update`-Direktive. Wenn ein Angreifer DNS-Daten überschreiben darf, kann er alle IP-Adressen Ihrer Domain fälschen und ein völlig anderes Bild der Topologie implementieren.

7.3.5 ftp-Konfiguration

ftp war immer wieder Ziel erfolgreicher Angriffe. Obwohl die bekannten Probleme im Moment gelöst sind, tauchen immer wieder neue Schwierigkeiten auf. Unabhängig von potenziellen Sicherheitslücken in den Serverprogrammen selbst liegt das größte Risiko in der lokalen Konfiguration. Die zwei Hauptprobleme

sind dabei die Einstellungen in den ftp-Konfigurationsdateien unter `/etc` sowie die Zugriffsrechte und Inhalte des öffentlichen, anonym zugänglichen Bereichs unter `/home/ftp`.

Bei installiertem ftp-Server dürfen normalerweise nur registrierte Benutzer nach Eingabe von Username und Passwort über ftp auf das System zugreifen. Wenn Sie zusätzlich das Paket für anonymes ftp eingespielt haben, sind darüber hinaus auch anonyme Benutzer erlaubt.

Der ftpd ist durch die `tcp_wrapper` geschützt. Bei Red Hat wird der ftpd-Server aus `inetd` heraus mit den beiden Optionen `-l` und `-a` gestartet. `-l` schaltet die Protokollierung aller Zugriffe durch `syslogd` ein. `-a` aktiviert die Zugriffskontrolldatei `/etc/ftppass`. Falls Sie einen öffentlichen, anonymen ftp-Dienst betreiben, empfehle ich zusätzlich die beiden Optionen `-i` und `-o`. Die Option `-i` zeichnet alle ankommenden Dateiübertragungen in `/var/log/xferlog` auf, die Option `-o` alle abgehenden.

Hinweis

Weitere Informationen zur ftp-Konfiguration

Nähere Informationen zur Konfiguration von ftp erhalten Sie in den Manual-Pages zu `ftpd(8)`, `ftppass(5)`, `ftpconversions(5)` und `xfer-log(5)`. Hinweise zu anonymem ftp enthalten die Dokumente »Anonymous FTP Configuration Guidelines« ([ftp://ftp.cert.org/pub/tech_tips/anonymous_ftp_config](http://ftp.cert.org/pub/tech_tips/anonymous_ftp_config)), »UNIX Configuration Guidelines« (http://www.cert.org/ftp/tech_tips/UNIX_configuration_guidelines) und »UNIX Computer Security Checklist« (http://www.cert.org/ftp/tech_tips/AUSCERT_checklist1.1). Zusätzliche Informationen zu Sicherheitsproblemen von ftp finden Sie in den Texten »Anonymous FTP Abuses« (http://www.cert.org/ftp/tech_tips/anonymous_ftp_abuses) und »Problems With The FTP PORT Command« (http://www.cert.org/ftp/tech_tips/FTP_PORT_attacks).

ftp mit Login: Konfigurationsdateien in /etc

Zu ftp gehören fünf Konfigurationsdateien in `/etc`:

- `/etc/ftppass` ist die Hauptkonfigurationsdatei für den ftpd-Server.
- `/etc/ftpconversions` beschreibt den Umgang mit durch `compress`, `gzip` und/oder `tar` komprimierten Dateien und Verzeichnissen.
- `/etc/ftpgroups` enthält eine Liste von Benutzergruppen sowie einen Bereich des Dateisystems, zu dem der Server jeweils ein `chroot` durchführt. Beispielsweise könnte man für verschiedene Entwickler-Gruppen unterschiedliche Gruppen definieren.
- `/etc/ftphosts` enthält eine Aufstellung, welche Benutzer von welchen Computern aus den ftp-Service benutzen dürfen.

- In `/etc/ftpusers` sind alle Accounts aufgeführt, die ftp *nicht* benutzen dürfen. Hierzu gehören mindestens `root` sowie die anderen Systemaccounts.

Sie werden vermutlich die Dateien `ftphosts` und `ftppass` anpassen. `ftpconversions`, `ftpgroups` und `ftpusers` hingegen müssen Sie möglicherweise nie verändern.

Ein Beispiel für `/etc/ftphosts`:

```
allow bob 192.168.1.*
allow jake 10.10.47.112
```

bob darf ftp von allen Rechnern im privaten LAN aus benutzen; jake nur von 10.10.47.112 aus.

Weil `ftpd` durch die `tcp_wrapper` geschützt ist, würde der folgende Eintrag in `/etc/hosts.allow` die geschilderten Verbindungen erlauben:

```
in.ftpd: LOCAL 192.168.1. 10.10.47.112
```

Damit authentifiziertes ftp (mit Eingabe von Username und Passwort) gestattet wird, benötigen die Benutzer neben einem Account auf dem System und einem gültigen Passwort auch eine Standardshell in `/etc/passwd`.

`/etc/ftppass` ist die Hauptkonfigurationsdatei für den `ftpd`-Server. Dabei sind viele Direktiven möglich. Hier ein Beispiel:

```
1. class friends real 192.168.1.*
2. class friends real 10.10.47.112
3. # class other anonymous *

4. deny !nameserved .no_name_server

5. greeting brief
6. # banner /home/ftp/banner

7. message /welcome.msg login
8. message .message cwd=*

9. email root@localhost

10. loginfails 5
11. limit-time anonymous 30
12. anonymous-root /home/ftp
13. defaultserver private

14. compress yes friends
15. tar yes friends
16. chmod no friends
17. delete yes friends
```

```
18. overwrite      yes      friends
19. rename         yes      friends

20. noretrieve     passwd   shadow group gshadow core

21. upload         /home/ftp *          no
22. upload         /home/ftp /incoming yes bob bob 0622 dirs
23. upload         /home/ftp /incoming yes jake jake 0622 nodirs

24. log transfers  anonymous      inbound,outbound
25. log security   anonymous

26. passwd-check   rfc822 enforce
```

/etc/ftppaccess enthält die Definitionen der Benutzergruppen mit den zugeordneten Zugriffsrechten, Hinweise auf Willkommensmeldungen und Verzeichnisinformationen sowie Protokolldateien.

- Zeile 1 definiert `friends` als eine Klasse authentifizierter Benutzer, die von jedem Host aus dem Netzwerk 192.168.1 zugreifen dürfen.
- Zeile 2 erlaubt `friends` zusätzlich den Zugriff von 10.10.47.112.
- Zeile 3 definiert `other` als eine Klasse anonymer Benutzer, die von überall zugreifen dürfen. Diese Zeile ist auskommentiert.
- Zeile 4 verweigert allen Sites den Zugriff, deren Adresse sich nicht auflösen lässt. Dem fremden User wird die in `/home/ftp/.no_name_server` enthaltene Meldung angezeigt.
- Zeile 5 legt fest, welche Informationen angezeigt werden, bevor sich der fremde Benutzer einloggt. `brief` gibt nur den Hostnamen aus. Die Voreinstellung würde zusätzlich noch die `ftpd`-Version anzeigen.
- Zeile 6 verweist auf eine Meldung, die noch vor dem Login gesendet wird. Die Zeile ist auskommentiert.
- Zeile 7 verweist auf eine Meldung, die nach erfolgreichem Login gesendet wird.
- Zeile 8 verweist auf eine Meldung, die bei jedem Verzeichniswechsel gesendet wird.
- Zeile 9 enthält die E-Mail-Adresse des `ftp`-Administrators.
- Zeile 10 gibt an, wie oft ein Loginversuch fehlschlagen darf, bevor der Server die Verbindung trennt.
- Zeile 11 setzt eine Obergrenze für die Dauer einer anonymen Verbindung. In diesem Fall wird eine Sitzung nach 30 Minuten unterbrochen, wenn der fremde Benutzer eine Mittagspause einlegt.

- Zeile 12 definiert die Wurzel des Verzeichnisbaums für anonymes ftp.
- Zeile 13 ist eine andere Möglichkeit, anonymes ftp zu verbieten.
- Die Zeilen 14-19 legen fest, für welche Funktionen die Mitglieder einer Benutzerklasse autorisiert sind. Im Beispiel dürfen authentifizierte Benutzer die Zugriffsrechte von Dateien nicht verändern.
- Zeile 20 gibt eine Liste von Dateien an, die niemand übertragen darf.
- Zeile 21 verbietet ankommende Dateiübertragungen für den anonymen ftp-Bereich.
- Zeile 22 erlaubt bob den Upload von Dateien in das incoming-Verzeichnis. Er darf dort auch neue Unterverzeichnisse anlegen.
- Zeile 23 erlaubt jake ebenfalls den Upload von Dateien ins incoming-Verzeichnis, verbietet aber die Erstellung von Unterverzeichnissen.
- Zeile 24 zeichnet alle ankommenden und abgehenden anonymen Dateiübertragungen auf.
- Zeile 25 protokolliert alle Versuche anonymen Benutzer, auf abgesicherte Dateien zuzugreifen oder verbotene Aktivitäten durchzuführen.
- Zeile 26 schaltet eine strenge Überprüfung anonymer Passwörter ein und sorgt für einen Verbindungsabbruch, wenn ein Benutzer kein passendes Passwort eingibt.

Anonymes ftp: /home/ftp

Wenn Sie das Paket für anonymes ftp installiert haben, dürfen ab dem Systemstart auch anonyme Benutzer auf ftp zugreifen. Dafür wird ein ftp-Account in `/etc/passwd` benötigt. Dessen Passwort sollte deaktiviert bleiben, d.h. im entsprechenden `passwd`-Feld muss ein `*` stehen. Das Heimatverzeichnis von ftp sollte `/home/ftp`, die Login-Shell `/bin/false` sein.

Anonyme Benutzer greifen als dieser Account (ftp) auf das System zu. Der ftpd-Server führt automatisch ein `chroot` in den anonymen Dateibereich (`/home/ftp`) durch, sobald `inetd` den Server aufgerufen und der fremde Benutzer eine anonyme Verbindung angefordert hat.

Neben den allgemeinen Servereinstellungen sollten Sie auch das `/home/ftp`-Verzeichnis sorgfältig überprüfen, und zwar in Bezug auf Inhalt, Eigentümer und Zugriffsrechte. Fehlkonfigurationen der Zugriffsrechte dieses Dateibereichs sind eine der Hauptursachen für Einbrüche in ftp-Sites.

`/home/ftp` einschließlich aller Unterverzeichnisse sollte nicht dem anonymen Benutzer ftp gehören, ebenso wenig der ftp-Gruppe. Stattdessen sollte der Eigentümer aller Dateien und Verzeichnisse `root` oder ein anderer Account sein. Nur der Eigentümer – `root` – darf in `/home/ftp` und die Unterverzeichnisse schreiben.

Normalerweise enthält der anonyme ftp-Bereich vier Unterverzeichnisse: `bin`, `etc`, `lib` und `pub`. Weil `ftpd` für anonyme Zugriffe ein `chroot` durchführt, müssen Sie einige der normalerweise in diesen Verzeichnissen befindlichen Dateien hierher kopieren. Jedes der angeführten Verzeichnisse erfordert leicht abweichende Zugriffsrechte:

- `/home/ftp/bin` enthält Kopien aller Programme, mit deren Hilfe ftp die Funktionalität von `ls`, `cd` und der Komprimierungsprogramme anbietet. Für `bin` einschließlich seines Inhalts sollten Sie das »Ausführen«-Bit für jedermann einschalten, die »Lesen«- und »Schreiben«-Bits für jedermann ausschalten. Mit anderen Worten: `bin` und alle enthaltenen Dateien sollten `chmod 0111` sein.
- `/home/ftp/etc` enthält eine Kopie des Caches mit den Daten über dynamisch ladbare Bibliotheken, `ld.so.cache`. Diese Datei soll jeder lesen, aber nur `root` beschreiben können.

`etc` enthält auch Pseudo-Kopien von `/etc/passwd` und `/etc/group` mit minimalen Informationen über Dateieigentümer innerhalb des anonymen ftp-Bereichs. Normalerweise betrifft das nur die Accounts `root` und `ftp`. Die Datei `/home/ftp/etc/passwd` darf unter keinen Umständen Passwörter enthalten! Die Passwortfelder werden durch Sternchen (*) ersetzt. `group` und `passwd` sind für jedermann lesbar, aber für niemand beschreibbar. Einzige Aufgabe dieser beiden Dateien ist es, für das `ls`-Kommando Informationen über Datei- und Verzeichniseigentümer zu liefern.

Für Verzeichnis `etc` sollten Sie das »Ausführen«-Bit einschalten, die »Lesen«- und »Schreiben«-Bits ausschalten, d.h. das Verzeichnis selbst sollte `chmod 0111` sein.

Warnung

Inhalt der Passwort-Datei für ftp

Es ist ganz besonders wichtig, dass `/home/ftp/etc/passwd` keine Passwörter enthält! Eine häufige Fehlkonfiguration besteht in der Verwendung von `/etc/passwd` mit den verschlüsselten Passwörtern. Damit können anonyme ftp-User sich ganz einfach die `passwd`-Datei herunterladen, die enthaltenen Passwörter knacken und sich völlig regulär einloggen.

- `/home/ftp/lib` enthält Kopien der wenigen dynamisch geladenen Bibliotheken, die ftp benutzt. `lib` samt Inhalt sollte für jedermann lesbar und ausführbar, für niemand beschreibbar sein: `lib` und alle darin gespeicherten Dateien sollten `chmod 0555` sein.
- `/home/ftp/pub` ist der öffentlich zugängliche ftp-Verzeichnisbaum. Anonyme Benutzer greifen auf die gespeicherten Dateien und Verzeichnisse zu. Dazu sollten die »Lesen«- und gegebenenfalls »Ausführen«-Bits eingeschaltet, die

»Schreiben«-Bits hingegen ausgeschaltet sein. D.h. `pub` und alle Unterverzeichnisse sollten `chmod 0555`, alle enthaltenen Dateien `chmod 0444` sein.

Warnung

Eigentümer des `pub`-Verzeichnisses

Seit Red Hat 6.0 gehört das `pub`-Verzeichnis dem Benutzer `root` und der Gruppe `ftp`. Darüber hinaus sind `pub` und alle Unterverzeichnisse `setgid ftp`. Zunächst stellt dieses Szenario noch kein Sicherheitsproblem dar, weil keine Schreibzugriffe erlaubt sind. Wäre das aber der Fall, und wäre anonymes `ftp` erlaubt, ergäbe sich die Möglichkeit für einen der häufigsten `ftp`-Angriffe: Ein fremder Benutzer könnte eine Datei auf Ihr System schreiben oder auch bestehende Dateien verändern. Damit könnte er über Ihren Computer illegale Raubkopien verbreiten, Ihr System durch riesige Mengen an Müll in die Knie zwingen oder, je nachdem, welche Services aktiv sind, sogar über den `ftp`-Account einen Shell-Zugang erreichen. Beachten Sie, dass er auch die Zugriffsrechte von Dateien und Verzeichnissen ändern könnte, wenn für anonyme Benutzer in `/etc/ftpaccess` die `chmod`-Funktion eingeschaltet wäre.

7.3.6 Konfiguration eines POP-Servers

Die Mailauslieferung über POP konfiguriert man auf einer kleinen Site in der Regel als privaten, lokalen Dienst, wenn man nicht ganz darauf verzichtet. Lokale User können ihre E-Mail über POP von einem zentralen Mailserver auf ihre eigenen Workstations übertragen. Wenn Sie auch Computern aus dem Internet POP anbieten müssen, achten Sie auf eine möglichst sichere Konfiguration, z.B. indem Sie Verbindungen nur über `ssh` gestatten.

Der `popd`-Server wird in `/etc/inetd.conf` aktiviert. Wenn `/etc/hosts.allow` allen lokalen Maschinen den Zugriff auf alle lokalen Dienste gestattet, die durch die `tcp_wrapper` geschützt sind, wird kein eigener `ipop3d`-Eintrag benötigt. Externe Verbindungen verbietet `/etc/hosts.deny`. Die Paketfilter-Firewall bietet eine zusätzliche Sicherheitsschicht, indem sie ankommende Verbindungen zum POP-Server ablehnt – es gibt keine Firewallregeln für Fernzugriffe auf den Server, also werden die Pakete durch die Policy blockiert.

Eine explizite Konfiguration ist für den POP-Server selbst nicht erforderlich. Nur wenn Sie den Zugang von anderen Computern aus dem Internet gestatten wollen, sind zwei weitere Maßnahmen erforderlich: Tragen Sie zum einen alle erlaubten Clients als IP-Adresse oder Hostname in `/etc/hosts.allow` ein. Der POP-Server heißt dort `ipop3d`. Zum anderen benötigen Sie explizite Firewall-Regeln, die einen Datentransfer zwischen Clients und POP-Server zulassen.

7.3.7 Konfiguration eines DHCP-Servers

Ein lokaler DHCP-Server (`dhcpcd`) auf der Firewall ist eine gefährliche Sache. Wenn Ihre Nachbarn im Subnetz des Internet-Providers DHCP-Clients sind, könnten sie Ihren Server mit dem des Providers verwechseln. Ein falsch konfigurierter DHCP-Server wird Ihnen garantiert eine Sperrung des Internet-Zugangs bescheren, zumindest bis Sie das Problem gelöst haben.

In manchen Situationen bietet ein lokaler Server für ein internes LAN trotzdem unübersehbare Vorteile. Ein Beispiel wäre ein ganz kleines Privatnetz, vielleicht nur mit einem einzigen Linux-Computer, plus einem Notebook, das zwischen dem Privat- und einem Firmennetz hin- und herwandert. Ein anderes Beispiel wäre eine kleine Firma, die schon mehr Computer besitzt, als man bequem per Hand konfigurieren kann.

Auf einer kleinen Site mit internem LAN müssen Sie den `dhcpcd`-Server auf der Firewall sorgfältig konfigurieren, damit er zwar die Rechner im LAN bedient, nicht aber Clients aus dem Internet. Die Firewall-Regeln müssen Nachrichten vom Server strengstens vom Internet abschirmen. Eine Alternative bestünde darin, den DHCP-Server für die LAN-Maschinen auf einem zweiten Gerät innerhalb des LANs zu betreiben. Die Firewall würde dann garantieren, dass DHCP-Nachrichten aus dem LAN nicht über das externe Netzwerkinterface ins Internet gelangen.

Größere Sites mit einem DMZ-Netzwerk und einer zweiten, internen Choke-Firewall können den `dhcpcd`-Server auf der internen Firewall betreiben. Sowohl externe als auch interne Firewall würden in diesem Szenario DHCP-Pakete auf das LAN beschränken.

Man kann den DHCP-Server relativ einfach dazu zwingen, Pakete nur über ein bestimmtes Netzwerkinterface zu versenden. Dieses muss aber definiert sein, bevor der Server erstmals aufgerufen wird. Wenn Sie Ihren `dhcpcd` auf ein einzelnes Interface beschränken wollen, editieren Sie sein Startup-Skript in `/etc/rc.d/init.d/dhcpcd`. Dieses ruft den Server normalerweise so auf:

```
daemon /usr/sbin/dhcpcd
```

Wenn Ihr externes Interface `eth0` heißt und Ihr internes Interface `eth1`, ändern Sie diese Zeile in:

```
daemon /usr/sbin/dhcpcd eth1
```

Wenn der Dämon auf der Firewall und nicht auf einer internen Maschine läuft, benötigen Sie gegebenenfalls einen zusätzlichen Eintrag in der Routing-Tabelle für das interne Interface. Dafür ergänzen Sie in `/etc/rc.d/rc.local` die folgende Zeile:

```
route add -host 255.255.255.0 dev eth1
```

Die Konfigurationsdatei für `dhcpd` heißt `/etc/dhcpd.conf`. Diese Datei kann globale Parameter enthalten; zusätzlich muss sie jedes Subnetz beschreiben, das an den Server angeschlossen ist. Ein einfaches Beispiel für ein privates LAN mit dynamischer Adressvergabe sähe so aus:

```
1. option domain-name "local.lan";
2. option domain-name-servers 192.168.1.1;
3. option subnet-mask 255.255.255.0;

4. subnet 192.168.1.0 netmask 255.255.255.0 {
5.     range 192.168.1.2 192.168.1.254;
6.     default-lease-time 86400;
7.     max-lease-time 2592000;
8.     option broadcast-address 192.168.1.255;
9.     option routers 192.168.1.1;
10. }
```

- Zeile 1 enthält einen globalen Parameter mit dem Domainnamen `domain-name`.
- Zeile 2 enthält einen globalen Parameter mit den Nameservern, `domain-name-servers`.
- Zeile 3 enthält einen globalen Parameter mit der Netzmaske, `subnet-mask`.
- Zeile 4 beginnt einen `subnet`-Eintrag für ein Netzwerkinterface. Die Netzwerkadresse ist hier `192.168.1.0`, die Netmask `255.255.255.0`, d.h. jede IP-Adresse, die in den ersten 24 Bits mit der Netzadresse übereinstimmt, gehört zum lokalen Adressbereich.
- Zeile 5 (`range`) definiert, welche Adressen dynamisch vergeben werden dürfen. Angegeben sind die erste und letzte Adresse.
- Zeile 6 legt die voreingestellte Gültigkeitsdauer einer dynamisch vergebenen IP-Adresse (`default-lease-time`) in Sekunden fest. Der Wert beträgt hier 86.400 Sekunden oder genau einen Tag.
- Zeile 7 setzt eine Obergrenze (`max-lease-time`) für die maximale Gültigkeitsdauer einer IP-Adresse, hier 2.592.000 Sekunden bzw. einen Monat.
- Zeile 8 enthält die Broadcast-Adresse des LANs (`broadcast-address`).
- Zeile 9 informiert über den Gateway-Router des LANs (`router`).
- Zeile 10 beendet den `subnet`-Eintrag.

Weitere Informationen zur Konfiguration von `dhcpd` finden Sie in den Manual-Pages zu `dhcpd(8)` und `dhcpd.conf(5)`.

7.3.8 NTP-Konfiguration

Das *Network Time Protocol* (NTP) dient der Synchronisation der Systemzeit Ihres Computers mit der eines autoritativen Zeitserver.

Typischerweise läuft auf einem einzelnen Computer einer Site der NTP-Client `ntpdate`. Er fragt die aktuelle Zeit von mehreren Servern ab; am besten funktioniert NTP mit mindestens drei fremden Servern. `ntpdate` schätzt die wahre Zeit aufgrund der von den Servern angegebenen Autorität und Genauigkeit sowie der Varianz der von den einzelnen Servern erhaltenen Zeitangaben ab. Die erhaltene Zeit schreibt es in die Systemuhr des Computers.

Damit nicht unnötig viele Datenpakete ausgetauscht werden müssen, betreibt man auf dem gleichen Computer i.d.R. einen lokalen NTP-Server, `xntpd`, der die aktuelle Systemzeit an die anderen Computer im LAN weitergibt. Auf einer sehr großen Site würde man gegebenenfalls den ersten `xntpd`-Server als autoritativen Master konfigurieren und dann weitere interne Maschinen mit zusätzlichen `xntpd`s einrichten. Diese sekundären Server übernehmen die Weitergabe der Zeit an die Endgeräte und reduzieren damit die Belastung des Master-Servers.

Beispiel für ein kleines LAN: Ein einzelner Host setzt seine Systemzeit beim Bootvorgang mit dem Client-Programm `ntpdate`. Anschließend startet er `xntpd` als Server und bietet damit den anderen Computern auf dem LAN einen Zeitdienst an. Diese anderen Rechner verwenden `xntpd` als Client, um regelmäßig den lokalen Server abzufragen.

Der Server-Computer führt beim Booten das Startup-Skript `/etc/rc.d/init.d/xntpd` aus. Wenn die Konfigurationsdatei `/etc/ntp/step-tickers` existiert, liest `ntpdate` die Namen der fremden Zeitserver aus dieser Datei. Anschließend ruft das Skript den Dämon `xntpd` auf.

Die Konfigurationsdatei für `xntpd` heißt `/etc/ntp.conf`. Für den Haupt-Server im LAN sieht diese Datei so aus:

1. `restrict default nomodify`
2. `server 127.0.0.1`
3. `restrict 127.0.0.1`

Zeile 1 definiert, inwieweit wir fremden Hosts vertrauen. Hier ist die Voreinstellung, dass den Zeitangaben der Server Vertrauen entgegengebracht wird, dass sie aber keine Änderungen an der Konfiguration unseres Servers vornehmen dürfen. Unser `xntpd` nimmt von sich aus keinen Kontakt zu anderen Servern auf.

Zeile 2 deklariert den `localhost` als Zeit-Server.

Zeile 3 entfernt die voreingestellten Einschränkungen vom lokalen Server, sodass die Konfiguration zur Laufzeit geändert werden kann.

Die Client-Geräte rufen ebenfalls beim Booten das Skript `/etc/rc.d/init.d/xntpd` auf. Wenn die Konfigurationsdatei `/etc/ntp/step-tickers` existiert, liest `ntpdate` die Namen der fremden Zeitserver aus dieser Datei. Anschließend ruft das Skript den Dämon `xntpd` auf.

Warnung**Wechselwirkungen zwischen *ntdate* und *xntpd***

ntdate aktualisiert die Systemzeit nicht, solange *xntpd* läuft. In der hier vorgestellten Konfiguration führt *xntpd* von sich aus ebenfalls keine Zeit-Updates durch. Damit die Zeit mit der eines anderen Servers synchronisiert wird, könnte man z.B. von *cron* aus regelmäßig den Befehl `/etc/rc.d/init.d/xntpd restart` ausführen. Er hält den *xntpd*-Dämon an, aktualisiert die Zeit mit *ntdate* und startet *xntpd* wieder.

Die Konfigurationsdatei für *xntpd* heißt `/etc/ntp.conf`. Auf den internen LAN-Maschinen sieht diese Datei so aus:

```
1. restrict 192.168.11.1 nomodify
2. server 192.168.11.1
3. server 127.0.0.1
4. restrict 127.0.0.1
```

Zeile 1 definiert, inwieweit wir fremden Hosts vertrauen. Hier ist die Voreinstellung, dass den Zeitangaben der Server Vertrauen entgegengebracht wird, dass sie aber keine Änderungen an der Konfiguration unseres Servers vornehmen dürfen.

Zeile 2 definiert die interne IP-Adresse der Firewall als fremden Server. Im Beispiel sind keine Zeitintervalle angegeben, wie oft die Zeit synchronisiert werden soll. Damit gelten die Voreinstellungen, und Anfragen werden etwa alle ein bis fünfzehn Minuten abgesetzt.

Zeile 3 deklariert den `localhost` seinerseits als Server.

Zeile 4 entfernt die voreingestellten Einschränkungen vom lokalen Server, sodass die Konfiguration zur Laufzeit geändert werden kann.

Hinweis**Offizielle NTP-Server**

Öffentlich zugängliche primäre NTP-Server erhalten ihre Zeit über Funk- oder Satellitenempfänger bzw. über Modem. Aktuell existieren ca. 50 öffentliche primäre Server sowie etwa 100 öffentliche sekundäre Server. Dazu kommen tausende öffentliche Server auf höheren Ebenen. Die meisten NTP-Server bedienen eine bestimmte geografische Region oder einen Adressblock. Welche öffentlich zugänglichen sekundären Server in Ihrer Nähe sind, erfahren Sie unter <http://www.eecis.udel.edu/~mills/ntp/servers.html>. Weitere Informationen über Protokoll, Software und Dokumentation rund um NTP finden Sie auf der gleichen Site unter <http://www.eecis.udel.edu/~mills/ntp/>.

7.3.9 Konfiguration von CGI-Skripten

CGI-Skripten sind vom Web-Server ausgeführte Programme, die Funktionalitäten jenseits der Fähigkeiten eines Webservers ermöglichen. Diese Skripten benutzen bei der Erfüllung ihrer Aufgabe normalerweise lokale Programme. Sie haben die gleiche Autorisation für den Zugriff auf Systemressourcen wie der Webserver bzw. der Account, unter dem sie `setuid` laufen. Was das Skript genau tut, hängt meistens von willkürlichen Daten ab, die der Benutzer auf der anderen Seite des Internets eingegeben hat.

Wenn nicht ganz besondere Vorkehrungen getroffen werden, sind CGI-Binaries besonders anfällig für Sicherheitsprobleme. An sich sollten CGI-Skripten über keinerlei Privilegien verfügen. Wenn sie aber irgendwelche besonderen Zugriffsrechte benötigen, setzt der Autor oft das `setuid`- oder `setgid`-Bit, wodurch das Skript als privilegierter Benutzer läuft. Der Webserver selbst arbeitet normalerweise unter einer unprivilegierten Benutzererkennung wie `nobody`. Wenn er nicht korrekt eingestellt ist, kann er aber ebenso als `root` ausgeführt werden. In dieser Situation verfügt auch das CGI-Skript über `root`-Rechte. In der Vergangenheit resultierten aus solchen Unstimmigkeiten bezüglich der Autorisierung des CGI-Skriptes ernsthafte Einbruchsmöglichkeiten, bei denen ein fremder Eindringling sich `root`-Rechte verschaffen konnte, indem er ein CGI-Skript mit unerwarteten Daten fütterte.

Neben den in der Dokumentation zum Apache-Webserver empfohlenen Sicherheitsmaßnahmen – die Sie von www.apache.org erhalten – stelle ich hier noch zwei weitere Überlegungen vor. Zum einen kann ein Zwischenprogramm namens `cgiwrap` CGI-Skripten in ähnlicher Weise behandeln, wie die `tcp_wrapper` `inetd`-Dienste schützen. Zum Zweiten sollte jedes CGI-Skript die Benutzereingaben sorgfältig überprüfen, bevor es auf Systemressourcen zugreift.

`cgiwrap` stellt keine Zugriffskontrolle für CGI-Binaries dar. Stattdessen sorgt es dafür, dass die Skripten wirklich mit den beabsichtigten Rechten laufen, nicht mit denen des Servers – selbst wenn der Server versehentlich als `root` arbeitet. `cgiwrap` erhalten Sie als Quellcode einschließlich Dokumentation von [ftp://ftp.cc.umr.edu/pub/cgi/cgiwrap](http://ftp.cc.umr.edu/pub/cgi/cgiwrap).

Die gewissenhafte Kontrolle der Benutzereingaben für die Skripten gestaltet sich schwieriger, weil jedes Programm andere Daten erwartet. Ein von CERT erhältliches Dokument namens »How To Remove Meta-characters From User-Supplied Data In CGI Scripts« (http://www.cert.org/tech_tips/cgi_metacharacters.html) beschäftigt sich mit einem allgemeinen Ansatz für die Fehlerprüfung und enthält Beispiele als Perl- und C-Quelltext. Alle Benutzereingaben müssen überprüft werden; Daten jenseits einer maximal zulässigen Größe müssen verworfen werden. Schon dieser eine Schritt schützt ein Skript vor den verbreiteten Buffer-Overflows. Ihr Programm sollte nicht versuchen, alle undefinierten und unerwarteten Benutzereingaben vorwegzunehmen, sondern umgekehrt ganz genau definieren, wie eine erlaubte Eingabe aussieht und alles andere verwerfen.

7.4 SOCKS: eine Proxy-Firewall auf Anwendungsebene

SOCKS ist ein *Circuit-Gateway-Proxy*.¹

In einer LAN-Umgebung arbeitet SOCKS als Proxy-Firewall auf Anwendungsebene. Clients im LAN dürfen nur auf den Computer mit dem SOCKS-Server zugreifen. Dieser kann als Einziger direkt mit dem Internet kommunizieren.

SOCKS ist ein transparenter Proxy, d.h. für den Benutzer sieht es so aus, als würde das Client-Programm direkt mit dem fremden Computer sprechen. Weil die SOCKS-Clientprogramme so modifiziert sind, dass sie nur mit dem SOCKS-Server Daten austauschen, besteht in Wirklichkeit lediglich eine Verbindung zum lokalen Server. Dieser prüft die Zugriffsberechtigung des Benutzers und baut für ihn eine Verbindung zum fremden Computer auf. Anschließend leitet er alle Daten zwischen lokalem Client und fremdem Server weiter.

Weil der Server als Proxy für die LAN-Clients fungiert, sind die Computer aus dem LAN für das Internet unsichtbar, genau wie bei IP-Masquerading auf Paketebene.

SOCKS unterstützt in der Version 4 nur TCP-Dienste. Die aktuelle Version 5 versteht zusätzlich auch UDP und Multicast, erlaubt starke Benutzer-Authentifizierung und DNS-Lookups und enthält den Client-Code in Form gemeinsam genutzter Bibliotheken, wodurch in manchen Fällen das Client-Programm für SOCKS-Unterstützung nicht mehr neu übersetzt werden muss.

Die neue UDP-Weiterleitung ist für eine mehrschichtige Sicherheitsstrategie ein besonders attraktives Feature. Paketfilter können UDP-Datentransfers nicht adäquat schützen. Die UDP-Weiterleitung von SOCKS assoziiert Hosts und Ports zwischen Client- und Server-Prozessen und garantiert damit, dass der Client nur vom angesprochenen Server Datagramme erhält. Bei Services wie RealAudio, die TCP und UDP benutzen, erwartet SOCKS ankommende UDP-Pakete vom fremden Ende der TCP-Verbindung als Absender.

Da SOCKS das Einhalten der Protokolle auf Anwendungsebene erzwingt, ist es auch für »Rückrufdienste« wie z.B. ftp im aktiven Modus interessant. SOCKS garantiert, dass die ankommende Datenverbindung von TCP-Port 20 von dem gleichen Host stammt, zu dem der Client die Kontrollverbindung auf Port 21 aufgebaut hat.

Zusammenfassend stellt SOCKS eine interessante Alternative zu den IP-Masquerading-Modulen von Linux dar, vorausgesetzt, dass alle Ihre Client-Programme über SOCKS kommunizieren können. Die Masquerading-Module von Linux ver-

1. SOCKS war der interne Entwicklungsname dieses Projektes und steht für »Sockets«. Nach der Veröffentlichung blieb der Name irgendwie hängen, und die Software heißt heute noch SOCKS.

stehen eine umfassende, aber doch begrenzte Reihe beliebter Dienste, während SOCKS jeden beliebigen Netzwerkdienst unterstützt.

Wenn Sie SOCKS als Proxy einsetzen, sperren Sie jeden Zugang zum Internet aus dem LAN und erlauben LAN-Pakete nur zum internen Netzwerkinterface der Firewall. Besonders wichtig ist, dass Sie ankommende Verbindungen aus dem Internet zum SOCKS-Server auf TCP-Port 1080 blockieren! Der Proxy sollte niemals eine Verbindung von einem fremden Rechner annehmen.

Als Proxy führt SOCKS praktisch ein IP-Masquerading für das LAN durch. Wenn SOCKS den einzigen Übergang zwischen LAN und Internet darstellt, benötigen Sie kein Masquerading auf Paketebene mehr.

SOCKS ist in einer kostenlosen und in einer kommerziellen Version verfügbar. Die SOCKS-Homepage finden Sie auf www.socks.nec.com, die nichtkommerzielle Referenzversion von SOCKS 5 erhalten Sie von <http://www.socks.nec.com/socks5.html>.

7.5 Diverse Systemaccounts in /etc/passwd und /etc/group

Die Dateien für die Benutzerauthentifizierung, `/etc/passwd` und `/etc/group`, enthalten eine Reihe besonderer Systemaccounts. Diese erlauben es den entsprechenden Services, ohne `root`-Privilegien zu laufen, und ermöglichen damit eine genauere Zugriffskontrolle auf Programme und Dateisystembereiche, die für diese Dienste reserviert sind. Je nachdem, was genau Sie installiert haben, sind die meisten dieser Accounts vermutlich unbenutzt, weil die zugehörige Software gar nicht vorhanden ist.

Bei Red Hat sind die Systemaccounts in der Voreinstellung ausgeschaltet. Dazu enthält `/etc/passwd` im Passwortfeld ein Sternchen (*). Dadurch sind Logins als Systembenutzer verboten. Die betreffenden Prozesse starten dennoch mit reduzierter Autorisierung, nicht mit `root`-Rechten.

Während Logins in die Systemaccounts also ohnehin nicht erlaubt sind, kann man als weitere Sicherheitsmaßnahmen `/bin/false` für die benötigten Accounts als Login-Shell definieren und die nicht benötigten Accounts ganz aus `/etc/passwd` und `/etc/group` entfernen. Mindestens benötigen Sie in `/etc/passwd` die folgenden Einträge:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/bin/false
nobody:x:99:99:Nobody:./:/bin/false
```

Warnung

Löschen Sie nicht zu viele Gruppen!

/etc/group enthält neben den Gruppen, die den Systemaccounts zugeordnet sind, noch eine Reihe weiterer Gruppen. Diese sollten Sie nicht aus der Datei entfernen. Sie dienen verschiedenen Subsystemen als Zugriffskontrolle, beispielsweise für die Verzeichnisse mit den Manual-Pages und für bestimmte Geräte wie Konsole, Kernel- und physikalischer Speicher, Disketten und Festplatten sowie für die Terminals.

Die restlichen Systemaccounts dienen besonderen Zwecken. Alle Accounts aus Tabelle 7.1, die Sie nicht brauchen, können Sie aus `passwd` und `group` löschen.

Account	Zugehöriger Server oder entsprechendes Subsystem
daemon	PPP- und POP-Server
lp	Drucker
news	Newsserver
uucp	UUCP-Server
operator	Alias für die Systemadministration
games	diverse Spiele
gopher	Gopher-Server
ftp	ftp-Server
xfs	Font-Server für X-Windows
gdm	Gnome Display Manager
postgres	SQL-Datenbankserver
squid	squid-Webproxy

Tab. 7.1: Systemaccounts

7.6 Die PATH-Variable

Die Umgebungsvariable `PATH` enthält die Verzeichnisse, in denen Ihre Shell ausführbare Programme erwartet. Die einzelnen Verzeichnisse werden genau in der angegebenen Reihenfolge durchsucht.

Diese Suchreihenfolge schafft besonders auf Mehrbenutzersystemen potenzielle Sicherheitsrisiken. Oft ist es nämlich sehr bequem, das aktuelle Verzeichnis `.` in den Pfad mit aufzunehmen. Wenn der Punkt im `PATH` eines Benutzers auftaucht, sollte er an letzter Stelle stehen, niemals vor einem Verzeichnis mit Systemprogrammen. Ansonsten könnte jemand z.B. einen Trojaner mit dem gleichen Namen wie `ls` in einem Verzeichnis ablegen, in dem Sie den `ls`-Befehl wahr-

scheinlich einmal anwenden werden. Das Verzeichnis `.` sollte auf gar keinen Fall im PATH von `root` vorkommen. `root` hat nur die normalen Programmverzeichnisse im Pfad; Software aus anderen Verzeichnissen sollte `root` immer durch explizite Angabe des vollständigen Dateinamens aufrufen.

Die Variable `PATH` wird in den Konfigurationsdateien der Benutzershell definiert. Je nachdem, welche Shell eingesetzt wird, könnte die entsprechende Datei z.B. `.profile`, `.login`, `.cshrc`, `.kshrc`, `.bashrc` usw. heißen.

Ein typischer `PATH` für `root`, im `sh`-Stil angegeben, wäre: `PATH=/bin:/sbin:/usr/bin:/usr/sbin`.

Für einen normalen Benutzer, der X-Windows verwendet, sähe `PATH` z.B. so aus: `PATH=/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:..`

7.7 /etc/issue.net

Wenn sich jemand von einem fremden Rechner aus einloggt, erhält er als Erstes den Inhalt von `/etc/issue.net` angezeigt. Erst anschließend erscheinen die Abfragen für Benutzername und Passwort. Bei lokalen Logins heißt die entsprechende Datei `/etc/issue`.

Einer der problematischen Punkte an `telnet` besteht eben darin, dass die Willkommensmeldung schon angezeigt wird, bevor die Benutzerauthentifizierung auch nur begonnen hat. Jedermann kann einfach eine `telnet`-Verbindung zu Ihrem Rechner herstellen und die entsprechenden Informationen auslesen. In der Voreinstellung liefern die meisten Unix-Versionen – und Red Hat Linux ist keine Ausnahme – neben dem Namen und der Versionsnummer des Betriebssystems auch die Art der CPU zurück. Bei Red Hat Version 6.0 sieht das so aus:

```
Red Hat Linux release 6.0 (Hedwig)
Kernel 2.2.5-15 on an i686
```

Genau das wollen Portscanner herausfinden – wenn `telnet` eingeschaltet ist, bekommen sie die Daten ohne weiteres.

Das Skript `/etc/rc.d/rc.local` von Red Hat erzeugt `/etc/issue` und `/etc/issue.net` bei jedem Bootvorgang neu. Wenn Sie einen offenen `telnet`-Port benötigen, sollten Sie entweder `/etc/rc.d/rc.local` so verändern, dass es weniger freigübig ist, oder den entsprechenden Code ganz aus `rc.local` entfernen und per Hand eine eigene `/etc/issue.net` erstellen.

Eine einfache Willkommensmeldung ergibt sich z.B. aus folgendem Code in `/etc/issue.net`:

```
Welcome to bastion
```

```
%d
```

Das bewirkt folgende Ausgabe an ankommende telnet-Verbindungen:

```
Welcome to bastion
```

```
13:45 on Monday, 24 July 2000
```

7.8 Protokollierung auf andere Rechner

Der Protokolldämon `syslogd` kann alle Logmeldungen an einen anderen Computer schicken. Für einen durchschnittlichen Privatanwender ist das wahrscheinlich unnötig komplex und aufwändig. Wenn eine Site aber eine Architektur wie in Kapitel 4 einsetzt und von internen Rechnern der DMZ aus Dienste anbietet, bietet eine Kopie des `syslogs` auf einem weiteren Rechner zwei Vorteile: Zum einen sind alle Protokolle auf einer einzigen Maschine versammelt und lassen sich damit leichter lesen und überwachen. Zum anderen bleibt die Information auch dann noch erhalten, wenn in einen der Server eingebrochen wird.

Kapitel 8 erklärt, wie wichtig nach einem Einbruch die Informationen aus dem `syslog` sind. Wenn ein Hacker sich erfolgreich den `root`-Zugang zu einem anderen Computer verschafft hat, löscht er als allererstes die Protokolle und installiert trojanische Programme, die seine Aktivitäten nicht mehr aufzeichnen: Genau dann, wenn man sie eigentlich am nötigsten brauchen würde, kann man sich auf die Protokolldateien nicht mehr verlassen, oder sie sind ganz verschwunden. Eine Kopie der Logs auf einem anderen Rechner schützt diese Informationen zumindest bis zu dem Punkt, an dem der Eindringling den Protokolldämon austauscht.

Die Einrichtung dieses Dienstes erfordert auf beiden Rechnern leichte Änderungen.

Auf dem fremden Rechner, der die Logs sammelt, editieren Sie das Startup-Skript `/etc/rc.d/init.d/syslog` und ergänzen beim `syslogd`-Aufruf die Option `-r`. Damit wartet `syslogd` auf dem UDP-Port 514 auf ankommende Protokolleinträge von anderen Rechnern.

Auf dem lokalen Rechner, der die Logs generiert, bearbeiten Sie die Konfigurationsdatei von `syslogd`, `/etc/syslog.conf`. Teilen Sie dem Dämon mit, Meldungen aus welchen Facilities und mit welchen Prioritäten er an den anderen Rechner senden soll. Z.B. bewirkt die folgende Zeile die Weiterleitung aller Meldungen an `hostname`:

```
*.* @hostname
```

7.9 Halten Sie Ihre Software auf dem Laufenden!

Eine der wichtigsten Sicherheitsvorkehrungen besteht darin, die Software des Systems immer auf dem Laufenden zu halten. In ganz besonderem Maße gilt das für die Netzwerkdienste und den Kernel selbst. Besonders kritisch ist das für Open-Source-Systeme wie Linux, bei dem neben den Entwicklern auch die Eindringlinge Zugang zum Quellcode haben.

7.9.1 Updates von Red Hat

Sicherheitsrelevante Patches und Updates für Red Hat Linux beziehen Sie von www.redhat.com, bei den Errata im Support-Bereich. Momentan ist die URL <http://www.redhat.com/support/errata/>. Ich würde Ihnen raten, diesen Bereich wöchentlich auf Sicherheits-Updates zu prüfen.

Tipp

Mailinglisten zum Thema Sicherheit

Red Hat bietet eine Mailingliste zum Thema Sicherheit an, auf der sicherheitsrelevante Updates angekündigt werden. Schreiben Sie eine Mail mit dem Betreff `subscribe an redhat-watch-list-request@redhat.com`.

Natürlich gibt es von SuSE eine ähnliche Liste, die Sie durch eine Mail an `suse-security-announce-subscribe@suse.com` abonnieren können.

7.9.2 Beispiel: Sicherheitslücke in mountd

Im Herbst 1998 wurde im `mountd`-Dämon von NFS ein Buffer-Overflow entdeckt, über den sich ein Hacker `root`-Zugriff auf das System verschaffen konnte. Die betreffende Version von `mountd` war kurz zuvor in Red Hat Version 5.1 veröffentlicht worden. Als die Sicherheitslücke bekannt wurde, verbreiteten sich schnell Programme zu ihrer Ausnutzung über das Internet. Zusammen mit den anderen Linux-Vertreibern gab Red Hat fast sofort einen Patch heraus, der das Problem beseitigte. (Nähere Informationen zum Thema finden Sie im CERT-Advisory <http://www.cert.org/advisories/CA-98.12.mountd.html>.)

Praktisch jeder aus meinem Bekanntenkreis, der `mountd` benutzte und keine Firewall verwendete, wurde gehackt, bevor er das Update installieren konnte.

Diese Sicherheitslücke in `mountd` ist ein Beispiel,

- wie wichtig es ist, immer die aktuellsten Softwareversionen zu verwenden,
- wie eine Buffer-Overflow-Sicherheitslücke funktioniert,

- warum Server, die nicht für das Internet gedacht sind, auch nicht öffentlich zugänglich sein sollten. Diese Sicherheitslücke bestand in `mountd` selbst und gefährdete ein System selbst dann, wenn keine NFS-Dateisysteme gemounted oder exportiert waren.
- weshalb man den externen Zugang auf Dienste und Dämonen des LANs sperren sollte.

7.10 Zusammenfassung

Selbst bei installierter Paketfilter-Firewall kommen Einbrüche in Unix-Systeme vor, wenn auf der Ebene der Systemadministration nicht die nötigen Sicherheitsüberlegungen angestellt und entsprechende Vorsichtsmaßnahmen ergriffen werden. Eine einzelne Schutzschicht, ein einzelner Mechanismus wird niemals vollständige Sicherheit gegen Eindringlinge bieten können. Dieses Kapitel erklärt verschiedene Möglichkeiten der Systemadministration und Server-Konfiguration, die bei der Absicherung Ihres Systems helfen.

Besprochen wurde bei der Authentifizierung, warum Shadow-Passwörter und MD5-Hashes besser sind als der klassische Passwortmechanismus von Unix. Danach ging es im Wesentlichen um die Autorisierung. Die `tcp_wrapper` wurden erklärt. Von den Gefahren frei beschreibbarer Dateien und Verzeichnisse war die Rede und davon, wie Sie ihnen auf die Spur kommen. Anschließend wurden für eine Reihe beliebiger Dienste spezifische Konfigurationsangelegenheiten und Sicherheitsmechanismen vorgestellt, u.a. für `telnet`, `smtp`, `ftp`, `pop` und `DNS`.

Kapitel 8

Entdecken von Eindringlingen und Melden der Vorfälle



- 8.1 Prüfung der Systemintegrität
- 8.2 Welche Symptome deuten auf einen Eindringling hin?
- 8.3 Reaktion auf einen Einbruch
- 8.4 Meldung eines Vorfalls
- 8.5 Wo finden Sie nähere Informationen?
- 8.6 Zusammenfassung

Ein ordentlich konfiguriertes Unix-System kann sehr sicher sein. Bei einem so komplizierten System wie Unix ergeben sich zwar zwangsläufig viele Gefahren, doch kennt man sich mit den Sicherheitsrisiken von Unix auch besser aus als mit denen irgendeines anderen Betriebssystems. Immerhin existiert Unix seit zwanzig Jahren und es hat sich in all der Zeit ständig weiterentwickelt. Unix ist die Grundlage des Internets.

Trotzdem: Man begeht nun einmal Konfigurationsfehler. Bugs in der Software tauchen immer wieder auf. In einer Welt, die nicht perfekt ist, muss man Kompromisse eingehen. Sicherheitsentwickler und Eindringlinge stehen in beständigem Wettbewerb miteinander: Jeder will dem anderen den kleinen, aber entscheidenden Schritt voraus bleiben. Was heute sicher ist, ist es morgen vielleicht nicht mehr. Sie können davon ausgehen, dass eine heute noch sichere Software morgen nicht mehr sicher sein wird.

In Kapitel 7 ging es darum, wie wichtig es ist, immer nur die aktuellen Softwareversionen zu benutzen. Ich hatte Ihnen erzählt, wie unmittelbar nach der Freigabe von Red Hat 5.1 eine Sicherheitslücke im `mountd`-Dämon gefunden und ausgenutzt wurde. Fast sofort erschien ein Update, das die Sicherheitslücke schloss. Dieser Vorfall war aber beileibe nicht der erste seiner Art und er wird auch nicht der letzte bleiben.

Die Techniken der Hacker werden immer raffinierter. Doch wenn in einem bislang sicheren Programm eine Schwäche gefunden wird, dann ist dieses Programm dadurch nicht etwa plötzlich unsicher geworden und wird nie wieder benutzt. Stattdessen korrigiert man die Sicherheitslücke und die Software besteht weiter. Sie ist wieder sicher – für's Erste. Freilich suchen die Eindringlinge immer weiter nach neuen Sicherheitslücken. Deshalb wird man eines Tages vielleicht ein neues Sicherheitsproblem in der gleichen Software entdecken. Es ist ein ewiger Kreislauf.

Ich würde Ihnen ja gerne versprechen, dass Ihr System niemals geknackt werden wird, wenn Sie jeden Vorschlag und jede Vorgehensweise befolgen, die ich in diesem Buch beschreibe. Doch das ist nun einmal nicht der Fall. Es gibt keine Garantien. Systemsicherheit ist kein Zustand, sondern ein fortlaufender Prozess, ein System beständiger Aufmerksamkeit und Aktualisierungen. Sie müssen versuchen, den Black Hats immer einen Schritt voraus zu sein. Die Gefahr wird immer bleiben.

Eine letzte Vorsichtsmaßnahme, die ich Ihnen dringend anrate und die Sie regelmäßig durchführen sollten, betrifft Programme zur Überprüfung der Systemintegrität. In diesem Kapitel stelle ich eine Reihe von kostenlos erhältlichen Prüfprogrammen für die Systemintegrität vor. Man spricht von einem IDS oder *Intrusion Detection System* – einem System zur Erkennung von Eindringlingen.

Am Ende des Kapitels beschreibe ich noch ein paar typische Symptome von Eindringlingen, und ich darf einige Ratschläge für den Fall geben, dass der schlimmste aller Fälle tatsächlich einmal eintritt und jemand in Ihr System ein-

bricht. Sie müssen sich dann unter anderem entscheiden, ob Sie den Vorfall weitermelden.

8.1 Prüfung der Systemintegrität

Programme zur Überprüfung der Systemintegrität spezialisieren sich auf eine Reihe von Tests. Einige sind reine Analyseprogramme. Sie helfen Ihnen, die Konfiguration Ihres Computers zu optimieren. Andere suchen aktiv nach bekannten Sicherheitslücken. Manche vergleichen das System, wie es jetzt ist, mit einem zuvor definierten Zustand und suchen damit nach unerlaubten oder zumindest verdächtigen Veränderungen. All diese Programme sind frei verfügbar: von Software-sammlungen zu Linux, von Web- und ftp-Sites, die sich mit dem Thema Sicherheit beschäftigen, von durch die DARPA geförderten Sicherheitsprojekten oder von Sites der US-Regierung.

8.1.1 COPS

Das *Computer Oracle and Password System* (COPS) von Dan Farmer ist eine Softwaresammlung, die eine riesige Menge potenzieller Sicherheitslücken auf einem Unix-System abtastet. `cops` sollte von `cron` aus regelmäßig aufgerufen werden. Die Ergebnisse schreibt es in eine Berichtsdatei; auf Wunsch sendet es sie zusätzlich per E-Mail an einen bestimmten Benutzer, z.B. an `root`.

`cops` prüft unter anderem die Zugriffsrechte von Dateien und Verzeichnissen, die Qualität der Passwörter, die Konfiguration von `ftp`, die Prüfsummen zentral wichtiger Programme, Zugriffsmöglichkeiten über `tftp`, die Konfiguration von `sendmail` und `inetd` sowie den Zustand diverser Konfigurationsdateien aus `/etc`.

COPS ist ziemlich weit verbreitet. Zwei Bezugsquellen sind <ftp://info.cert.org/pub/tools/cops/cops.1.04.tar.gz> und http://metalab.unc.edu/pub/Linux/system/security/cops_104_linux.tgz.

Weil COPS ein sehr allgemeines Sicherheitsanalyseprogramm für Unix-Systeme aller Art ist, müssen die Konfigurationsdateien und Programme für Ihre spezielle Distribution und Version von Linux angepasst werden.

8.1.2 crack

`crack` ist ein Programm, das Passwörter erraten kann. (Einige der anderen Programme zur Prüfung der Systemintegrität verfügen über eine ähnliche Funktionalität.) Dieses Tool soll einem Systemadministrator dabei helfen, Accounts mit schwachen und leicht zu knackenden Passwörtern zu finden. `crack` verwendet eine Kombination allgemeiner und spezialisierter Wörterbücher sowie diverse Heuristiken zur Suche nach bestimmten Mustern. `crack` kann die `passwd`- und `shadow`-Dateien automatisch zusammenführen, falls Sie Shadow-Passwörter benutzen. Es verwendet die DES-Verschlüsselung, die ursprünglich bei Unix-

Passwörtern zum Einsatz kam. Die Red Hat Linux-Distribution erlaubt Ihnen bei der Installation die Auswahl von MD5-Passwörtern. `crack` unterstützt MD5-Passwörter momentan nicht ohne Weiteres.

`crack` erhalten Sie von <ftp://info.cert.org/pub/tools/crack/crack5.0.tar.gz>.

8.1.3 ifstatus

`ifstatus` überprüft die Konfiguration der Netzwerkschnittstellen. Unter anderem zeigt es alle Interfaces an, die jemand in den Debug-Modus bzw. *Promiscuous Mode* geschaltet hat. So etwas wäre möglicherweise ein Zeichen dafür, dass sich jemand Fremdes Zugang verschafft hat und jetzt den Netzwerkverkehr aufzeichnet. Das ist bei Black Hats eine beliebte Methode, um beispielsweise von `telnet` übertragene Passwörter herauszufinden.

Eine Bezugsquelle für `ifstatus` ist <ftp://coast.cs.purdue.edu/pub/tools/unix/ifstatus>.

Das CERT-Dokument »Ongoing Network Monitoring Attacks« (<http://www.cert.org/advisories/CA-94.01.ongoing.network.monitoring.attacks.html>) enthält weiterführende Informationen zu Angriffen, bei denen jemand die über das Netz gesendeten Daten aufzeichnet.

8.1.4 MD5

MD5 ist ein Algorithmus, der eine kryptografische Prüfsumme zur Sicherung der Datenintegrität erstellt. MD5 generiert aus einer Zeichenfolge oder einer Datei eine 128-Bit-Prüfsumme. Red Hat Linux enthält ein Programm, das MD5-Prüfsummen erzeugt, sowie die entsprechenden C-Bibliotheken für eigene Software. Einige der hier vorgestellten Pakete rund um die Systemintegrität verwenden ihre eigenen MD5-Bibliotheken und erstellen damit Datenbanken mit den Prüfsummen ausgewählter Systemdateien.

8.1.5 SATAN

SATAN, das »Security Administrator Tool for Analyzing Networks« von Wietse Venema und Dan Farmer, hilft bei der Lokalisierung von Schwachstellen in der Konfiguration von Netzdiensten. SATAN sucht nach häufigen Problemen in Bezug auf NFS, NIS, `tftp`, `ftp` und `BSD-r`-Kommandos. Daneben identifiziert es offene Ports durch Portscans. Anschließend erzeugt es einen ausführlichen Bericht über die gefundenen Probleme und liefert Anleitungen, wie sie behoben werden können. SATAN läuft unter Linux nicht auf Anhieb, doch Qualität und Umfang seiner Ausgabe lohnen die Anstrengung. Die Benutzeroberfläche von SATAN ist auch als Web-Interface verfügbar.

Weitere Informationen zu SATAN erhalten Sie von <ftp://ftp.porcupine.org/pub/security/index.html>. SATAN selbst finden Sie an zahlreichen Stellen im Internet,

u.a. bei <ftp://ftp.porcupine.org/pub/security/satan-1.1.1.tar.Z>, <ftp://ftp.net.ohio-state.edu/pub/security/satan/> und <ftp://sunsite.unc.edu/pub/packages/security/Satan-for-Linux/>.

8.1.6 tiger

tiger beinhaltet eine Sammlung von Skripten und C-Programmen, die nach Sicherheitslücken suchen, über die jemand sich root-Zugang zum System verschaffen könnte. tiger überprüft hinsichtlich der Systemkonfiguration z.B. die PATH-Variablen, `inetd.conf`, über NFS exportierte Dateisysteme, ungewöhnliche Dateinamen, Zugriffsrechte auf Dateien und Verzeichnisse sowie `.rhost`-Dateien. Es verwaltet eine Datenbank mit digitalen Signaturen wichtiger Systemdateien, mit deren Hilfe es Veränderungen entdeckt.

tiger ist ziemlich weit verbreitet und beispielsweise von <http://metalab.unc.edu/pub/Linux/system/security/tiger-2.2.4.tgz> oder <http://www.net.tamu.edu/ftp/security/TAMU/tiger-2.2.4p1.tar.gz> erhältlich.

8.1.7 tripwire

tripwire erstellt und verwaltet eine Datenbank mit MD5-Prüfsummen für alle oder ausgewählte Dateien auf Ihrem System. Es will unerlaubte Ergänzungen, Löschungen oder Veränderungen entdecken. Die letzte öffentliche tripwire-Version, 1.2, finden Sie z.B. auf <http://www.cert.org/ftp/tools/tripwire/tripwire-1.2.tar.Z> oder <ftp://ftp.auscert.org.au/pub/coast/COAST/Tripwire/tripwire-1.2.tar.Z>.

tripwire wird aktuell als kommerzielles Produkt von der Firma Tripwire Security Systems (www.tripwiresecurity.com) vertrieben. Die Version 2.0 ist für Red Hat Linux kostenlos.

8.2 Welche Symptome deuten auf einen Eindringling hin?

Wenn sich die Ausgaben der Überwachungsprogramme aus dem letzten Abschnitt unerwartet ändern, besonders wenn eine digitale Signatur plötzlich nicht mehr stimmt oder wenn neue Zugriffsrechte für Dateien oder Verzeichnisse gesetzt wurden, besteht der akute Verdacht auf einen Einbruch in das System. Oft versucht ein erfolgreicher Hacker, seine Spuren zu verwischen. Er will nicht, dass Sie ihn bemerken. Ihr Computer dient ihm schließlich als neue Ausgangsbasis. Zum Glück befindet der Eindringling sich auf einem System, das er nicht kennt, und auch sehr gute Black Hats machen manchmal Fehler. Trotzdem gilt der Leitspruch: Beständige Aufmerksamkeit! Ein Hacker ist möglicherweise viel erfahrener darin, seine Gegenwart zu verschleiern, als Sie es darin sind, ungewöhnliche Zustände Ihres Systems zu erkennen.

Unix-Systeme sind zu vielfältig, zu frei konfigurierbar und zu kompliziert, als dass man eine definitive und erschöpfende Liste mit Einbruchssymptomen

zusammenstellen könnte. Wie bei jeder Diagnostik handelt es sich um Detektivarbeit: Suchen Sie Spuren, wo immer Sie können, und zwar so systematisch wie möglich. Die RFC 2196 («Site Security Handbook») enthält eine Liste denkbarer Indizien. Die »Intruder Detection Checklist« von CERT (http://www.cert.org/ftp/tech_tips/intruder_detection_checklist) weist auf weitere Anomalien hin, ebenso das Dokument »Steps for Recovering from a UNIX Root Compromise« (http://www.cert.org/tech_tips/root_compromise.html).

Die folgenden Abschnitte greifen auf alle drei angesprochenen Listen zurück, wobei Sie alle oder zumindest die meisten der dort enthaltenen Punkte in der einen oder anderen Form wiederfinden werden. Ich habe die Systemanomalien grob in die folgenden Kategorien unterteilt: Hinweise in Bezug auf das `syslog`; Änderungen an der Systemkonfiguration; Inhalt, Zugriffsrechte und Größe von Dateien; Änderungen an Benutzer-Accounts, -Passwörtern und -Zugriffsrechten; aus den Sicherheitszusammenfassungen erkennbare Probleme sowie unerwartete Geschwindigkeitseinbußen. Oft lassen sich Indizien in mehrere dieser Gruppen einsortieren.

8.2.1 Hinweise aus dem `syslog`

Hinweise aus dem `syslog` sind z.B. unerwartete Fehler- und Statusmeldungen, schrumpfende Logdateien, gänzlich fehlende Files oder per E-Mail versandte Statusinformationen.

- **Protokolldateien:** Tauchen hier unerwartete Einträge auf oder verlieren einzelne Dateien an Größe bzw. verschwinden sogar ganz, dann ist wahrscheinlich etwas faul. `/var/log/auth` enthält Aufzeichnungen über den Zugriff auf allgemeine Benutzerkonten, `/var/log/secure` nur privilegierte Informationen. `/var/log/maillog` zeichnet Verbindungen vom und zum Mailserver auf. `/var/log/xferlog` schreibt ftp- und uucp-Verbindungen mit. Die Mehrzahl der `syslog`-Meldungen erscheint in `/var/log/messages`.
- **Statusmeldungen der Dämonen:** Zusätzlich zu den oder anstelle der `syslog`-Meldungen verschicken manche Dämonen (z.B. `crond`) auch Informationen per E-Mail. Ungewöhnliche oder fehlende Mails deuten in diesem Fall auf Probleme hin.
- **Anomale Meldungen auf Konsole und Terminal:** Wenn Sie sich eine Meldung nicht erklären können, deutet das möglicherweise auf die Gegenwart eines Hackers hin.
- **Wiederholte Zugriffsversuche:** Wenn jemand immer wieder einen Login oder einen Zugriff auf eine Datei über ftp oder das Web versucht, ist das verdächtig, besonders wenn CGI-Skripten im Spiel sind oder die Versuche außergewöhnlich hartnäckig erscheinen. Das gilt auch, wenn das System anscheinend alle Zugriffe abweist.

Die in Kapitel 5 vorgestellten Programme zur automatischen Überwachung von Protokolldateien dienen als wertvolle Helfer. Sie können Alarmmeldungen verschicken oder sofort eine andere Reaktion einleiten.

8.2.2 Änderungen der Systemkonfiguration

Hinsichtlich der Systemkonfiguration dienen veränderte Konfigurationsdateien und Skripten, unerwartet laufende Programme und ungewöhnlicherweise aktive Ports und Portzuweisungen ebenso als Hinweis auf unerlaubte Aktivitäten wie Zustandsänderungen der Netzwerkgeräte.

- **cron-Jobs:** Prüfen Sie die Konfigurationsdateien und -skripten auf Veränderungen.
- **Modifizierte Konfigurationsdateien:** Ein Vergleich der Prüfsummen zeigt veränderte Konfigurationsdateien in */etc* an. Die hier gespeicherten Dateien sind für die einwandfreie Funktion des Systems unerlässlich. Änderungen in diesen Files (z.B. in */etc/inetd.conf*, */etc/named.conf* und die zugehörigen Datenbankdateien in */var/named*, */etc/passwd*, */etc/group*, */etc/hosts.equiv* oder in anderen Dateien der Netzwerkkonfiguration) sollten Sie unbedingt näher untersuchen.
- Wenn **ps** Ihnen unerklärliche Services und Prozesse zeigt, ist das ein schlechtes Zeichen.
- Zeigt **netstat** unerwartete Verbindungen und aktive Ports, gilt Alarmstufe eins.
- Ein Absturz eines Servers oder des ganzen Systems ist hochverdächtig.
- Bei der Installation eines Paket-Sniffers wird die Netzwerkkarte in den *Promiscuous-Mode* umgeschaltet.

8.2.3 Änderungen am Dateisystem

In diese Kategorie gehören neue Dateien und Verzeichnisse, fehlende Dateien und Verzeichnisse, veränderte Dateiinhalte, Änderungen in der MD5-Signatur, neue **setuid**-Programme sowie schnell wachsende oder überlaufende Dateisysteme.

- **Neue Dateien und Verzeichnisse:** Besonders verdächtig sind Dateinamen, die mit einem oder mehreren Punkten beginnen, sowie »normale« Dateinamen an ungewöhnlichen Stellen.
- **Neue **setuid**- und **setgid**-Programme** sowie solche, bei denen dieses Bit neu gesetzt wurde, sollten Sie genauer unter die Lupe nehmen.
- Wenn eine Datei fehlt – besonders, wenn es sich um eine Protokolldatei handelt –, deutet das auf Schwierigkeiten hin.

- Nachdem in den Computer eingebrochen wurde, zeigt `df` möglicherweise einen stark ansteigenden Platzbedarf auf den Dateisystemen. Ursache können Werkzeuge des Hackers sein, die große Protokolldateien erzeugen.
- Prüfen Sie die Inhalte Ihrer Web- und `ftp`-Bereiche auf neue oder veränderte Dateien.
- Neue Dateien unter `/dev`: CERT warnt besonders eindringlich vor neuen ASCII-Dateien unterhalb von `/dev`. Dabei handelt es sich typischerweise um Konfigurationsdateien für Programme des Eindringlings.

8.2.4 Änderungen an Benutzer-Accounts

Dazu zähle ich neue Benutzeraccounts, Modifikationen an der `passwd`-Datei, ungewöhnliche oder fehlende Protokolle über Benutzeraktivitäten, Änderungen an Benutzerdateien – besonders Files mit Umgebungsvariablen – sowie gesperrte Accounts.

- Neue und geänderte Benutzer-Accounts in `/etc/passwd` deuten ebenso wie neue oder unerwartete Benutzerkennungen in der `ps`-Ausgabe darauf hin, dass sich jemand einen neuen Zugang geschaffen hat. Fehlt bei einem Account das Passwort, spricht man von einem »offenen« Account.
- Falls bei der Protokollierung der Benutzeraktivitäten Unregelmäßigkeiten auftreten, stecken Sie ebenfalls in Schwierigkeiten. Zum Beispiel könnten unerwartete Logins vorliegen oder Protokolldateien (z.B. `/var/log/lastlog`, `/var/log/pacct` oder `/var/log/ustracct`) könnten fehlen oder bearbeitet worden sein.
- Modifikationen an den Accounts von `root` sowie der anderen Benutzer, besonders Änderungen in der Login-Umgebung, sind sehr ernsthafte Warnungen. Gefährlich sind vor allem Modifikationen an den Dateien `.rhost` und `.forward` sowie an der Umgebungsvariablen `PATH`.
- Nicht mehr zugängliche Accounts fallen ebenfalls in diese Gruppe, ob jetzt das Passwort verändert, der Account gelöscht oder der Runlevel des Systems auf den Einzelplatzmodus umgestellt wurde.

8.2.5 Hinweise der Überwachungswerkzeuge

Die diversen Überwachungswerkzeuge warnen Sie, wenn die Prüfsumme einer Datei plötzlich nicht mehr stimmt, wenn sich Größe oder Zugriffsrechte einer Datei ändern oder wenn neue `setuid`- und `setgid`-Programme auftauchen.

Die Prüfsumme einer Datei verändert sich für neue Dateien sowie für Files mit geänderter Größe, anderem Zeitstempel oder neu gesetzten Zugriffsrechten. Hier befürchtet man vor allem die Installation eines trojanischen Pferds. Häufig tauscht der Angreifer dabei die Programme aus, die von `/etc/inetd.conf` gestartet werden, außerdem `ls`, `ps`, `netstat`, `ifconfig`, `telnet`, `login`, `su`, `ftp`, `inetd`, `syslogd`, `du`, `df` und `sync` sowie die `libc`.

8.2.6 Geschwindigkeitseinbußen

Zur Kategorie der Geschwindigkeitseinbußen rechne ich eine ungewöhnlich hohe Systemlast sowie massive Festplattengeräusche.

Ursachen hierfür sind z.B. ungewöhnliche oder ungewöhnlich viele Prozesse, exzessiver Datenaustausch mit dem Internet oder erhöhte Zugriffe auf das Dateisystem.

Wenn Sie auf Ihrem System Zeichen eines erfolgreichen Einbruchversuchs feststellen, geraten Sie bitte nicht in Panik. Rebooten Sie den Rechner nicht – dabei könnten wichtige Daten verloren gehen. Trennen Sie das System einfach physikalisch vom Internet.

8.3 Reaktion auf einen Einbruch

Die mögliche Vorgehensweise nach einem erfolgreichen Einbruchversuch wird in den beiden Dokumenten »Steps for Recovering from a UNIX Root Compromise« (erhältlich von CERT unter http://www.cert.org/tech_tips/root_compromise.html) sowie RFC 2196 (»Site Security Handbook«) erörtert. Beide Dokumente präsentieren eine recht formelle Vorgehensweise, die z.B. für eine Firma, eine Behörde oder eine Universität adäquat ist. Sie setzen eine gewisse Menge an freiem Speicher voraus, in dem Kopien des Systems aufbewahrt werden, dazu freie Zeit von Mitarbeitern, die das Sicherheitsproblem analysieren und diagnostizieren können. Es geht dabei um Situationen, in denen das Opfer möglicherweise Klage vor einem Gericht erheben möchte.

Bei kleinen Sites, deren Internetzugang über einen öffentlichen, verbraucherorientierten Internet-Provider erfolgt, passiert normalerweise Folgendes: Die Site wird geknackt, ohne dass der Eigentümer das bemerkt; sie dient in der Folge als Ausgangsbasis für Angriffe gegen andere Sites; jemand beschwert sich beim Provider und der Endverbraucher erfährt von dem Einbruch erst dadurch, dass sein Provider anruft und den Internetzugang mit ein paar bösen Worten sperrt. Sobald der Benutzer den Provider davon überzeugt hat, dass in sein System eingebrochen wurde, das Problem mittlerweile aber behoben ist, gibt der Provider den Zugang normalerweise wieder frei. Darüber können durchaus einige Wochen vergehen.

Wenn also der schlimmste aller denkbaren Fälle eintritt: Was tun Sie, wenn Sie bemerken, dass jemand in Ihr System eingebrochen hat? Wie schon gesagt, trennen Sie als Erstes die Maschine vom Internet.

Rebooten Sie bitte noch nicht. Wenn der Hacker Programme manuell installiert oder gestartet hat, zerstört ein Reboot alle Informationen über den aktuellen Zustand des Systems.

Wenn Sie über den entsprechenden Speicherplatz verfügen, fertigen Sie eine Kopie des gesamten Systems im aktuellen Zustand an. Diese können Sie später analysieren. Wenn das für Sie nicht möglich ist, erstellen Sie zumindest eine Kopie der Protokolldateien aus `/var/log` und der Konfigurationsdateien aus `/etc`.

Legen Sie sich ein Protokoll Ihres Vorgehens auf Papier an. Schreiben Sie alles auf, was Sie tun und was Sie herausfinden. Das ist nicht nur wichtig, wenn Sie den Vorfall einer Sicherheitsbehörde, Ihrem Internetprovider oder einem Rechtsanwalt darlegen wollen. Es hilft Ihnen auch, weil Sie genau wissen, was Sie schon untersucht haben und was Sie noch tun müssen.

Die Vorgehensweise bei der Analyse eines kompromittierten Systems ist im Wesentlichen die gleiche, wie wenn Sie herausfinden wollen, ob das System kompromittiert ist.

1. Prüfen Sie die Protokolldateien. Welche Prozesse laufen? Welche Ports werden verwendet? Prüfen Sie den Inhalt der Konfigurationsdateien. Testen Sie die Prüfsummen aller Dateien und Verzeichnisse und verifizieren Sie Inhalt und Zugriffsrechte. Suchen Sie nach neuen `setuid`-Programmen. Vergleichen Sie die Konfigurations- und Benutzerdateien mit einwandfreien Kopien von einem Backup.

Es ist sehr wahrscheinlich, dass der Eindringling anstelle der Werkzeuge, mit denen Sie das System analysieren, trojanische Pferde installiert hat. Geben Sie also Acht!

2. Schreiben Sie alle vergänglichen Informationen auf. Welche Prozesse laufen, welche Ports werden benutzt?
3. Booten Sie von einer Bootdiskette oder einem Backup Ihres Systems. Untersuchen Sie Ihr altes System mit zuverlässigen Tools von dem nicht betroffenen Backup-System.
4. Finden Sie heraus, wie der Hacker sich Zugang zu Ihrem System verschafft und was er getan hat.
5. Installieren Sie Ihren Computer von den Originalmedien Ihrer Linux-Distribution komplett neu.
6. Korrigieren Sie die Sicherheitslücke, indem Sie z.B. diesmal die laufenden Dienste sorgfältiger auswählen, Ihre Server sicherer konfigurieren, auf der Ebene der `tcp_wrapper` oder der einzelnen Server Zugriffskontrolllisten festlegen, eine Paketfilter-Firewall installieren oder einen Proxy-Server einrichten.
7. Schalten Sie alle Protokolloptionen ein.
8. Restaurieren Sie Konfigurationsdateien, von denen Sie wissen, dass der Eindringling sie nicht angetastet hat.
9. Installieren Sie alle Sicherheitsupdates Ihres Linux-Distributors. Installieren und konfigurieren Sie Pakete zur Überwachung der Systemintegrität. Berechnen Sie MD5-Prüfsummen für alle Programme und speichern Sie die entstehende Datenbank auf einer Diskette oder auf einem anderen Computer.
10. Überwachen Sie das System und achten Sie darauf, ob der Eindringling erneut einen Zugriff versucht.

Die meisten Leute, mit denen ich gesprochen habe, fühlen sich nach einem erfolgreichen Einbruch schuldig und ein bisschen dumm. Aber denken Sie daran: Sicherheit ist ein fortwährender Wettlauf zwischen den Eindringlingen auf der einen Seite und den Administratoren auf der anderen. Sie haben den Black Hat nicht eingeladen. Er hat die Schwächen Ihres Systems gezielt gesucht – vermutlich mit einiger Mühe – und sie ausgenutzt. Die Schuld für ein Verbrechen liegt nicht beim Opfer. Und: Sie sind nicht alleine. In viele, viele Systeme wird eingebrochen. Versuchen Sie, das nächste Mal noch ein bisschen besser darauf zu achten, was die bösen Jungs tun.

8.4 Meldung eines Vorfalls

Unter »Vorfall« kann man alles mögliche verstehen; Sie müssen die Bedeutung dieses Wortes für sich selbst festlegen. Die Palette der Konnotationen reicht von der allgemeinen Beschreibung »jeder anomale Zugriffsversuch aus dem Internet« bis hin zu einer restriktiveren Definition, die das Wort beschränkt auf Dinge wie einen erfolgreichen unerlaubten Login, einen erfolgreichen Denial-of-Service-Angriff, die erfolgreiche Ausnutzung einer Sicherheitslücke bzw. die erfolgreiche und illegale Inanspruchnahme der Ressourcen und Services Ihres Systems.

Als Administrator eines ans Internet angeschlossenen Computers sollten Sie Ihre Protokolldateien regelmäßig überwachen, ebenso die Berichte Ihrer Sicherheits-Tools und die Logs der Benutzeraktivitäten. Selbst wenn Sie nur minimale Mengen protokollieren, werden Sie früher oder später sicherlich etwas sehen, das Sie melden wollen. Wenn Sie alle Log-Optionen einschalten, werden Sie 24 Stunden am Tag Stoff zum Nachdenken finden.

Einige Zugriffsversuche sind ernster zu nehmen als andere und einige werden Sie persönlich mehr verärgern als andere.

In den folgenden Abschnitten geht es zunächst darum, warum Sie überhaupt einen Vorfall melden wollen und welche Sorte von Vorfällen man sinnvollerweise meldet. Diese Entscheidungen sind jedoch sehr individuell. Falls Sie sich dazu entschließen, etwas zu melden, stelle ich anschließend einige der Gruppen vor, denen Sie die Vorfälle melden können. Ich berichte über die Informationen, die Sie dabei unbedingt angeben müssen.

8.4.1 Warum sollten Sie Vorfälle melden?

Möglicherweise gibt es auch dann einen guten Grund für die Meldung eines Vorfalls, wenn dieser nicht von Erfolg gekrönt war. Zum Beispiel:

- Weil Sie die Scans unterbinden wollen. Ihre Firewall garantiert zwar, dass die meisten Abtastversuche harmlos bleiben. Wenn sie fortwährend auftreten, sind aber auch ungefährliche Proben lästig. Sie füllen Ihre Protokolldateien. Je nachdem, wie Sie Ihre Analyseprogramme konfiguriert haben, werden Sie evtl. sogar ständig mit warnenden E-Mails belästigt.

- Weil Sie andere schützen wollen. Automatisierte Proben und Scans dienen im Allgemeinen dem Aufbau einer Datenbank aller verwundbaren Rechner in einem großen IP-Adressblock. Nachdem die Hosts als anfällig identifiziert worden sind, unterzieht man sie im Folgenden selektiven Angriffen. Die raffinierten Hackertools der heutigen Zeit können innerhalb von Sekunden in ein verwundbares System eindringen und alle Spuren verwischen. Wenn Sie den Scan melden, wird der Angreifer vielleicht gestoppt, bevor er irgendwo irgendwem Schaden zufügt.
- Weil Sie den Administrator informieren wollen. Angreifende Sites wurden oft selbst kompromittiert. Auf ihnen hat ein Hacker einen Account, es läuft falsch konfigurierte Software, ihre Adresse ist gefälscht oder einer der legitimen Benutzer ist einfach ein Aufrührer. Der Administrator wird in einem solchen Fall meistens dankbar für einen Hinweis sein. Auch Internet-Provider klemmen Kunden, die sich nicht benehmen können, lieber frühzeitig ab, bevor andere Kunden sich beschweren, dass fremde Sites den Zugriff für diesen Provider gesperrt haben und plötzlich kein E-Mail-Austausch mehr funktioniert.
- Weil Sie eine Bestätigung des Angriffs wollen. Manchmal sind Sie sich vielleicht nicht so ganz sicher, ob ein bestimmter Eintrag aus einer Protokolldatei nun auf ein Problem hinweist oder nicht. Manchmal wünschen Sie sich eine Bestätigung, dass eine andere Site nur wegen einer fehlerhaften Konfiguration problematische Pakete verschickt. Diese Site wird oft dankbar sein, dass Sie sie auf ein Problem mit ihrer Netzwerkkonfiguration hinweisen.
- Weil Sie in allen Beteiligten ein verstärktes Bewusstsein für die Problematik wecken wollen. Wenn Sie die angreifende Site über den Vorfall unterrichten, wird man dort in Zukunft hoffentlich besser auf die Konfiguration und die Aktivitäten der eigenen Benutzer achten. Wenn Sie eine Missbrauch-Abteilung informieren, kann diese den Sünder mit mehr Nachdruck ansprechen, als es einem Einzelnen möglich wäre. Sie wird darüber hinaus auf fortlaufende Aktivitäten achten und ist so in die Lage versetzt, kompromittierten Kunden besser zu helfen. Wenn Sie an eine sicherheitsbezogene News-Gruppe schreiben, weiß jeder Leser besser darüber Bescheid, worauf er Acht geben muss.

8.4.2 Welche Vorfälle sollten Sie melden?

Welche Vorfälle Sie melden, hängt von Ihrer Toleranz ab, von Ihrer Einschätzung, wie ernst der Vorfall war, und davon, wieviel Zeit Sie in die Bekämpfung dieser weltweit wachsenden Seuche investieren wollen. Letztlich läuft alles darauf hinaus, was Sie unter einem »Vorfall« verstehen. Unterschiedliche Leute meinen damit mal einen einfachen Portscan, mal einen Zugriffsversuch auf private Dateien und Ressourcen, mal einen Denial-of-Service-Angriff, mal einen Absturz eines Server-Programms oder eines ganzen Computers, mal einen unerlaubten Zugriff auf den `root`-Account.

- Denial-of-Service-Angriffe: Alle Denial-of-Service-Angriffe sind massiv feindlicher Natur. Es ist ziemlich schwierig, so etwas nicht als persönlichen Angriff zu werten. Hierbei handelt es sich um die elektronische Form von Vandalismus, Belästigung und Diebstahl. Weil manche Varianten solcher Angriffe in der Eigenart der jeweils betroffenen Dienste begründet sind, kann man sich oft nicht gut dagegen wehren, außer indem man den Vorfall weitermeldet und den Angreifer komplett aussperrt.
- Versuche, Ihr System umzukonfigurieren: Ein Hacker kann Ihre Server ohne Zugriff auf den `root`-Account nicht umkonfigurieren, aber er kann durchaus die netzwerkbezogenen Tabellen umschreiben – oder das zumindest versuchen. In diese Kategorie gehören:
 - Unerlaubte DNS-Zone-Transfers von oder auf Ihren Computer über TCP. Weitere Informationen zum Thema finden Sie in *DNS & BIND* von Albitz und Liu (O'Reilly).
 - Änderungen Ihrer dynamischen Routing-Tabellen über ICMP-Redirects oder über Proben des UDP-Ports 520 (*routed* oder *gated*). Denken Sie daran: Dynamisches Routing sollte auf einer Firewall ausgeschaltet sein! Weitere Informationen zu dieser Sorte Angriff finden Sie in *Firewalls and Internet Security: Repelling the Wily Hacker* von Cheswick und Bellovin (Addison-Wesley).
 - Versuche, die Konfiguration Ihrer Netzwerkkarten oder Ihre Routing-Tabellen über SNMP und den UDP-Port 161 zu modifizieren.
- Versuche, Daten über die lokale Konfiguration und Netzwerktopologie abzufragen. Solche Anfragen richten sich meistens an SNMP auf dem UDP-Port 161. DNS-Anfragen an den TCP-Port 53 liefern Informationen über die Netzwerktopologie; ebenso Pakete an den UDP-Port 520, wenn dort die Dämonen *routed* oder *gated* laufen.
- Versuche, einen Login zu erlangen. Verbindungsversuche zum `telnet`-Port 23 oder zum `ssh`-Port 22 sind offensichtlich. Weniger klar ist die Situation, wenn der Angreifer einen Server kontaktiert, zu dem man eine aktuelle oder historische Sicherheitslücke kennt. Normalerweise versucht er dann einen Buffer-Overflow, der ihm letztlich einen Shell-Zugang mit der Möglichkeit liefert, beliebige Befehle auszuführen. Der weiter oben besprochene *mountd*-Angriff ist ein Beispiel.
- Versuche, private Dateien zu lesen. Beispiele sind `/etc/passwd`, Konfigurationsdateien oder private Daten. Die entsprechenden Zugriffsversuche erscheinen in den Übertragungsprotokollen des `ftp`-Servers (`/var/log/xferlog` oder `/var/log/messages`) und des Webservers (`/var/log/httpd/error_log`).
- Versuche, private Dienste zu nutzen. *Per definitionem* gehören dazu alle Dienste, die Sie nicht frei anbieten wollen. Ich beziehe mich hierbei besonders auf private Dienste, die auf Ihren öffentlich zugänglichen Servern zumindest

potenziell tatsächlich verfügbar sind, z.B. auf einen Mailserver. Wenn jemand Mail über Ihren Mailserver statt über den seines Internet-Providers verschicken will, hat er wohl nichts Gutes im Sinn. Derartige »Angriffe« erscheinen in der Protokolldatei für E-Mail (`/var/log/maillog`).

- Versuche, Dateien auf Ihrer Festplatte abzulegen. Wenn Sie eine falsch konfigurierte anonyme ftp-Site betreiben, ist darüber der Austausch von Raubkopien (»WAREZ«) realisierbar. Wenn jemand Ihnen eine Datei schickt und Sie Ihren ftpd entsprechend konfiguriert haben, erscheint eine Meldung im ftp-Log (`/var/log/xferlog`).
- Versuche, einzelne Server-Programme oder Ihren ganzen Computer abstürzen zu lassen. Buffer-Overflows gegen CGI-Skripten auf Ihrer Website lassen sich wohl am einfachsten identifizieren: Es erscheint eine Fehlermeldung in den Logdateien des CGI-Skripts. Andere Fehlermeldungen suchen Sie in der allgemeinen syslog-Datei (`/var/log/messages`) oder in den spezifischen Protokolldateien der Dämonen (`/var/log/daemon`), des E-Mail-Subsystems (`/var/log/maillog`), des FTP-Subsystems (`/var/log/xferlog`) oder im Protokoll der privilegierten Zugriffe (`/var/log/secure`).
- Versuche, spezifische, bekannte Sicherheitslücken auszunutzen. In jeder neuen Software-Version finden Hacker Sicherheitslücken. Bleiben Sie anhand der Advisories von www.cert.org und von Ihrem Linux-Distributor auf dem Laufenden. In der letzten Zeit lagen die am häufigsten ausgenutzten Sicherheitslücken in den folgenden Systemen: sunrpc bzw. mount, pop3, imap, socks, ftp, CGI-Skripten, Mailversand über smtp sowie in den BSD-r-Befehlen zur Programmausführung auf fremden Computern.

8.4.3 Wem melden Sie?

Für die Meldung eines Vorfalls existieren verschiedene sinnvolle Ansprechpartner:

- Die Accounts `root`, `postmaster` und `abuse` auf der Site des Angreifers. Natürlich ist der Administrator der betreffenden Site der erste Ansprechpartner für derartige Probleme. Oft reicht ein Hinweis an ihn völlig aus. Leider wird das nicht immer möglich sein. Viele Scans kommen von gefälschten IP-Adressen, die in der Realität nicht existieren. Wenn Sie `nslookup` auf die Absenderadresse anwenden, erhalten Sie eine Meldung, dass der Host oder die Domain nicht gefunden werden können.
- Der Netzwerkkoordinator. Wenn die IP-Adresse nicht auflösbar ist, hilft es oft, den Koordinator des zugehörigen Netzwerkblocks zu informieren. Dieser kann Kontakt zum Administrator der betreffenden Site aufnehmen oder Sie zu ihm durchstellen. Wenn `nslookup` keine Daten über die IP-Adresse liefert, finden Sie den Netzwerkkoordinator normalerweise über die `whois`-Datenbanken. Alle vier `whois`-Datenbanken sind auch über das Web zugänglich:

- ARIN: Die *American Registry for Internet Numbers* ist für Nord- und Südamerika zuständig (<http://whois.arin.net/whois/arinwhois.html>).
- APNIC: Das *Asia Pacific Network Information Centre* verwaltet die IP-Adressen Asiens (<http://www.apnic.net/reg.html>).
- RIPE: Die europäische Registry sind die *Réseaux IP Européens* (<http://www.ripe.net/db/whois.html>).
- NIRPNET: Die Whois-Datenbank des amerikanischen Verteidigungsministeriums ist für militärische Netze verantwortlich (<http://nic.mil/cgi-bin/whois>).
- Die Missbrauchsstelle Ihres Internet-Providers. Wenn der Scan vom Adressbereich Ihres eigenen Providers ausgeht, sollten Sie zunächst dessen Missbrauchsstelle kontaktieren. Ihr Provider hilft Ihnen auch beim Umgang mit fremden Scans, indem er für Sie den Übeltäter herausfindet. Meistens sind Sie nicht der einzige Kunde, der angegriffen wird.
- CERT: Das CERT-Koordinationszentrum verfügt normalerweise nicht über die Ressourcen, auf normale Vorfälle zu reagieren. Seine Prioritäten liegen im Bereich weltweiter Probleme, großer Institutionen und Notsituationen, die das gesamte Internet betreffen. Trotzdem begrüßt CERT Berichte über Vorfälle aller Art, die es für Überwachungszwecke und für seine Statistik verwendet. CERT erreichen Sie über http://www.cert.org/contact_cert/contactinfo.html oder per E-Mail unter cert@cert.org.
- Der Hersteller Ihrer Linux-Distribution: Wenn jemand in Ihr System einbricht, weil eine Schwäche in der mitgelieferten Software besteht, dann interessiert sich auch Ihr Linux-Distributor dafür. Er wird ein Sicherheits-Update entwickeln und veröffentlichen.

8.4.4 Was melden Sie?

Bei der Meldung eines Vorfalls müssen Sie so ausreichende Informationen angeben, dass die betreffende Einrichtung oder Behörde das Problem weiterverfolgen kann. Denken Sie beim Ansprechen der angreifenden Site aber auch daran, dass Ihre Kontaktperson möglicherweise mit dem Übeltäter identisch ist! Welche Daten aus der folgenden Liste Sie in die Meldung mit aufnehmen, hängt mithin stark davon ab, wem Sie schreiben und wie wohl Ihnen dabei ist, die Daten freizugeben.

- Ihre E-Mail-Adresse
- gegebenenfalls Ihre Telefonnummer
- Ihre IP-Adresse, Ihr Host- und Domainname
- Die am Angriff beteiligten IP-Adressen, Host- und Domainnamen, sofern verfügbar

- Zeit und Datum des Vorfalls. Geben Sie auch Ihre Zeitzone (relativ zu GMT, in Deutschland also »+0100« oder »+0200«) an.
- Eine Beschreibung des Angriffs
- Wie Sie den Angriff bemerkt haben
- Repräsentative Auszüge aus Ihren Protokolldateien, die den Vorfall demonstrieren
- Eine Beschreibung des Formats der Protokolldateien
- Verweise auf Advisories und Sicherheitsdokumente, die Art und Schweregrad des Angriffs erörtern
- Was Sie von der angesprochenen Person erwarten, z.B. das Problem abstellen, bestätigen, erklären oder überwachen – oder Sie senden eine Nachricht »nur zu Ihrer Information«

8.5 Wo finden Sie nähere Informationen?

In dem Maße, wie Netzwerksicherheit ein öffentliches Thema wird, nimmt die Zahl sicherheitsbezogener Websites rasch zu. CERT und COAST sind nach wie vor die wichtigsten Fundorte für sicherheitsrelevante Informationen und Software. Darüber hinaus stellen diverse Abteilungen v.a. der US-amerikanischen Regierung riesige Mengen an Informationen zum Thema zur Verfügung. Die folgende Liste enthält einige der bekanntesten, größten, ältesten und offiziellsten Websites:

- <http://www.cert.org/> – das CERT-Koordinationszentrum an der Informatikabteilung der Carnegie Mellon University. CERT verwaltet Advisories, Hinweise, Mailing-Listen, Sicherheitsdokumente, relevante Softwaresammlungen sowie jährliche und längerfristige Zusammenfassungen der Black-Hat-Aktivität im Internet.
- <http://www.first.org/> – FIRST ist das *Forum of Incident Response and Security Teams*. Auf der Website finden Sie eine Liste der für Firmen, Universitäten und Regierungsbehörden zuständigen Ansprechpartner.
- <http://www.cerias.purdue.edu/> – von CERIAS (*Center for Education and Research in Information Assurance and Security*) stammt u.a. das COAST-Projekt (<http://www.cs.purdue.edu/coast/>). Hier finden Sie Software und zahlreiche Dokumente.
- <http://ciac.llnl.gov/> – die CIAC (*Computer Incident Advisory Capability*) stellt eigentlich nur einen Ansprechpartner für Sites des amerikanischen Energieministeriums (DOE) dar. Auf seinen Webseiten finden Sie jedoch zahlreiche Dokumente und Tutorials sowie Software und Links auf andere sicherheitsrelevante Sites.

- <http://csrc.nist.gov/> – NIST steht für das *National Institute of Standards and Technology*, die amerikanische Normungsbehörde. Sie veröffentlicht im Web diverse sicherheitsbezogene Texte.
- <http://www.fedcirc.gov/> – FedCIRC (*Federal Computer Incident Response Capability*) versorgt Rechenzentren der amerikanischen Regierung mit allgemeiner Unterstützung und Hilfe bei der Aufklärung von Vorfällen. Auch FedCIRC bietet auf der Homepage zahlreiche Hinweise und Dokumente sowie Links und Software an.

8.6 Zusammenfassung

In diesem Kapitel ging es primär um die Überwachung der Systemintegrität sowie das Entdecken von Eindringlingen. Von vielen Web- und ftp-Sites erhältliche Analyseprogramme wurden vorgestellt, u.a. COPS, crack, SATAN, tiger und tripwire. Dabei sahen Sie, wie wichtig digitale Prüfsummen sind, wie Sie sie z.B. von MD5 oder tripwire erhalten. Es wurde besprochen, welche potenziellen Hinweise auf einen Eindringling hindeuten. Eine Checkliste half in dem Fall, dass Sie einige dieser Indizien auf Ihrem System wiederfinden und den Schluss ziehen, dass in Ihre Site eingebrochen wurde. Schließlich erörterten wir Überlegungen zur Meldung von Vorfällen. Dabei erhielten Sie Hinweise auf mögliche Ansprechpartner.



Teil IV

Anhänge

Kapitel A	Ressourcen rund um das Thema Sicherheit
Kapitel B	Firewall-Beispiele und hilfreiche Skripten
Kapitel C	Glossar

Anhang A

Ressourcen rund um das Thema Sicherheit



- A.1 Informationsquellen
- A.2 Softwaresammlungen
- A.3 Sicherheitstools
- A.4 Firewall-Tools
- A.5 Nachschlagewerke und FAQ-Listen
- A.6 Online-Dokumentation
- A.7 Web-Sites zu allgemeinen Themen
- A.8 Bücher

Dieser Anhang bietet eine Übersicht wichtiger Internet-Ressourcen rund um das Thema Sicherheit: Hinweise und Informationen, Tools, Updates und Patches ... Natürlich existieren noch viele weitere Sites, und jeden Tag entstehen viele neue; sehen Sie diese Links also eher als Ausgangspunkt an, nicht als vollständige Liste.

A.1 Informationsquellen

Sicherheitsinformationen aller Art, Hinweise und Alarmmeldungen, White Papers, Einführungen usw. finden Sie an den folgenden Stellen:

- CERT Coordination Center:
<http://www.cert.org/>
- CIAC (Computer Incident Advisory Capability):
<http://ciac.llnl.gov/ciac/>
- COAST Internet Firewalls-Resources:
<http://www.cs.purdue.edu/coast/firewalls/>
- COAST Security Archive:
<ftp://coast.cs.purdue.edu/pub/>
- Sicherheitsseite von Dave Dittrich:
<http://www.washington.edu/People/dad/>
- Firewall Wizards Mailing List Archive:
<http://www.nfr.net/firewall-wizards/fwsearch.html>
- Internet Engineering Task Force (IETF):
<http://www.ietf.cnri.reston.va.us/home.html>
- »Linux Firewall and Security Site«:
<http://linux-firewall-tools.com/linux>
- Linux-Security-Ressourcen:
<http://www.linux-security.org/>
- Matt's UNIX Security:
<http://www.deter.com/unix/>
- Sicherheitsinfos des NIH:
<http://www.alw.nih.gov/Security/>

- Sicherheitsseite von Red Hat:
<http://www.redhat.com/LinuxIndex/Administration/Security/>
- SANS Institute:
<http://www.sans.org/>

A.2 Softwaresammlungen

Die folgenden Sites spezialisieren sich auf sicherheitsbezogene Software aller Art:

- CERT-Archiv:
<http://www.cert.org/ftp/tools/>
- COAST Security-Tools:
<http://www.cs.purdue.edu/coast/coast-tools.html>
- FreeFire Security-Tools:
<http://sites.inka.de/sites/lina/freefire-l/tools.html>
- Linux Router Project:
<http://www.linuxrouter.org/>
- Linuxberg Software:
<http://idirect.Linuxberg.com/software.html>
- Masquerading Applications:
<http://www.tsmservices.com/masq>
- Security-Bereich von Sunsite:
<ftp://sunsite.unc.edu/pub/Linux/system/security/>

A.3 Sicherheitstools

Softwaretools für den Bereich Sicherheit werden an vielen Stellen vorgehalten. Die folgende Liste enthält einige der beliebtesten Tools mit Bezugsquellen.

- argus :
<ftp://ftp.sei.cmu.edu/pub/argus/>
- COPS:
ftp://sunsite.unc.edu/pub/Linux/system/security/cops_104_linux.tgz

<ftp://coast.cs.purdue.edu/pub/tools/unix/cops>

<http://www.cert.org/ftp/tools/cops/>

- crack:

<http://www.cert.org/ftp/tools/crack/>

<http://www.cert.org/ftp/tools/cracklib/>

<ftp://coast.cs.purdue.edu/pub/tools/unix/crack>

<ftp://coast.cs.purdue.edu/pub/tools/unix/cracklib/>

- dig:

<ftp://coast.cs.purdue.edu/pub/tools/unix/dig/>

- icmpinfo:

<ftp://coast.cs.purdue.edu/pub/tools/unix/netmon/icmpinfo/>

<http://www.deter.com/unix/software/icmpinfo-1.10.tar.gz>

- ifstatus:

<ftp://coast.cs.purdue.edu/pub/tools/unix/ifstatus/>

- IPSec:

<http://www.xs4all.nl/~freeswan/>

- ISS:

<http://www.deter.com/unix/software/iss13.tar.gz>

- Linux IP Firewalling Chains:

<http://www.rustcorp.com/linux/ipchains/>

- Linux IPFW Firewall Design Tool:

<http://linux-firewall-tools.com/linux/firewall/>

- logcheck:

<ftp://coast.cs.purdue.edu/pub/tools/unix/logcheck/>

- lsof:

<ftp://coast.cs.purdue.edu/pub/tools/unix/lsof/>

- md5:

<http://www.cert.org/ftp/tools/md5/>

<ftp://coast.cs.purdue.edu/pub/tools/unix/md5/>

- netcat:
<ftp://coast.cs.purdue.edu/pub/tools/unix/netcat/>
- nmap:
<ftp://coast.cs.purdue.edu/pub/tools/unix/nmap/>
- SATAN:
<ftp://coast.cs.purdue.edu/pub/tools/unix/satan/satan/satan-1.1.1.tar.Z>
<http://www.fish.com/~zen/satan/satan.html>
- sbscan:
<ftp://sunsite.unc.edu/pub/Linux/system/security/sbscan-0.05.tar.gz>
- SINUS-Firewall-Seite:
<http://www.ifi.unizh.ch/ikm/SINUS/firewall/>
- SOCKS:
<http://www.socks.nec.com>
- SSH:
<http://www.ssh.fi/sshprotocols2/>
<http://www.freessh.org/>
<http://www.openssh.com/>
- SSL:
<http://psych.psy.uq.oz.au/~ftp/Crypto/>
- strobe:
<ftp://coast.cs.purdue.edu/pub/tools/unix/strobe/>
- tiger (Linux):
<ftp://sunsite.unc.edu/pub/Linux/system/security/tiger-2.2.4.tgz>
<http://www.net.tamu.edu/ftp/security/TAMU/tiger-2.2.4.tgz>
- TIS-Firewall-Toolkit:
<http://www.tis.com/research/software/>
- tripwire:
<http://www.tripwiresecurity.com/>
<ftp://coast.cs.purdue.edu/pub/tools/unix/Tripwire/>
<http://www.cert.org/ftp/tools/tripwire/>

A.4 Firewall-Tools

- Firewall Design Tool:
<http://linux-firewall-tools.com/linux/firewall/>
- FWCONFIG:
<http://www.mindstorm.com/~sparlin/fwconfig.shtml>
- IP Filter:
<http://cheops.anu.edu.au/~avalon/>
- ipfwadm Dotfile-Modul:
<http://www.wolfenet.com/~jhardin/ipfwadm.html>
- Isinglass PPP Firewall:
<http://www.tummy.com/isinglass/>
- Mason:
<http://www.pobox.com/~wstearns/mason/>
- SINUS-Firewall:
<http://www.ifl.unizh.ch/ikm/SINUS/firewall/>
- TIS-Firewall-Toolkit:
<ftp://ftp.tis.com/pub/firewalls/toolkit/>

A.5 Nachschlagewerke und FAQ-Listen

Die meisten Nachschlagewerke in den folgenden, themenorientierten Abschnitten werden von CERT verwaltet und auf zahlreichen anderen Sites gespiegelt.

A.5.1 Sicherheit unter Unix

- »Anonymous FTP Abuses« (Missbrauch von anonymem FTP):
http://www.cert.org/ftp/tech_tips/anonymous_ftp_abuses
- »ANONYMOUS FTP CONFIGURATION GUIDELINES« (Richtlinien für die Konfiguration von anonymem FTP):
http://www.cert.org/ftp/tech_tips/anonymous_ftp_config
- »Denial of Service Attacks« (Denial-of-Service-Angriffe):
http://www.cert.org/ftp/tech_tips/denial_of_service

- »Intruder Detection Checklist« (Checkliste, wie man einen Eindringling entdeckt):
http://www.cert.org/ftp/tech_tips/intruder_detection_checklist
- »Problems with the FTP PORT Command« (Probleme mit dem PORT-Befehl von FTP):
http://www.cert.org/ftp/tech_tips/FTP_PORT_attacks
- »Protecting Yourself from Password File Attacks« (Wie Sie sich vor Angriffen auf die Passwortdatei schützen):
http://www.cert.org/ftp/tech_tips/passwd_file_protection
- »Steps for Recovering from a UNIX Root Compromise« (Wie Sie Ihr System wiederherstellen, nachdem jemand den root-Account geknackt hat):
http://www.cert.org/tech_tips/root_compromise.html
- »UNIX Computer Security Checklist« (Checkliste Unix-Sicherheit):
http://www.cert.org/ftp/tech_tips/AUSCERT_checklist1.1
- »UNIX Configuration Guidelines« (Richtlinien für die Konfiguration eines Unix-Systems):
http://www.cert.org/ftp/tech_tips/UNIX_configuration_guidelines

A.5.2 Rund um Firewalls

- »Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing« (Denial-of-Service-Angriffe mit gefälschten IP-Absenderadressen abwehren):
<http://www.ietf.cnri.reston.va.us/rfc/rfc2267.txt>
- »Firewall Policy Guide (ICSA)« (Firewall-Sicherheitsrichtlinien):
<http://www.icsa.net/services/consortia/firewalls/fwpg.shtml>
- »Internet Firewalls Frequently Asked Questions« (FAQs zu Internet-Firewalls):
<http://www.clark.net/pub/mjr/pubs/fwfaq/>
- »IP Masquerade for Linux« (IP-Masquerading unter Linux):
<http://www.tor.shaw.wave.ca/~ambrose/>
- »Linux Firewall & LAN Security FAQ«:
<http://linux-firewall-tools.com/linux/faq/>

- »Linux firewall facilities for kernel-level packet screening« (Firewall-Unterstützung für Paketfilter auf Kernelebene unter Linux):
<http://www.xos.nl/linux/ipfwadm/paper/>
- Linux IP Masquerading Web Site:
<http://www.indyramp.com/masq/>
- »Packet Filtering for Firewall Systems« (Paketfilter für Firewall-Computer):
http://www.cert.org/ftp/tech_tips/packet_filtering
- Port-Forwarding:
<http://www.ox.compsoc.org.uk/~steve/portforwarding.html>
- PORTUS Firewall Tutorial:
<http://www.lsl.com/tutorial.html>
- »TCP SYN Flooding and IP Spoofing Attacks« (CERT-Advisory CA-96.21):
http://www.cert.org/advisories/CA-96.21.tcp_syn_flooding.html
- Liste der TCP- und UDP-Portnummern (IANA):
<http://www.isi.edu/in-notes/iana/assignments/port-numbers>

A.5.3 Web-Server

- Sicherheitstipps zum Apache Web-Server:
http://www.apache.org/docs/misc/security_tips.html
- »How To Remove Meta-characters From User-Supplied Data In CGI Scripts« (Wie man in CGI-Skripten Sonderzeichen aus Benutzerdaten entfernt):
http://www.cert.org/ftp/tech_tips/cgi_metacharacters
- »The World Wide Web Security FAQ«:
<http://www.w3.org/Security/Faq/www-security-faq.html>

A.6 Online-Dokumentation

Eine typische Linux-Distribution enthält eine sehr umfassende Dokumentation in unterschiedlichen Formaten. Die hier aufgeführten HTML-Dokumente sind besonders wichtig für das Thema Netzwerksicherheit:

- Firewalling and Proxy Server HOWTO:
</usr/doc/HOWTO/other-formats/html/Firewall-HOWTO.html>

- Alle HOWTOs:
`/usr/doc/HOWTO/other-formats/html/HOWTO-INDEX.html`
- Linux IP Masquerade mini HOWTO:
`/usr/doc/HOWTO/other-formats/html/mini/IP-Masquerade.html`
- Linux IPCHAINS-HOWTO:
`/usr/doc/HOWTO/other-formats/html/IPCHAINS-HOWTO.html`
- Linux Security HOWTO:
`/usr/doc/HOWTO/other-formats/html/Security-HOWTO.html`
- Network Administrator's Guide (NAG):
`/usr/doc/LDP/nag/nag.html`
- System Administrators' Guide (SAG):
`/usr/doc/LDP/sag/sag.html`

A.7 Web-Sites zu allgemeinen Themen

Sowohl Informationen zu als auch Software für Linux finden Sie im Web an allen Ecken und Enden. Die folgenden Links empfehle ich Ihnen als besonders gute Ausgangspunkte:

- CableModem Info:
<http://www.cablemodeminfo.com/>
- FreshMeat:
<http://freshmeat.net/>
- Allgemeine Linux-Links:
<http://www.emuse.net/>
- Linux-Anwendungen und -Utilities:
<http://www.xnet.com/~blatura/linapps.shtml>
- Linux-Anwendungen:
<http://www.linuxapps.com/>
- Das Linux-Dokumentation-Projekt:
<http://metalab.unc.edu/LDP/>
- »Linux Powered«:
http://www.linuxpowered.com/html/linux_links/netw.html

- »Linux Start«:
<http://linuxstart.com/>
- »Linux Today«:
<http://linuxtoday.com/>
- »Linux World«:
<http://www.linuxworld.com/>
- Red Hat:
<http://www.redhat.com/>
- SlashDot:
<http://slashdot.org/>
- SUNET (Swedish University Network):
<http://ftp.sunet.se/pub/os/Linux/>
- Sunsite:
<http://metalab.unc.edu/pub/Linux/>

A.8 Bücher

Ken A. L. Coar: *Apache Server for Dummies*. Foster City, CA: IDG Books Worldwide, Inc., 1998.

D. Brent Chapman, Simon Cooper und Elizabeth D. Zwicky: *Building Internet Firewalls*. Sebastopol, CA: O'Reilly & Associates, Inc., 2000.

William R. Cheswick und Steven M. Bellovin: *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 1994.

Paul G. Sery: *Linux Network Toolkit*. Foster City, CA: IDG Books Worldwide, Inc., 1998.

Anhang B

Firewall-Beispiele und hilfreiche Skripten



- B.1 Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3
- B.2 Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3
- B.3 Optimierung der Firewall-Regeln
- B.4 Spezialskripten

Kapitel 3 beschreibt eine Firewall für ein Einzelplatzsystem bzw. für ein privates LAN. Kapitel 4 ergänzt dieses Beispiel zu einer Bastion-Firewall mit vollständigen Firewallregeln sowohl für das externe Interface zum Internet als auch für das interne Interface zum LAN. Gleichzeitig wird die Einzelplatzlösung aus Kapitel 3 zu einer sekundären Choke-Firewall erweitert. Die Bastion dient als Gateway ins Internet sowie zu einer DMZ mit öffentlichen Servern. Die Choke agiert als Gateway zwischen einem privaten LAN auf der einen Seite und der DMZ oder der Bastion auf der anderen Seite, je nach Design der DMZ.

Die Beispiel-Firewall wird in Kapitel 3 Stück für Stück vorgestellt. In diesem Anhang ist das gleiche Beispiel im Zusammenhang abgedruckt, so wie es in der Praxis in einem Firewall-Skript Verwendung fände. Dabei wird einmal die Notation von `ipchains` und einmal die von `ipfwadm` benutzt. Beide Beispiele sind wiederum nicht optimiert. Die Fassungen für `ipchains` und `ipfwadm` werden fast kongruent dargeboten.

Die Optimierung von Firewallregeln ist an sich nicht Thema dieses Buches. Sowohl für `ipchains` als auch für `ipfwadm` werden anhand eines einfachen Firewall-Skripts einige grundlegende Optimierungen exemplarisch vorgestellt.

Schließlich finden Sie hier noch ein paar hilfreiche Skripten. Unix-Systeme mögen es überhaupt nicht gerne, wenn sich ihre IP-Adresse ändert. Besonders DNS erwartet eigentlich, dass die Hostadresse stabil bleibt. Viele Computer erhalten aber über DHCP eine dynamische IP-Adresse, vor allem private Geräte, die über eine Flatrate oder DSL an das Internet angeschlossen sind. Dabei muss man sowohl DNS als auch dem Firewall-Skript ein wenig unter die Arme greifen, wenn eine IP-Adresse neu zugewiesen wird oder sie sich später ändert, während die Maschine online ist.

B.1 Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

In Kapitel 3 werden die Anwendungsprotokolle und Firewall-Regeln vorgestellt, die Sie auf einem Einzelplatzsystem unter Linux am wahrscheinlichsten anwenden. Wenn so ein Rechner ein kleines, privates LAN mit dem Internet verbindet, leitet die Firewall alle Pakete zwischen dem LAN und dem Internet weiter und übt gleichzeitig Masquerading aus. Kapitel 3 zeigt beispielhaft viele Sicherheitsmaßnahmen und Protokollfunktionen, die für eine funktionale Firewall nicht wirklich notwendig sind. Darüber hinaus sind Client- und Serverregeln enthalten, die nicht jeder braucht.

Es folgt das vollständige Firewall-Skript, wie es auf einem realen System in `/etc/rc.d/rc.firewall` stünde.

```
#!/bin/sh

echo "Initialisiere die Firewall ... "

# Ein paar Definitionen, die uns die Arbeit erleichtern

# -----
#  PASSEN SIE DIESEN ABSCHNITT AN IHR SYSTEM UND IHREN PROVIDER AN

EXTERNAL_INTERFACE="eth0"           # Interface zum Internet
LOOPBACK_INTERFACE="lo"             # Loopback
LAN_INTERFACE_1="eth1"              # internes LAN-Interface

IPADDR="meine.IP.Adresse"           # Ihre IP-Adresse
LAN_1="192.168.1.0/24"              # privater Adressbereich
LAN_IPADDR_1="192.168.1.1"          # Adresse des internen Interfaces

ANYWHERE="any/0"                    # passt auf jede IP

DHCP_SERVER="mein.DHCP.Server"      # falls Sie einen benutzen
MY_ISP="Adressbereich.meines.ISPs"  # Adressbereich von ISP und NOC
NAMESERVER_1="erster.Name.Server"   # jeder braucht wenigstens einen

SMTP_SERVER="any/0"                 # externer Mail-Server
SMTP_GATEWAY="Server.des.ISPs"      # externes Mail-Relay
POP_SERVER="ext.POP.Server"         # externer POP-Server, falls vorhanden
IMAP_SERVER="ext.IMAP.Server"       # externer IMAP-Server, falls vorhanden
NEWS_SERVER="ext.News.Server"       # externer Newsserver, falls vorhanden
WEB_PROXY_SERVER="mein.WWW.Proxy"   # Web-Proxy des Providers, falls vorhanden
WEB_PROXY_PORT="WWW.Proxy.Port"     # Port des Proxies, falls vorhanden
                                     # meistens 8008 oder 8080

LOOPBACK="127.0.0.0/8"              # reservierter Loopback-Bereich
CLASS_A="10.0.0.0/8"                # private Netze der Klasse A
CLASS_B="172.16.0.0/12"             # private Netze der Klasse B
CLASS_C="192.168.0.0/16"            # private Netze der Klasse C
CLASS_D_MULTICAST="224.0.0.0/4"     # Multicast-Adressen der Klasse D
CLASS_E_RESERVED_NET="240.0.0.0/5"  # reservierte Adressen der Klasse E
BROADCAST_SRC="0.0.0.0"             # Broadcast-Absender
BROADCAST_DEST="255.255.255.255"    # Broadcast-Empfänger
PRIVPORTS="0:1023"                  # privilegierter Portbereich
UNPRIVPORTS="1024:65535"            # unprivilegierter Portbereich

TRACEROUTE_SRC_PORTS="32769:65535"
TRACEROUTE_DEST_PORTS="33434:33523"

# -----
```

412 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# Wenn Sie Ihre IP-Adresse dynamisch von einem DHCP-Server erhalten,  
# stehen die Nameserver in /etc/dhcpc/resolv.conf. Gegebenenfalls wird das  
# ifdhcpc-done-Skript o.ä. diese automatisch updaten und an  
# /etc/dhcpc/hostinfo-$EXTERNAL_INTERFACE oder  
# /etc/dhcpc/dhcpd-$EXTERNAL_INTERFACE.info anhängen.
```

```
# Wenn Sie das ifdhcpc-done-Skript aus dem Beispiel verwenden,  
# werden die folgenden NAMESERVER-Definitionen (eine pro Server,  
# höchstens drei) korrekt überschrieben.
```

```
# Die IP-Adresse $IPADDR wird von DHCP definiert.
```

```
if [ -f /etc/dhcpc/hostinfo-$EXTERNAL_INTERFACE ]; then  
    . /etc/dhcpc/hostinfo-$EXTERNAL_INTERFACE  
elif [ -f /etc/dhcpc/dhcpd-$EXTERNAL_INTERFACE.info ]; then  
    . /etc/dhcpc/dhcpd-$EXTERNAL_INTERFACE.info  
    DHCP_SERVER=$DHCP_SIADDR  
else  
    echo "rc.firewall: DHCP nicht konfiguriert."  
    ipchains -F  
    ipchains -P input DENY  
    ipchains -P output DENY  
    ipchains -A input -i $LOOPBACK_INTERFACE -j ACCEPT  
    ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT  
    ipchains -A input -i $LAN_INTERFACE -j ACCEPT  
    ipchains -A output -i $LAN_INTERFACE -j ACCEPT  
    exit 1  
fi
```

```
# -----  
# TRAGEN SIE HIER DIE ZAHL DER ERLAUBTEN SERVER BZW. VERBINDUNGEN EIN.
```

```
# Die von X-Windows benutzten Ports beginnen bei 6000. Jeder weitere  
# Server verwendet den nächsthöheren Port, bis zu 6063.
```

```
XWINDOW_PORTS="6000"                # (TCP) X Window
```

```
# SSH beginnt mit 1023 und belegt für weitere gleichzeitig bestehende  
# Verbindungen die nächstniedrigeren Ports, bis zu 513.
```

```
SSH_PORTS="1020:1023"                # vier gleichzeitige Verbindungen
```

```
# -----
```

```
SOCKS_PORT="1080"                    # (TCP) SOCKS  
OPENWINDOWS_PORT="2000"              # (TCP) OpenWindows  
NFS_PORT="2049"                      # (TCP/UDP) NFS
```

```
# -----

# Vorherbestehende Regeln aus den Chains löschen.
ipchains -F

# Voreinstellung: Pakete ablehnen.
ipchains -P input    DENY
ipchains -P output   REJECT
ipchains -P forward  REJECT

# Masquerade: Timeout für TCP-Verbindungen ist zehn Stunden.
ipchains -M -S 36000 0 0

# Fragmentierte Pakete ablehnen.
ipchains -A input -f -i LAN_INTERFACE_1 -j DENY

# TCP-SYN-Cookies einschalten
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Schutz vor gefälschten IPs:
# Prüfung der Absenderadresse einschalten
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# ICMP-Redirects ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# vom Absender geroutete Pakete ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# Module für Masquerading laden
#/sbin/modprobe ip_masq_ftp.o
#/sbin/modprobe ip_masq_raudio.o
#/sbin/modprobe ip_masq_irc.o
#/sbin/modprobe/ip_masq_vdolive.o
#/sbin/modprobe/ip_masq_cuseeme.o
#/sbin/modprobe/ip_masq_quake.o

# -----
# LOOPBACK

# Keine Einschränkungen für das Loopback-Interface.
```

414 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipchains -A input -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT

# -----
# Verbindungen von problematischen Sites ablehnen.

# /etc/rc.d/rc.firewall.blocked enthält eine Liste gesperrter Adressen
# im Format
# ipchains -A input -i $EXTERNAL_INTERFACE -s <address/mask> -j DENY

# Pakete von einem Absender in der Blockadeliste ablehnen.
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

# -----
# Gefälschte und illegale Adressen ablehnen.
# Gefälschte Pakete sowie völlig illegale Absenderadressen ablehnen.
# Wir wollen auch nicht an fehlerhafte Empfänger senden.

# Gefälschte Pakete von unserer eigenen Adresse ablehnen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $IPADDR -j DENY -1

# Pakete mit einem privaten Klasse-A-Netz in Absender oder Empfänger.
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_A -j DENY -1

# Pakete mit einem privaten Klasse-B-Netz in Absender oder Empfänger.
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_B -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_B -j DENY -1

# Pakete mit einem privaten Klasse-C-Netz in Absender oder Empfänger.
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY
ipchains -A input -i $EXTERNAL_INTERFACE -d $CLASS_C -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY -1
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_C -j DENY -1

# Pakete von Loopback ablehnen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY
ipchains -A output -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY -1

# Fehlerhafte Broadcast-Pakete ablehnen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j DENY -1
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j DENY -l
ipchains -A output -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC -j DENY -l
```

Klasse-D-Multicast-Adressen ablehnen.

Multicast ist nur als Absender illegal; es benutzt UDP.

```
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST \
-j DENY -l
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST \
-j REJECT -l
```

Reservierte Adressen der Klasse E ablehnen.

```
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_E_RESERVED_NET \
-j DENY -l
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -d $CLASS_E_RESERVED_NET \
-j REJECT
```

Von der IANA reservierte Adressen ablehnen.

0.*.*.*, 1.*.*.*, 2.*.*.*, 5.*.*.*, 7.*.*.*, 23.*.*.*, 27.*.*.*

31.*.*.*, 37.*.*.*, 39.*.*.*, 41.*.*.*, 42.*.*.*, 58-60.*.*.*

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 1.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 2.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 5.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 7.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 23.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 27.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 31.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 37.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 39.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 41.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 42.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 58.0.0.0/7 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 60.0.0.0/8 -j DENY -l
```

65: 01000001 - /3 würde 64 beinhalten - deshalb sind 65-79 explizit angegeben

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 65.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 66.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 67.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 68.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 69.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 70.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 71.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 72.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 73.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 74.0.0.0/8 -j DENY -l
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 75.0.0.0/8 -j DENY -l
```

416 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipchains -A input -i $EXTERNAL_INTERFACE -s 76.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 77.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 78.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 79.0.0.0/8 -j DENY -1

# 80: 01010000 - /4 beinhaltet 80-95
ipchains -A input -i $EXTERNAL_INTERFACE -s 80.0.0.0/4 -j DENY -1

# 96: 01100000 - /4 beinhaltet 96-111
ipchains -A input -i $EXTERNAL_INTERFACE -s 96.0.0.0/4 -j DENY -1

# 126: 01111110 - /3 würde 127 einschließen, deshalb 112-126 explizit
ipchains -A input -i $EXTERNAL_INTERFACE -s 112.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 113.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 114.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 115.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 116.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 117.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 118.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 119.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 120.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 121.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 122.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 123.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 124.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 125.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 126.0.0.0/8 -j DENY -1

# 217: 11011001 - /5 würde 216 einschließen, deshalb 217-219 explizit
ipchains -A input -i $EXTERNAL_INTERFACE -s 217.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 218.0.0.0/8 -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -s 219.0.0.0/8 -j DENY -1

# 223: 11011111 - /6 beinhaltet 220-223
ipchains -A input -i $EXTERNAL_INTERFACE -s 220.0.0.0/6 -j DENY -1

# -----
# ICMP

# (4) Source_Quench
#      Bremsst ankommende und abgehende Pakete (Flusskontrolle)

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
-s $ANYWHERE 4 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
-s $IPADDR 4 -d $ANYWHERE -j ACCEPT
```


Anhang B • Firewall-Beispiele und hilfreiche Skripten **417**

```
# (12) Parameter_Problem
#     Ankommende und abgehende Fehlermeldungen

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 12 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 12 -d $ANYWHERE -j ACCEPT

# (3) Dest_Unreachable, Service_Unavailable
#     Größenaushandlung, Dienst oder Empfänger nicht verfügbar,
#     letzte Antwort auf traceroute

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 3 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 3 -d $MY_ISP -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR fragmentation-needed -d $ANYWHERE -j ACCEPT

# (11) Time_Exceeded
#     Ankommender oder abgehender Timeout,
#     Antwort einer Zwischenstation auf traceroute

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 11 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 11 -d $MY_ISP -j ACCEPT

# Abgehende pings sind immer erlaubt.

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 8 -d $ANYWHERE -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 0 -d $IPADDR -j ACCEPT

# Ankommende pings nur von vertrauenswürdigen Hosts.

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $MY_ISP 8 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 0 -d $MY_ISP -j ACCEPT
```

418 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# -----  
  
# UNPRIVILEGIERTE PORTS  
# Manche Ports würden Protokoll- oder Administrationsschwierigkeiten beschwören.  
  
# OpenWindows: Verbindungsaufbau  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \  
-s $IPADDR \  
-d $ANYWHERE $OPENWINDOWS_PORT -j REJECT  
  
# OpenWindows: ankommende Verbindung  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $IPADDR $OPENWINDOWS_PORT -j DENY  
  
# X Window: Verbindungsaufbau  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \  
-s $IPADDR \  
-d $ANYWHERE $XWINDOW_PORTS -j REJECT  
  
# X Window: ankommende Verbindung  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $IPADDR $XWINDOW_PORTS -j DENY -l  
  
# SOCKS: Verbindungsaufbau  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \  
-s $IPADDR \  
-d $ANYWHERE $SOCKS_PORT -j REJECT -l  
  
# SOCKS: ankommende Verbindung  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $IPADDR $SOCKS_PORT -j DENY  
  
# NFS: TCP-Verbindungen  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $IPADDR $NFS_PORT -j DENY -l  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \  
-d $ANYWHERE $NFS_PORT -j REJECT -l  
  
# NFS: UDP-Verbindungen  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-d $IPADDR $NFS_PORT -j DENY -l  
  
# NFS: ankommende Verbindung (normaler UDP-Modus)  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-d $ANYWHERE $NFS_PORT -j REJECT -l
```

```
# -----
# HINWEIS:
#     Die symbolischen Portnamen aus /etc/services sind nicht in
#     jeder Distribution gleich. Ihre Verwendung ist aber weniger
#     fehleranfällig und macht das Skript lesbarer.

# -----
# Notwendige Dienste

# DNS-Client (53)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR $UNPRIVPORTS \
-d $NAMESERVER_1 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s $NAMESERVER_1 53 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# TCP Anfragen von einem Client sind vom Protokoll erlaubt,
# wenn eine UDP-Anfrage fehlschlägt. Das kommt selten vor.
# TCP wird meistens von sekundären Nameservern für einen
# Zone-Transfer vom primären Nameserver benutzt, und von
# Hackern.

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $NAMESERVER_1 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $NAMESERVER_1 53 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# DNS-Server (53)
# -----

# DNS Forwarding- und Caching-Nameserver
# -----

# Anfrage vom Server.
# Ein Server verwendet nur UDP, nicht TCP

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
-s $IPADDR 53 \
-d $NAMESERVER_1 53 -j ACCEPT
```

420 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $NAMESERVER_1 53 \  
-d $IPADDR 53 -j ACCEPT
```

```
# DNS: voller Nameserver  
# -----
```

```
# Client an Server  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s DNS.Clients $UNPRIVPORTS \  
-d $IPADDR 53 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 53 \  
-d DNS.Clients $UNPRIVPORTS -j ACCEPT
```

```
# Peer-to-Peer-Transaktion zwischen Servern  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s DNS.Clients 53 \  
-d $IPADDR 53 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 53 \  
-d DNS.Clients 53 -j ACCEPT
```

```
# Zone-Transfers  
# Wegen der potenziellen Gefahren eines Zone-Transfers  
# sollten Sie TCP-Pakete nur an ausgewählte sekundäre  
# Server zulassen.
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s sekundärer.Server $UNPRIVPORTS \  
-d $IPADDR 53 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 53 \  
-d sekundärer.Server $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# AUTH-Client (113)  
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 113 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 113 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
# AUTH-Server (113)  
# -----  
  
# Eingehende Anfragen annehmen  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR 113 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 113 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT  
  
# OR  
  
# Eingehende Anfragen ablehnen  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-d $IPADDR 113 -j REJECT  
  
# -----  
# TCP-Dienste auf ausgewählten Ports  
  
# SMTP-Client sendet ohne lokalen Server an einen Provider-Computer  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $SMTP_GATEWAY 25 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ SMTP_GATEWAY 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
# -----  
# ODER Mailversand durch lokalen SMTP-Server  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 25 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 25 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

422 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# Mailempfang als lokaler SMTP-Server (25)
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 25 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 25 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

# -----

# POP (110) - Mailempfang als POP-Client
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $POP_SERVER 110 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $POP_SERVER 110 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# POP (110) - ein POP-Server für fremde Clients
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.POP.Clients $UNPRIVPORTS \
-d $IPADDR 110 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 110 \
-d IPs.meiner.POP.Clients $UNPRIVPORTS -j ACCEPT

# -----

# IMAP (143) - Mailempfang als IMAP-Client
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d IP.meines.IMAP.Servers 143 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s IP.meines.IMAP.Servers 143 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# IMAP (143) - ein IMAP-Server für fremde Clients
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.IMAP.Clients $UNPRIVPORTS \
-d $IPADDR 143 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 143 \
-d IPs.meiner.IMAP.Clients $UNPRIVPORTS -j ACCEPT

# -----

# NNTP (119) - News als Usenet-Client lesen und schreiben
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $NEWS_SERVER 119 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $NEWS_SERVER 119 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# NNTP (119) - ein Newsserver für fremde Clients
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s IPs.meiner.News.Clients $UNPRIVPORTS \
-d $IPADDR 119 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 119 \
-d IPs.meiner.News.Clients $UNPRIVPORTS -j ACCEPT

# NNTP (119) - Newsfeeds für einen lokalen Newsserver
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d IP.meines.News.Feeds 119 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s IP.meines.News.Feeds 119 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

424 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# -----  
  
# TELNET (23) - abgehende Verbindungen zu fremden Sites  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 23 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 23 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
# TELNET (23) - ankommende Verbindungen zu Ihrem Server  
# -----  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR 23 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 23 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT  
  
# -----  
  
# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 22 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 22 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $SSH_PORTS \  
-d $ANYWHERE 22 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 22 \  
-d $IPADDR $SSH_PORTS -j ACCEPT
```



```
# SSH-Server (22) - ankommende Verbindungen zu Ihrem SSH-Server
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 22 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 22 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $SSH_PORTS \
-d $IPADDR 22 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 22 \
-d $ANYWHERE $SSH_PORTS -j ACCEPT

# -----

# FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern
# -----

# Abgehende Verbindung

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 21 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 21 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE 20 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus
```

426 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# FTP-Server (20, 21) - ankommende Verbindungen zu Ihrem FTP-Server
# -----

# Ankommende Verbindung

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 21 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 21 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

# Datenkanalaufbau im aktiven Modus

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR 20 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 20 -j ACCEPT

# Datenkanalaufbau im passiven Modus

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

# -----

# HTTP (80) - Zugriff auf fremde Sites als Client
# -----
```

Anhang B • Firewall-Beispiele und hilfreiche Skripten 427

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 80 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 80 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# HTTP (80) - fremde Zugriffe auf einen lokalen Server
# -----
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 80 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 80 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# HTTPS (443) - Zugriff auf fremde Sites über SSL als Client
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 443 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 443 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# HTTPS (443) - fremde Zugriffe auf einen lokalen SSL-Web-Server
# -----
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 443 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 443 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

428 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# HTTP Proxy-Client (8008/8080)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
        -s $IPADDR $UNPRIVPORTS \
        -d $WEB_PROXY_SERVER $WEB_PROXY_PORT -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
        -s $WEB_PROXY_SERVER $WEB_PROXY_PORT \
        -d $IPADDR $UNPRIVPORTS -j ACCEPT

# -----

# FINGER (79) - Zugriff auf fremde finger-Server als Client
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
        -s $IPADDR $UNPRIVPORTS \
        -d $ANYWHERE 79 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
        -s $ANYWHERE 79 \
        -d $IPADDR $UNPRIVPORTS -j ACCEPT

# FINGER (79) - fremde Zugriffe auf einen lokalen finger-Server
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
        -s IPs.meiner.finger.Clients $UNPRIVPORTS \
        -d $IPADDR 79 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
        -s $IPADDR 79 \
        -d IPs.meiner.finger.Clients $UNPRIVPORTS -j ACCEPT

# -----

# WHOIS-Client (43)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
        -s $IPADDR $UNPRIVPORTS \
        -d $ANYWHERE 43 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
        -s $ANYWHERE 43 \
        -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----  
  
# Gopher-Client (70)  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 70 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 70 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
# -----  
  
# WAIS-Client (210)  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 210 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 210 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT  
  
# -----  
# UDP nur auf ausgewählten Ports annehmen  
  
# TRACEROUTE  
# traceroute verwendet meistens -S 32769:65535 -D 33434:33523  
# -----  
  
# Abgehende traceroutes  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR $TRACEROUTE_SRC_PORTS \  
-d $ANYWHERE $TRACEROUTE_DEST_PORTS -j ACCEPT  
  
# Ankommende traceroutes vom Internet-Provider.  
# Alle anderen werden abgelehnt.  
# -----  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $MY_ISP 32769:65535 \  
-d $IPADDR 33434:33523 -j ACCEPT
```

430 — Die rc.firewall für ipchains für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# -----  
  
# DHCP-Client (67, 68)  
# -----  
  
# INIT oder REBINDING: Keine IP oder Gültigkeit der IP abgelaufen  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $BROADCAST_0 68 \  
-d $BROADCAST_1 67 -j ACCEPT  
  
# Eine neue IP holen  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $BROADCAST_0 67 \  
-d $BROADCAST_1 68 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $DHCP_SERVER 67 \  
-d $BROADCAST_1 68 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $BROADCAST_0 68 \  
-d $DHCP_SERVER 67 -j ACCEPT  
  
# Im Ergebnis ändern wir unsere IP-Adresse. Das Paket geht  
# an unsere neue Adresse, bevor der DHCP-Client das Update  
# erhalten hat.  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $DHCP_SERVER 67 \  
-d $MY_ISP 68 -j ACCEPT  
  
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $DHCP_SERVER 67 \  
-d $IPADDR 68 -j ACCEPT  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR 68 \  
-d $DHCP_SERVER 67 -j ACCEPT  
  
# -----  
  
# NTP (123) - Zugriff auf fremde Network Time Server  
# -----  
  
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR $UNPRIVPORTS \  
-d mein.Zeit.Server 123 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
-s mein.Zeit.Server 123 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# -----
# Keine lokalen Einschränkungen.

# Alle internen Computer dürfen auf die Firewall zugreifen.

ipchains -A input -i $LAN_INTERFACE_1 \
-s $LAN_1 -j ACCEPT

ipchains -A output -i $LAN_INTERFACE_1 \
-d $LAN_1 -j ACCEPT

# -----
# Interne Adressen durch Masquerading verbergen.

# Alle internen Pakete werden durch Masquerading geschickt.

ipchains -A forward -i $EXTERNAL_INTERFACE -s $LAN_1 -j MASQ

# -----

echo "fertig"

exit 0
```

B.2 Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

Alle Firewall-Beispiele aus diesem Buch beziehen sich auf Red Hat Linux 6.0 mit ipchains. Auf vielen Computern laufen aber noch frühere Linux-Versionen, in denen Firewalls über ipfwadm realisiert wurden. Dieser Abschnitt enthält das gleiche vollständige Firewall-Skript für /etc/rc.d/rc.firewall, diesmal aber für ipfwadm.

```
#!/bin/sh

echo "Initialisiere die Firewall ... "

# Ein paar Definitionen, die uns die Arbeit erleichtern

# -----
# PASSEN SIE DIESEN ABSCHNITT AN IHR SYSTEM UND IHREN PROVIDER AN
```

432 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
EXTERNAL_INTERFACE="eth0"           # Interface zum Internet
LOOPBACK_INTERFACE="lo"             # Loopback
LAN_INTERFACE_1="eth1"              # internes LAN-Interface

IPADDR="meine.IP.Adresse"           # Ihre IP-Adresse
LAN_1="192.168.1.0/24"              # privater Adressbereich
LAN_IPADDR_1="192.168.1.1"          # Adresse des internen Interfaces

ANYWHERE="any/0"                    # passt auf jede IP

DHCP_SERVER="mein.DHCP.Server"      # falls Sie einen benutzen
MY_ISP="Adressbereich.meines.ISPs"  # Adressbereich von ISP und NOC
NAMESERVER_1="erster.Name.Server"   # jeder braucht wenigstens einen

SMTP_SERVER="any/0"                 # externer Mail-Server
SMTP_GATEWAY="Server.des.ISPs"      # externes Mail-Relay
POP_SERVER="ext.POP.Server"         # externer POP-Server, falls vorhanden
IMAP_SERVER="ext.IMAP.Server"       # externer IMAP-Server, falls vorhanden
NEWS_SERVER="ext.News.Server"       # externer Newsserver, falls vorhanden
WEB_PROXY_SERVER="mein.WWW.Proxy"   # Web-Proxy des Providers, falls vorhanden
WEB_PROXY_PORT="WWW.Proxy.Port"     # Port des Proxies, falls vorhanden
                                     # meistens 8008 oder 8080

LOOPBACK="127.0.0.0/8"              # reservierter Loopback-Bereich
CLASS_A="10.0.0.0/8"                # private Netze der Klasse A
CLASS_B="172.16.0.0/12"              # private Netze der Klasse B
CLASS_C="192.168.0.0/16"             # private Netze der Klasse C
CLASS_D_MULTICAST="224.0.0.0/4"      # Multicast-Adressen der Klasse D
CLASS_E_RESERVED_NET="240.0.0.0/5"   # reservierte Adressen der Klasse E
BROADCAST_SRC="0.0.0.0"              # Broadcast-Absender
BROADCAST_DEST="255.255.255.255"     # Broadcast-Empfänger
PRIVPORTS="0:1023"                  # privilegierter Portbereich
UNPRIVPORTS="1024:65535"            # unprivilegierter Portbereich

TRACEROUTE_SRC_PORTS="32769:65535"
TRACEROUTE_DEST_PORTS="33434:33523"

# -----

# Wenn Sie Ihre IP-Adresse dynamisch von einem DHCP-Server erhalten,
# stehen die Nameserver in /etc/dhpcp/resolv.conf. Gegebenenfalls wird das
# ifdhpcp-done-Skript o.ä. diese automatisch updaten und an
# /etc/dhpcp/hostinfo-$EXTERNAL_INTERFACE oder
# /etc/dhpcp/dhpcpd-$EXTERNAL_INTERFACE.info anhängen.

# Wenn Sie das ifdhpcp-done-Skript aus dem Beispiel verwenden,
# werden die folgenden NAMESERVER-Definitionen (eine pro Server,
```



```
# höchstens drei) korrekt überschrieben.

# Ansonsten sollten Sie, wenn Sie eine statische IP-Adresse
# besitzen, sowohl diese statische IP als auch die IP Ihres
# externen Nameservers hier eintragen.

if [ -f /etc/dhpcp/hostinfo-$EXTERNAL_INTERFACE ]; then
    . /etc/dhpcp/hostinfo-$EXTERNAL_INTERFACE
elif [ -f /etc/dhpcp/dhpcpd-$EXTERNAL_INTERFACE.info ]; then
    . /etc/dhpcp/dhpcpd-$EXTERNAL_INTERFACE.info
else
    NAMESERVER_1="mein.erster.Name.Server"
    IPADDR="meine.eigene.IP.Adresse"
fi

# -----
# TRAGEN SIE HIER DIE ZAHL DER ERLAUBTEN SERVER BZW. VERBINDUNGEN EIN.

# Die von X-Windows benutzten Ports beginnen bei 6000. Jeder weitere
# Server verwendet den nächsthöheren Port, bis zu 6063.

XWINDOW_PORTS="6000"                # (TCP) X Window

# SSH beginnt mit 1023 und belegt für weitere gleichzeitig bestehende
# Verbindungen die nächstniedrigeren Ports, bis zu 513.

SSH_PORTS="1020:1023"                # vier gleichzeitige Verbindungen

# -----

SOCKS_PORT="1080"                    # (TCP) SOCKS
OPENWINDOWS_PORT="2000"              # (TCP) OpenWindows
NFS_PORT="2049"                      # (TCP/UDP) NFS

# -----

# TCP-SYN-Cookies einschalten
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Schutz vor gefälschten IPs:
# Prüfung der Absenderadresse einschalten
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# ICMP-Redirects ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
```

434 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
echo 0 > $f
done

# vom Absender geroutete Pakete ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# Module für Masquerading laden
#/sbin/modprobe ip_masq_ftp.o
#/sbin/modprobe ip_masq_raudio.o
#/sbin/modprobe ip_masq_irc.o
#/sbin/modprobe/ip_masq_vdolive.o
#/sbin/modprobe/ip_masq_cuseeme.o
#/sbin/modprobe/ip_masq_quake.o

# -----

# Vorherbestehende Regeln aus den Chains löschen.
ipfwadm -I -f
ipfwadm -O -f
ipfwadm -F -f

# Voreinstellung: Pakete ablehnen.
ipfwadm -I -p deny
ipfwadm -O -p reject
ipfwadm -F -p deny

# -----
# LOOPBACK

# Keine Einschränkungen für das Loopback-Interface.
ipfwadm -I -a accept -W $LOOPBACK_INTERFACE
ipfwadm -O -a accept -W $LOOPBACK_INTERFACE

# -----
# Verbindungen von problematischen Sites ablehnen.

# /etc/rc.d/rc.firewall.blocked enthält eine Liste gesperrter Adressen
# im Format
# ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S <address/mask>

# Pakete von einem Absender in der Blockadeliste ablehnen.
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

# -----
```

```
# Gefälschte und illegale Adressen ablehnen.
# Gefälschte Pakete sowie völlig illegale Absenderadressen ablehnen.
# Wir wollen auch nicht an fehlerhafte Empfänger senden.

# Gefälschte Pakete von unserer eigenen Adresse ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $IPADDR -o

# Pakete mit einem privaten Klasse-A-Netz in Absender oder Empfänger.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $CLASS_A -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -D $CLASS_A -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -S $CLASS_A -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -D $CLASS_A -o

# Pakete mit einem privaten Klasse-B-Netz in Absender oder Empfänger.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $CLASS_B -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -D $CLASS_B -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -S $CLASS_B -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -D $CLASS_B -o

# Pakete mit einem privaten Klasse-C-Netz in Absender oder Empfänger.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $CLASS_C -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -D $CLASS_C -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -S $CLASS_C -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -D $CLASS_C -o

# Pakete von Loopback ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $LOOPBACK -o

# Fehlerhafte Broadcast-Pakete ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $BROADCAST_DEST -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -D $BROADCAST_SRC -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -S $BROADCAST_DEST -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -D $BROADCAST_SRC -o

# Klasse-D-Multicast-Adressen ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $MULTICAST -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -S $MULTICAST -o

# Reservierte Adressen der Klasse E ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $RESERVED_NET -o
ipfwadm -O -a reject -W $EXTERNAL_INTERFACE -D $RESERVED_NET -o

# Von der IANA reservierte Adressen ablehnen.
# 0.*.*.*, 1.*.*.*, 2.*.*.*, 5.*.*.*, 7.*.*.*, 23.*.*.*, 27.*.*.*
# 31.*.*.*, 37.*.*.*, 39.*.*.*, 41.*.*.*, 42.*.*.*, 58-60.*.*.*
```

436 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 1.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 2.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 5.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 7.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 23.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 27.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 31.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 37.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 39.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 41.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 42.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 58.0.0.0/7 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 60.0.0.0/8 -o
```

65: 01000001 - /3 würde 64 beinhalten - deshalb sind 65-79 explizit angegeben

```
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 65.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 66.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 67.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 68.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 69.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 70.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 71.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 72.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 73.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 74.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 75.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 76.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 77.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 78.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 79.0.0.0/8 -o
```

80: 01010000 - /4 beinhaltet 80-95

```
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 80.0.0.0/4 -o
```

96: 01100000 - /4 beinhaltet 96-111

```
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 96.0.0.0/4 -o
```

126: 01111110 - /3 würde 127 einschließen, deshalb 112-126 explizit

```
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 112.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 113.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 114.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 115.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 116.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 117.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 118.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 119.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 120.0.0.0/8 -o
```

```
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 121.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 122.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 123.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 124.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 125.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 126.0.0.0/8 -o

# 217: 11011001 - /5 würde 216 einschließen, deshalb 217-219 explizit
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 217.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 218.0.0.0/8 -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 219.0.0.0/8 -o

# 223: 11011111 - /6 beinhaltet 220-223
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S 220.0.0.0/6 -o

# -----
# ICMP

# (4) Source_Quench
# Bremsst ankommende und abgehende Pakete (Flusskontrolle)

ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 4 -D $IPADDR

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $IPADDR 4 -D $ANYWHERE

# (12) Parameter_Problem
# Ankommende und abgehende Fehlermeldungen

ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 12 -D $IPADDR

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $IPADDR 12 -D $ANYWHERE

# (3) Dest_Unreachable, Service_Unavailable
# Größenaushandlung, Dienst oder Empfänger nicht verfügbar,
# letzte Antwort auf traceroute

ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 3 -D $IPADDR

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $IPADDR 3 -D $ANYWHERE

# (11) Time_Exceeded
# Ankommender oder abgehender Timeout,
# Antwort einer Zwischenstation auf traceroute
```

438 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 11 -D $IPADDR
```

```
ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $IPADDR 11 -D $MY_ISP
```

```
# Abgehende pings sind immer erlaubt
```

```
ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $IPADDR 8 -D $ANYWHERE
```

```
ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 0 -D $IPADDR
```

```
# Ankommende pings nur von vertrauenswürdigen Hosts.
```

```
ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $MY_ISP 8 -D $IPADDR
```

```
ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
-S $IPADDR 0 -D $MY_ISP
```

```
# -----
# Manche abgehenden Pakete ablehnen - Schutz vor eigenen Fehlern
```

```
# OpenWindows: Verbindungsaufbau
```

```
ipfwadm -O -a reject -P tcp -y -W $EXTERNAL_INTERFACE \
-S $IPADDR \
-D $ANYWHERE $OPENWINDOWS_PORT
```

```
# X Window: Verbindungsaufbau
```

```
ipfwadm -O -a reject -P tcp -y -W $EXTERNAL_INTERFACE \
-S $IPADDR \
-D $ANYWHERE $XWINDOW_PORTS
```

```
# SOCKS: Verbindungsaufbau
```

```
ipfwadm -O -a reject -P tcp -y -W $EXTERNAL_INTERFACE \
-S $IPADDR \
-D $ANYWHERE $SOCKS_PORT
```

```
# -----
# TCP UNPRIVILEGIERTE PORTS
```

```
# NFS: ankommende Verbindung (atypischer TCP-Modus)
```

```
ipfwadm -I -a deny -P tcp -y -W $EXTERNAL_INTERFACE \
-D $IPADDR $NFS_PORT
```

```
# OpenWindows: ankommende Verbindung
ipfwadm -I -a deny -P tcp -y -W $EXTERNAL_INTERFACE \
-D $IPADDR $OPENWINDOWS_PORT

# X Window: ankommende Verbindung
ipfwadm -I -a deny -P tcp -y -W $EXTERNAL_INTERFACE \
-D $IPADDR $XWINDOW_PORTS

# SOCKS: ankommende Verbindung
ipfwadm -I -a deny -P tcp -y -W $EXTERNAL_INTERFACE \
-D $IPADDR $SOCKS_PORT

# -----
# UDP UNPRIVILEGIERTE PORTS

# NFS: ankommende Verbindung (normaler UDP-Modus)
ipfwadm -I -a deny -P udp -W $EXTERNAL_INTERFACE \
-D $IPADDR $NFS_PORT

# -----
# HINWEIS:
# Die symbolischen Portnamen aus /etc/services sind nicht in
# jeder Distribution gleich. Ihre Verwendung ist aber weniger
# fehleranfällig und macht das Skript lesbarer.

# -----
# Notwendige Dienste

# DNS-Client (53)
# -----

ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $NAMESERVER_1 53

ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \
-S $NAMESERVER_1 53 \
-D $IPADDR $UNPRIVPORTS

# TCP Anfragen von einem Client sind vom Protokoll erlaubt,
# wenn eine UDP-Anfrage fehlschlägt. Das kommt selten vor.
# TCP wird meistens von sekundären Nameservern für einen
# Zone-Transfer vom primären Nameserver benutzt, und von
# Hackern.

ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $NAMESERVER_1 53

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
```

440 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
-S $NAMESERVER_1 53 \  
-D $IPADDR $UNPRIVPORTS  
  
# DNS-Server (53)  
# -----  
  
# DNS Forwarding- und Caching-Nameserver  
# -----  
  
# Anfrage vom Server.  
# Ein Server verwendet nur UDP, nicht TCP  
  
ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $IPADDR 53 \  
-D $NAMESERVER_1 53  
  
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $NAMESERVER_1 53 \  
-D $IPADDR 53  
  
# DNS: voller Nameserver  
# -----  
  
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S DNS.Clients 53 $UNPRIVPORTS \  
-D $IPADDR 53  
  
ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $IPADDR 53 \  
-D DNS.Clients 53 $UNPRIVPORTS  
  
# Zone-Transfers  
# Wegen der potenziellen Gefahren eines Zone-Transfers  
# sollten Sie TCP-Pakete nur an ausgewählte sekundäre  
# Server zulassen.  
  
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S sekundärer.Server $UNPRIVPORTS \  
-D $IPADDR 53  
  
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 53 \  
-D sekundärer.Server $UNPRIVPORTS  
  
# -----
```


Anhang B • Firewall-Beispiele und hilfreiche Skripten 441

```
# AUTH-Client (113)
# -----
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $ANYWHERE 113
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 113 \
-D $IPADDR $UNPRIVPORTS
```

```
# AUTH-Server (113)
# -----
```

```
# Eingehende Anfragen ablehnen
```

```
ipfwadm -I -a reject -P tcp -W $EXTERNAL_INTERFACE \
-D $IPADDR 113
```

```
# ODER
```

```
# Eingehende Anfragen annehmen
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE $UNPRIVPORTS \
-D $IPADDR 113
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $IPADDR 113 \
-D $ANYWHERE $UNPRIVPORTS
```

```
# -----
# TCP-Dienste auf ausgewählten Ports
```

```
# SMTP-Client (25)
# -----
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $ANYWHERE 25
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 25 \
-D $IPADDR $UNPRIVPORTS
```

442 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
# -----
# ODER SMTP-Client sendet ohne lokalen Server an einen Provider-Computer

ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $SMTP_SERVER 25

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $SMTP_SERVER 25 \
-D $IPADDR $UNPRIVPORTS

# SMTP-Server (25)
# -----

ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE $UNPRIVPORTS \
-D $IPADDR 25

ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $IPADDR 25 \
-D $ANYWHERE $UNPRIVPORTS

# -----

# POP-Client (110)
# -----

ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $POP_SERVER 110

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $POP_SERVER 110 \
-D $IPADDR $UNPRIVPORTS

# POP-Server (110)
# -----

ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S Ips.meiner.POP.Clients $UNPRIVPORTS \
-D $IPADDR 110

ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $IPADDR 110 \
-D Ips.meiner.POP.Clients $UNPRIVPORTS

# -----
```

IMAP-Client (143)

```
ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D IP.meines.IMAP.Servers 143
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S IP.meines.IMAP.Servers 143 \
-D $IPADDR $UNPRIVPORTS
```

IMAP-Server (143)

```
ipfwadm -I -a accept -P tcp      -W $EXTERNAL_INTERFACE \
-S IPs.meiner.IMAP.Clients $UNPRIVPORTS \
-D $IPADDR 143
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $IPADDR 143 \
-D IPs.meiner.IMAP.Clients $UNPRIVPORTS
```

NNTP NEWS-Client (119)

```
ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $NEWS_SERVER 119
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $NEWS_SERVER 119 \
-D $IPADDR $UNPRIVPORTS
```

TELNET-Client (23)

```
ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $ANYWHERE 23
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 23 \
-D $IPADDR $UNPRIVPORTS
```

444 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

TELNET-Server (23)

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR 23
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 23 \  
-D $ANYWHERE $UNPRIVPORTS
```

SSH-Client (22)

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 22
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 22 \  
-D $IPADDR $UNPRIVPORTS
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $SSH_PORTS \  
-D $ANYWHERE 22
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 22 \  
-D $IPADDR $SSH_PORTS
```

SSH-Server (22)

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR 22
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 22 \  
-D $ANYWHERE $UNPRIVPORTS
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $SSH_PORTS \  
-D $IPADDR 22
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 22 \  
-D $ANYWHERE $SSH_PORTS
```

```
# -----
```

```
# FTP-Client (20, 21)
```

```
# -----
```

```
# Abgehende Verbindung
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 21
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 21 \  
-D $IPADDR $UNPRIVPORTS
```

```
# Datenkanalaufbau im aktiven Modus
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 20 \  
-D $IPADDR $UNPRIVPORTS
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 20
```

```
# Datenkanalaufbau im passiven Modus
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE $UNPRIVPORTS
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR $UNPRIVPORTS
```

```
# FTP-Server (20, 21)
```

```
# -----
```

```
# Ankommende Verbindung
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR 21
```

446 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 21 \  
-D $ANYWHERE $UNPRIVPORTS
```

```
# Datenkanalaufbau im aktiven Modus
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR 20 \  
-D $ANYWHERE $UNPRIVPORTS
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR 20
```

```
# Datenkanalaufbau im passiven Modus
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR $UNPRIVPORTS
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE $UNPRIVPORTS
```

```
# -----
```

```
# HTTP-Client (80)
```

```
# -----
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 80
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 80 \  
-D $IPADDR $UNPRIVPORTS
```

```
# HTTP-Server (80)
```

```
# -----
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR 80
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 80 \  
-D $ANYWHERE $UNPRIVPORTS
```

Anhang B • Firewall-Beispiele und hilfreiche Skripten **447**

```
# -----

# HTTPS-Client (443)
# -----

ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $ANYWHERE 443

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 443 \
-D $IPADDR $UNPRIVPORTS

# HTTPS-Server (443)
# -----

ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $ANYWHERE $UNPRIVPORTS \
-D $IPADDR 443

ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $IPADDR 443 \
-D $ANYWHERE $UNPRIVPORTS

# -----

# HTTP Proxy-Client (8008/8080)
# -----

ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $WEB_PROXY_SERVER $WEB_PROXY_PORT

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $WEB_PROXY_SERVER $WEB_PROXY_PORT \
-D $IPADDR $UNPRIVPORTS

# -----

# FINGER-Client (79)
# -----

ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $ANYWHERE 79
```

448 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 79 \  
-D $IPADDR $UNPRIVPORTS
```

```
# FINGER-Server (79)  
# -----
```

```
ipfwadm -I -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S Ips.meiner.finger.Clients $UNPRIVPORTS \  
-D $IPADDR 79
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $IPADDR 79 \  
-D Ips.meiner.finger.Clients $UNPRIVPORTS
```

```
# -----
```

```
# WHOIS-Client (43)  
# -----
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 43
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 43 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# Gopher-Client (70)  
# -----
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 70
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 70 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# WAIS-Client (210)  
# -----
```



```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \
-S $IPADDR $UNPRIVPORTS \
-D $ANYWHERE 210

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
-S $ANYWHERE 210 \
-D $IPADDR $UNPRIVPORTS

# -----

# TRACEROUTE
# traceroute verwendet meistens -S 32769:65535 -D 33434:33523
# -----

# Abgehende traceroutes
# -----

ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \
-S $IPADDR 32769:65535 \
-D $ANYWHERE 33434:33523

# Ankommende traceroute vom Internet-Provider.
# Alle anderen werden abgelehnt.
# -----

ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \
-S $MY_ISP 32769:65535 \
-D $IPADDR 33434:33523

# -----
# UDP nur auf ausgewählten Ports annehmen

# DHCP-Client (67, 68)
# -----

# INIT oder REBINDING: Keine IP oder Gültigkeit der IP abgelaufen

ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \
-S $BROADCAST_SRC 68 \
-D $BROADCAST_DEST 67

# Eine neue IP holen

ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \
-S $BROADCAST_SRC 67 \
-D $BROADCAST_DEST 68
```

450 — Die rc.firewall für ipfwadm für ein Einzelplatzsystem oder ein Privat-LAN aus Kapitel 3

```
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $DHCP_SERVER 67 \  
-D $BROADCAST_DEST 68
```

```
ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $BROADCAST_SRC 68 \  
-D $DHCP_SERVER 67
```

```
# Im Ergebnis ändern wir unsere IP-Adresse. Das Paket geht  
# an unsere neue Adresse, bevor der DHCP-Client das Update  
# erhalten hat.
```

```
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $DHCP_SERVER 67 \  
-D $MY_ISP 68
```

```
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $DHCP_SERVER 67 \  
-D $IPADDR 68
```

```
ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $IPADDR 68 \  
-D $DHCP_SERVER 67
```

```
# -----
```

```
# NTP (123) - Zugriff auf fremde Network Time Server  
# -----
```

```
ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D mein.Zeit.Server 123
```

```
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S mein.Zeit.Server 123 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# Keine lokalen Einschränkungen.
```

```
# Alle internen Computer dürfen auf die Firewall zugreifen.
```

```
ipfwadm -I -a accept -W $LAN_INTERFACE_1 \  
-S $LAN_1
```

```
ipfwadm -O -a accept -W $LAN_INTERFACE_1 \  
-D $LAN_1
```

```
# -----  
# Interne Adressen durch Masquerading verbergen.  
  
# Alle internen Pakete werden durch Masquerading geschickt.  
  
ipfwadm -F -a masquerade -W $EXTERNAL_INTERFACE -S $LAN_1  
  
# -----  
  
echo "fertig"  
  
exit 0
```

B.3 Optimierung der Firewall-Regeln

Die Optimierung der Firewall ist für ein Privatsystem kein so wichtiges Thema. Vermutlich kann der Netzwerkcode von Linux die Pakete viel schneller bearbeiten, als das Netzwerk sie liefert. Nachdem die Semantik der Firewall-Regeln von der Reihenfolge abhängt und eine Firewall auch nicht ganz einfach zu konstruieren ist, halte ich es für vorteilhafter, auf die Lesbarkeit als auf die Geschwindigkeit zu achten.

Firewall-Regeln müssen vom Speziellen zum Allgemeinen hin angeordnet werden. Innerhalb dieser Einschränkungen kann man die Reihenfolge der Regeln etwas optimieren, indem man die Regeln für die häufigsten Pakete an den Anfang stellt. Wenn Sie z.B. nur selten ftp einsetzen, werden Web-Pakete viel häufiger auftreten als ftp-Pakete. Die Liste der Firewallregeln wird nur solange durchsucht, wie keine passende Regel gefunden wurde. Indem Sie also die Regeln für http vor die für ftp stellen, verringern Sie die Rechenzeit für die Suche nach einer http-Regel.

Darüber hinaus lassen sich die Regeln auch dadurch optimieren, dass man die input- und output-Regeln nicht beieinander hält, sondern sie jeweils relativ zu den anderen Regeln in der betreffenden Kette anordnet. Z.B. schickt ein Web-Browser viel weniger Daten zum Web-Server, als er umgekehrt von diesem empfängt. Für ftp gilt das Gleiche: Die Kontrolldaten an Port 21 nehmen im Vergleich zu den Datenpaketen von Port 20 einen verschwindend geringen Stellenwert ein.

Das folgende Beispiel geht von einem privaten Linux-Computer mit einer Internet-Anbindung aus. Die Maschine verwendet die Firewall aus Kapitel 3. Sie besitzt eine permanente, statische IP-Adresse. Die Firewall erledigt für ein kleines, privates LAN das IP-Masquerading.

Ein lokaler DNS-Server leitet Anfragen zunächst an den Nameserver des Internet-Providers weiter, bevor er gegebenenfalls einen weiteren fremden Nameserver befragt, falls das System des Providers nicht antwortet. Lokal ist ein vollwertiger SMTP-Dienst im Einsatz, aber die Firewall unterstützt für den Mailempfang

zusätzlich eine POP-Anbindung zu einem Rechner des Providers. Der Computer verfügt über einen eigenen Webserver, aber nicht über einen ftp-Server.

Das zugehörige Firewall-Skript sähe etwa so aus:

```
#!/bin/sh

echo "Initialisiere die Firewall ..."

# Ein paar Definitionen, die uns die Arbeit erleichtern

# -----
# PASSEN SIE DIESEN ABSCHNITT AN IHR SYSTEM UND IHREN PROVIDER AN

EXTERNAL_INTERFACE="eth0"           # Interface zum Internet
INTERNAL_INTERFACE="eth1"           # internes LAN-Interface
LOOPBACK_INTERFACE="lo"             # Loopback

IPADDR="my.ip.address"              # Ihre IP-Adresse
LAN_IPADDR="192.168.1.1"             # Adresse des internen Interfaces
LAN_ADDRESSES="192.168.1.0/24"       # privater Adressbereich

ANYWHERE="any/0"                    # passt auf jede IP

MY_ISP="Adressbereich.meines.ISPs"  # Adressbereich von ISP und NOC
NAMESERVER_1="erster.Name.Server"   # jeder braucht wenigstens einen

SMTP_SERVER="any/0"                 # externer Mail-Server
SMTP_GATEWAY="Server.des.ISPs"      # externes Mail-Relay
POP_SERVER="ext.POP.Server"          # externer POP-Server, falls vorhanden
NEWS_SERVER="ext.News.Server"        # externer Newsserver, falls vorhanden
WEB_PROXY_SERVER="mein.WWW.Proxy"    # Web-Proxy des Providers, falls vorhanden
WEB_PROXY_PORT="WWW.Proxy.Port"      # Port des Proxies, falls vorhanden
                                      # meistens 8008 oder 8080

LOOPBACK="127.0.0.0/8"               # reservierter Loopback-Bereich
CLASS_A="10.0.0.0/8"                 # private Netze der Klasse A
CLASS_B="172.16.0.0/12"               # private Netze der Klasse B
CLASS_C="192.168.0.0/16"              # private Netze der Klasse C
CLASS_D_MULTICAST="224.0.0.0/4"       # Multicast-Adressen der Klasse D
BROADCAST_SRC="0.0.0.0"               # Broadcast-Absender
BROADCAST_DEST="255.255.255.255"      # Broadcast-Empfänger
PRIVPORTS="0:1023"                   # privilegierter Portbereich
UNPRIVPORTS="1024:65535"              # unprivilegierter Portbereich
TRACEROUTE_SRC_PORTS="32769:65535"    # 
TRACEROUTE_DEST_PORTS="33434:33523"   # 
SOCKS_PORT="1080"                     # (TCP) SOCKS
OPENWINDOWS_PORT="2000"               # (TCP) OpenWindows
NFS_PORT="2049"                       # (TCP/UDP) NFS
```

```
# .....
# TRAGEN SIE HIER DIE ZAHL DER ERLAUBTEN SERVER BZW. VERBINDUNGEN EIN.

XWINDOW_PORTS="6000"                # (TCP) X Window

SSH_PORTS="1020:1023"                # vier gleichzeitige Verbindungen

# -----

# TCP-SYN-Cookies einschalten
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Schutz vor gefälschten IPs:
# Prüfung der Absenderadresse einschalten
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# ICMP-Redirects ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# vom Absender geroutete Pakete ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# -----

# Vorherbestehende Regeln aus den Chains löschen.
ipchains -F

# Voreinstellung: Pakete ablehnen.
ipchains -P input    DENY
ipchains -P output   REJECT
ipchains -P forward  REJECT

# -----
# LOOPBACK

# Keine Einschränkungen für das Loopback-Interface.
ipchains -A input  -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT

# -----
# Verbindungen von problematischen Sites ablehnen.
```

```
# /etc/rc.d/rc.firewall.blocked enthält eine Liste gesperrter Adressen
# im Format
# ipchains -A input -i $EXTERNAL_INTERFACE -s <address/mask> -j DENY

# Pakete von einem Absender in der Blockadeliste ablehnen.
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

# -----
# Gefälschte und illegale Adressen ablehnen.
# Gefälschte Pakete sowie völlig illegale Absenderadressen ablehnen.
# Wir wollen auch nicht an fehlerhafte Empfänger senden.

# Gefälschte Pakete von unserer eigenen Adresse ablehnen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $IPADDR -j DENY -1

# Pakete mit einem privaten Klasse-A-Netz in Absender
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_A -j DENY

# Pakete mit einem privaten Klasse-B-Netz in Absender
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_B -j DENY

# Pakete mit einem privaten Klasse-C-Netz in Absender
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_C -j DENY

# Pakete von Loopback ablehnen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $LOOPBACK -j DENY

# Fehlerhafte Broadcast-Pakete ablehnen.
ipchains -A input -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j DENY -1
ipchains -A input -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC -j DENY -1

# Klasse-D-Multicast-Adressen ablehnen.
# Multicast ist nur als Absender illegal; es benutzt UDP.
ipchains -A input -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST \
    -j DENY -1

# -----
# ICMP

# (4) Source_Quench
#     Bremsst ankommende und abgehende Pakete (Flusskontrolle)

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
    -s $ANYWHERE 4 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
    -s $IPADDR 4 -d $ANYWHERE -j ACCEPT
```

```
# (12) Parameter_Problem
#     Ankommende und abgehende Fehlermeldungen

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 12 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 12 -d $ANYWHERE -j ACCEPT

# (3) Dest_Unreachable, Service_Unavailable
#     Größenaushandlung, Dienst oder Empfänger nicht verfügbar,
#     letzte Antwort auf traceroute

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 3 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 3 -d $MY_ISP -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR fragmentation-needed -d $ANYWHERE -j ACCEPT

# (11) Time_Exceeded
#     Ankommender oder abgehender Timeout,
#     Antwort einer Zwischenstation auf traceroute

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 11 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 11 -d $MY_ISP -j ACCEPT

# Abgehende pings sind immer erlaubt

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 8 -d $ANYWHERE -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $ANYWHERE 0 -d $IPADDR -j ACCEPT

# Ankommende pings nur von vertrauenswürdigen Hosts.

ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -s $MY_ISP 8 -d $IPADDR -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR 0 -d $MY_ISP -j ACCEPT
```

```
# -----

# UNPRIVILEGIERTE PORTS
# Manche Ports würden Protokoll- oder Administrationsschwierigkeiten beschwören

# OpenWindows: ankommende Verbindung
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \
        -d $IPADDR $OPENWINDOWS_PORT -j DENY

# X Window: ankommende Verbindung
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \
        -d $IPADDR $XWINDOW_PORTS -j DENY -l

# SOCKS: ankommende Verbindung
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \
        -d $IPADDR $SOCKS_PORT -j DENY

# NFS: TCP-Verbindungen
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp -y \
        -d $IPADDR $NFS_PORT -j DENY -l

# NFS: UDP-Verbindungen
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR $NFS_PORT -j DENY -l

# -----

# DNS-Client (53)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
        -s $IPADDR 53 \
        -d $NAMESERVER_1 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -s $NAMESERVER_1 53 \
        -d $IPADDR 53 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
        -s $IPADDR $UNPRIVPORTS \
        -d $NAMESERVER_1 53 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -s $NAMESERVER_1 53 \
        -d $IPADDR $UNPRIVPORTS -j ACCEPT

# TCP-Anfragen von einem Client sind vom Protokoll erlaubt,
```


Anhang B • Firewall-Beispiele und hilfreiche Skripten **457**

```
# wenn eine UDP-Anfrage fehlschlägt. Das kommt selten vor.
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $NAMESERVER_1 53 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $NAMESERVER_1 53 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# AUTH (113)  
# -----
```

```
# Eingehende Anfragen annehmen
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR 113 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $IPADDR 113 \  
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
# Abgehende AUTH-Anfragen dürfen passieren
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 113 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 113 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# HTTP-Client (80)  
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 80 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 80 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# HTTP-Server (80)
# -----
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 80 -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 80 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# SSL-Client (443)
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 443 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 443 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# NNTP (119) - News als Usenet-Client lesen und schreiben
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $NEWS_SERVER 119 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $NEWS_SERVER 119 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# POP (110) - Mailempfang als POP-Client
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $POP_SERVER 110 -j ACCEPT
```

```

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $POP_SERVER 110 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# -----

# SMTP-Client (25)
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 25 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 25 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# SMTP-Server (25)
# -----

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE $UNPRIVPORTS \
-d $IPADDR 25 -j ACCEPT

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR 25 \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT

# -----

# TELNET (23) - abgehende Verbindungen zu fremden Sites
# -----

ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 23 -j ACCEPT

ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 23 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT

# -----

# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern
# -----

```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 22 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $SSH_PORTS \
-d $ANYWHERE 22 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 22 \
-d $IPADDR $SSH_PORTS -j ACCEPT
```

```
# -----
```

```
# FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern
```

```
# -----
```

```
# abgehende Verbindung
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 21 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $ANYWHERE 21 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# Datenkanalaufbau im aktiven Modus
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
-s $ANYWHERE 20 \
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE 20 -j ACCEPT
```

```
# Datenkanalaufbau im passiven Modus
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
-s $IPADDR $UNPRIVPORTS \
-d $ANYWHERE $UNPRIVPORTS -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE $UNPRIVPORTS \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# WHOIS-Client (43)
```

```
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \  
-s $IPADDR $UNPRIVPORTS \  
-d $ANYWHERE 43 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \  
-s $ANYWHERE 43 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----
```

```
# TRACEROUTE
```

```
# -----
```

```
# Abgehende traceroutes
```

```
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR $TRACEROUTE_SRC_PORTS \  
-d $ANYWHERE $TRACEROUTE_DEST_PORTS -j ACCEPT
```

```
# Ankommende traceroutes vom Internet-Provider.
```

```
# -----
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s $MY_ISP $TRACEROUTE_SRC_PORTS \  
-d $IPADDR $TRACEROUTE_DEST_PORTS -j ACCEPT
```

```
# -----
```

```
# NTP (123) - Zugriff auf fremde Network Time Server
```

```
# -----
```

```
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \  
-s $IPADDR $UNPRIVPORTS \  
-d mein.Zeit.Server 123 -j ACCEPT
```

```
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \  
-s mein.Zeit.Server 123 \  
-d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
# -----  
  
echo "fertig"  
  
exit 0
```

B.3.1 Optimierung der Reihenfolge der ipfwadm-Regeln

Neben der allgemeinen Änderung der Regelreihenfolge unterstützt `ipfwadm` noch einen zusätzlichen Mechanismus zur Optimierung: Portnummern und ICMP-Nachrichtentypen lassen sich als Listen von bis zu zehn Werten in einer Regel angeben. Ein Wertebereich zählt dabei als zwei Werte. Innerhalb jeder Absender- oder Empfängerliste darf ein Wertebereich definiert sein. Mehrere Regeln lassen sich so zu einer einzigen Regel zusammenfassen, indem man die Ports oder ICMP-Nachrichtentypen zu einer Liste kombiniert. Ein Paket muss damit nur noch mit einer einzigen Regel verglichen werden, nicht mehr mit allen Einzelregeln.

Es folgt das vorherige Firewall-Skript, optimiert für `ipfwadm`:

```
#!/bin/sh  
  
echo "Initialisiere die Firewall ... "  
  
# Ein paar Definitionen, die uns die Arbeit erleichtern  
  
# -----  
# PASSEN SIE DIESEN ABSCHNITT AN IHR SYSTEM UND IHREN PROVIDER AN  
  
EXTERNAL_INTERFACE="eth0"           # Interface zum Internet  
INTERNAL_INTERFACE="eth1"           # internes LAN-Interface  
LOOPBACK_INTERFACE="lo"             # Loopback  
  
IPADDR="my.ip.address"              # Ihre IP-Adresse  
LAN_IPADDR="192.168.1.1"             # Adresse des internen Interfaces  
LAN_ADDRESSES="192.168.1.0/24"       # privater Adressbereich  
  
ANYWHERE="any/0"                    # passt auf jede IP  
  
MY_ISP="Adressbereich.meines.ISPs"   # Adressbereich von ISP und NOC  
NAMESERVER_1="erster.Name.Server"    # jeder braucht wenigstens einen  
  
SMTP_SERVER="any/0"                 # externer Mail-Server  
SMTP_GATEWAY="Server.des.ISPs"      # externes Mail-Relay  
POP_SERVER="ext.POP.Server"         # externer POP-Server, falls vorhanden  
NEWS_SERVER="ext.News.Server"       # externer Newsserver, falls vorhanden  
WEB_PROXY_SERVER="mein.WWW.Proxy"    # Web-Proxy des Providers, falls vorhanden
```

```

WEB_PROXY_PORT="WWW.Proxy.Port"      # Port des Proxies, falls vorhanden
                                      # meistens 8008 oder 8080

LOOPBACK="127.0.0.0/8"                # reservierter Loopback-Bereich
CLASS_A="10.0.0.0/8"                  # private Netze der Klasse A
CLASS_B="172.16.0.0/12"                # private Netze der Klasse B
CLASS_C="192.168.0.0/16"              # private Netze der Klasse C
CLASS_D_MULTICAST="224.0.0.0/4"       # Multicast-Adressen der Klasse D
BROADCAST_SRC="0.0.0.0"                # Broadcast-Absender
BROADCAST_DEST="255.255.255.255"      # Broadcast-Empfänger
PRIVPORTS="0:1023"                    # privilegierter Portbereich
UNPRIVPORTS="1024:65535"               # unprivilegierter Portbereich
TRACEROUTE_SRC_PORTS="32769:65535"
TRACEROUTE_DEST_PORTS="33434:33523"
SOCKS_PORT="1080"                     # (TCP) SOCKS
OPENWINDOWS_PORT="2000"               # (TCP) OpenWindows
NFS_PORT="2049"                       # (TCP/UDP) NFS

# .....
# TRAGEN SIE HIER DIE ZAHL DER ERLAUBTEN SERVER BZW. VERBINDUNGEN EIN.

XWINDOW_PORTS="6000"                  # (TCP) X Window

SSH_PORTS="1020:1023"                 # vier gleichzeitige Verbindungen

# -----

# TCP-SYN-Cookies einschalten
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Schutz vor gefälschten IPs:
# Prüfung der Absenderadresse einschalten
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# ICMP-Redirects ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# vom Absender geroutete Pakete ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# -----

```

```
# Vorherbestehende Regeln aus den Chains löschen.
ipfwadm -I -f
ipfwadm -O -f
ipfwadm -F -f

# Voreinstellung: Pakete ablehnen.
ipfwadm -I -p deny
ipfwadm -O -p reject
ipfwadm -F -p reject

# -----
# LOOPBACK

# Keine Einschränkungen für das Loopback-Interface.
ipfwadm -I -a accept -W $LOOPBACK_INTERFACE
ipfwadm -O -a accept -W $LOOPBACK_INTERFACE

# -----
# Verbindungen von problematischen Sites ablehnen.

# /etc/rc.d/rc.firewall.blocked enthält eine Liste gesperrter Adressen
# im Format
# ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S <address/mask>

# Pakete von einem Absender in der Blockadeliste ablehnen.
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

# -----
# Gefälschte und illegale Adressen ablehnen
# Gefälschte Pakete sowie völlig illegale Absenderadressen ablehnen.
# Wir wollen auch nicht an fehlerhafte Empfänger senden.

# Gefälschte Pakete von unserer eigenen Adresse ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $IPADDR -o

# Pakete mit einem privaten Klasse-A-Netz in Absender
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $CLASS_A -o

# Pakete mit einem privaten Klasse-B-Netz in Absender
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $CLASS_B -o

# Pakete mit einem privaten Klasse-C-Netz in Absender
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $CLASS_C -o

# Pakete von Loopback ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $LOOPBACK -o
```



```
# Fehlerhafte Broadcast-Pakete ablehnen.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $BROADCAST_DEST -o
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -D $BROADCAST_SRC -o

# Klasse-D-Multicast-Adressen ablehnen.
# Multicast ist nur als Absender illegal; es benutzt UDP.
ipfwadm -I -a deny -W $EXTERNAL_INTERFACE -S $MULTICAST -o

# -----
# ICMP

# (0) Echo_Reply ("pong")
# (3) Dest_Unreachable, Service_Unavailable
# Größenaushandlung, Dienst oder Empfänger nicht verfügbar,
# letzte Antwort auf traceroute
# (4) Source_Quench
# Bremsst ankommende und abgehende Pakete (Flusskontrolle)
# (8) Echo_Request ("ping")
# (12) Parameter_Problem

ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $ANYWHERE 0 3 4 11 12 -D $IPADDR

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $IPADDR 3 4 8 12 -D $ANYWHERE

ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $MY_ISP 8 -D $IPADDR

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $IPADDR 0 11 -D $MY_ISP

# -----

# DNS: UDP-Forwards (53)
# -----

ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \
        -S $IPADDR 53 \
        -D $NAMESERVER_1 53

ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \
        -S $NAMESERVER_1 53 \
        -D $IPADDR 53

# DNS: UDP-Client (53)
# -----
```

```
ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 53
```

```
ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 53 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# HTTP-Client (80)  
# SSL-Client (443)  
# SMTP - Mailversand (25)  
# AUTH-Client (113)  
# WHOIS-Client(43)  
# Ankommende Server-Antworten  
# -----
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 80 443 25 113 43 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# NNTP (119) - News als Usenet-Client lesen und schreiben  
# -----
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $NEWS_SERVER 119 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# POP (110) - Mailempfang als POP-Client  
# -----
```

```
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $POP_SERVER 110 \  
-D $IPADDR $UNPRIVPORTS
```

```
# -----
```

```
# HTTP-Client (80)  
# SSL-Client (443)  
# SMTP - Mailversand (25)  
# AUTH-Client (113)  
# WHOIS-Client(43)
```

```
# Abgehende Client-Anfragen
# -----
```

```
ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D $ANYWHERE 80 443 25 113 43
```

```
# -----
```

```
# NNTP (119) - News als Usenet-Client lesen und schreiben
# -----
```

```
ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D $NEWS_SERVER 119
```

```
# -----
```

```
# POP (110) - Mailempfang als POP-Client
# -----
```

```
ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D $POP_SERVER 110
```

```
# -----
```

```
# Web-Server (80)
# AUTH-Server (113)
# SMTP - Mailempfang (25)
# Ankommende Client-Anfragen
```

```
ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
        -S $IPADDR 80 113 25 \
        -D $ANYWHERE $UNPRIVPORTS
```

```
ipfwadm -I -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $ANYWHERE $UNPRIVPORTS \
        -D $IPADDR 80 113 25
```

```
# -----
```

```
# TELNET (23) - abgehende Verbindungen zu fremden Sites
# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern
# FTP-Client (21) - abgehende Kontrollverbindungen zu fremden Servern
# -----
```

```
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $UNPRIVPORTS \  
-D $ANYWHERE 23 22 21  
  
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 23 22 21 \  
-D $IPADDR $UNPRIVPORTS  
  
# -----  
# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern  
# -----  
  
ipfwadm -O -a accept -P tcp -W $EXTERNAL_INTERFACE \  
-S $IPADDR $SSH_PORTS \  
-D $ANYWHERE 22  
  
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE 22 \  
-D $IPADDR $SSH_PORTS  
  
# -----  
# UNPRIVILEGIERTE PORTS  
# Manche Ports würden Protokoll- oder Administrationsschwierigkeiten beschwören  
  
# X Window: ankommende Verbindung  
# SOCKS: ankommende Verbindung  
# NFS: ankommende Verbindung (atypischer TCP-Modus)  
  
ipfwadm -I -a deny -P tcp -y -W $EXTERNAL_INTERFACE \  
-D $IPADDR $XWINDOW_PORTS $SOCKS_PORT $NFS_PORT  
  
# NFS: ankommende Verbindung (normaler UDP-Modus)  
ipfwadm -I -a deny -P udp -W $EXTERNAL_INTERFACE \  
-D $IPADDR $NFS_PORT  
  
# -----  
# FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern  
# -----  
  
# Datenkanalaufbau im passiven Modus  
  
ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \  
-S $ANYWHERE $UNPRIVPORTS \  
-D $IPADDR $UNPRIVPORTS
```

```
# Datenkanalaufbau im aktiven Modus

ipfwadm -I -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $ANYWHERE 20 \
        -D $IPADDR $UNPRIVPORTS

# -----

# FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern
# -----

# Datenkanalaufbau im passiven Modus

ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D $ANYWHERE $UNPRIVPORTS

# Datenkanalaufbau im aktiven Modus

ipfwadm -O -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D $ANYWHERE 20

# -----

# NTP (123) - Zugriff auf fremde Network Time Server
# -----

ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D mein.Zeit.Server 123

ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \
        -S mein.Zeit.Server 123 \
        -D $IPADDR $UNPRIVPORTS

# -----

# TRACEROUTE
# -----

# Abgehende traceroutes
# -----

ipfwadm -O -a accept -P udp -W $EXTERNAL_INTERFACE \
        -S $IPADDR $TRACEROUTE_SRC_PORTS \
        -D $ANYWHERE $TRACEROUTE_DEST_PORTS
```

470 ————— *Optimierung der Firewall-Regeln*

```
# Ankommende traceroutes vom Internet-Provider.
# -----

ipfwadm -I -a accept -P udp -W $EXTERNAL_INTERFACE \
        -S $MY_ISP $TRACEROUTE_SRC_PORTS \
        -D $IPADDR $TRACEROUTE_DEST_PORTS

# (11) Time_Exceeded
#      Abgehender Timeout

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $IPADDR 11 -D $MY_ISP

# -----

# PING
# ----

# Ankommende pings nur von vertrauenswürdigen Hosts.

ipfwadm -I -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $MY_ISP 8 -D $IPADDR

ipfwadm -O -a accept -P icmp -W $EXTERNAL_INTERFACE \
        -S $IPADDR 0 -D $MY_ISP

# -----

# DNS: TCP-Client (53)
# -----

# TCP-Anfragen von einem Client sind vom Protokoll erlaubt,
# wenn eine UDP-Anfrage fehlschlägt. Das kommt selten vor.

ipfwadm -I -a accept -P tcp -k -W $EXTERNAL_INTERFACE \
        -S $ANYWHERE 53 \
        -D $IPADDR $UNPRIVPORTS

ipfwadm -O -a accept -P tcp      -W $EXTERNAL_INTERFACE \
        -S $IPADDR $UNPRIVPORTS \
        -D $ANYWHERE 53

# -----

echo "fertig"

exit 0
```

B.3.2 Optimierung der ipchains-Regeln mit benutzerdefinierten Chains

Bei `ipchains` bietet sich neben der bereits besprochenen Umstellung der Regelreihenfolge noch eine weitere Möglichkeit der Optimierung: benutzerdefinierte Chains. Indem Sie ein Paket in Abhängigkeit von den Werten aus dem Paket-Header an eine andere Chain weitergeben, können Sie eine Untermenge der `input`-, `output`- oder `forward`-Regeln darauf anwenden und müssen nicht immer jedes Paket mit jeder Regel vergleichen, bis eine passende Regel gefunden ist.

Beispielsweise muss ein ankommendes Paket von einem NTP-Server bei dem nicht-optimierten Firewall-Skript mit ungefähr 40 Regeln verglichen werden, bevor es seine `ACCEPT`-Regel erreicht. Wenn die Firewall über benutzerdefinierte Chains optimiert ist, reichen nur noch 15 Regeln aus.

In `ipchains` bestimmen Regeln nicht nur über die Bedingungen, unter denen ein Paket angenommen oder abgelehnt wird, sondern auch darüber, wann es an eine andere Chain weitergegeben wird. Wenn ein Paket auf keine Regel einer benutzerdefinierten Chain passt, fällt es in die aufrufende Chain zurück. Wenn ein Paket nicht auf eine Regel passt, die eine Chain auswählt, wird es auch nicht an diese Chain übergeben, sondern der Firewall-Code geht sofort zur nächsten Regel über.

Es folgt das obige Firewall-Skript in einer Fassung, die für `ipchains` optimiert ist.

```
#!/bin/sh

echo "Initialisiere die Firewall ..."

# Ein paar Definitionen, die uns die Arbeit erleichtern
# -----
# PASSEN SIE DIESEN ABSCHNITT AN IHR SYSTEM UND IHREN PROVIDER AN

EXTERNAL_INTERFACE="eth0"          # Interface zum Internet
INTERNAL_INTERFACE="eth1"          # internes LAN-Interface
LOOPBACK_INTERFACE="lo"            # Loopback

IPADDR="my.ip.address"             # Ihre IP-Adresse
LAN_IPADDR="192.168.1.1"           # Adresse des internen Interfaces
LAN_ADDRESSES="192.168.1.0/24"     # privater Adressbereich

ANYWHERE="any/0"                   # passt auf jede IP

MY_ISP="Adressbereich.meines.ISPs" # Adressbereich von ISP und NOC
NAMESERVER_1="erster.Name.Server"  # jeder braucht wenigstens einen

SMTP_SERVER="any/0"                # externer Mail-Server
```

```
SMTP_GATEWAY="Server.des.ISPs"      # externes Mail-Relay
POP_SERVER="ext.POP.Server"         # externer POP-Server, falls vorhanden
NEWS_SERVER="ext.News.Server"       # externer Newsserver, falls vorhanden
WEB_PROXY_SERVER="mein.WWW.Proxy"   # Web-Proxy des Providers, falls vorhanden
WEB_PROXY_PORT="WWW.Proxy.Port"     # Port des Proxies, falls vorhanden
                                     # meistens 8008 oder 8080

LOOPBACK="127.0.0.0/8"              # reservierter Loopback-Bereich
CLASS_A="10.0.0.0/8"                # private Netze der Klasse A
CLASS_B="172.16.0.0/12"             # private Netze der Klasse B
CLASS_C="192.168.0.0/16"            # private Netze der Klasse C
CLASS_D_MULTICAST="224.0.0.0/4"     # Multicast-Adressen der Klasse D
BROADCAST_SRC="0.0.0.0"              # Broadcast-Absender
BROADCAST_DEST="255.255.255.255"    # Broadcast-Empfänger
PRIVPORTS="0:1023"                  # privilegierter Portbereich
UNPRIVPORTS="1024:65535"            # unprivilegierter Portbereich
TRACEROUTE_SRC_PORTS="32769:65535"  #
TRACEROUTE_DEST_PORTS="33434:33523" #
SOCKS_PORT="1080"                   # (TCP) SOCKS
OPENWINDOWS_PORT="2000"             # (TCP) OpenWindows
NFS_PORT="2049"                     # (TCP/UDP) NFS

# -----
# TRAGEN SIE HIER DIE ZAHL DER ERLAUTEN SERVER BZW. VERBINDUNGEN EIN.

XWINDOW_PORTS="6000"                # (TCP) X Window

SSH_PORTS="1020:1023"               # vier gleichzeitige Verbindungen

# -----

# TCP-SYN-Cookies einschalten
echo 1 >/proc/sys/net/ipv4/tcp_syncookies

# Schutz vor gefälschten IPs:
# Prüfung der Absenderadresse einschalten
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# ICMP-Redirects ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# vom Absender geroutete Pakete ablehnen
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
```



```
    echo 0 > $f
done
```

```
# -----
```

```
# Vorherbestehende Regeln aus den Chains löschen.
```

```
ipchains -F
ipchains -F spoofed
ipchains -F tcp-c-o
ipchains -F tcp-s-i
ipchains -F udp-c-o
ipchains -F udp-s-i
ipchains -F tcp-c-i
ipchains -F tcp-s-o
ipchains -F misc-out
ipchains -F misc-in
ipchains -F icmp-in
ipchains -F icmp-out
ipchains -F log-in
ipchains -F log-out
```

```
ipchains -X spoofed
ipchains -X tcp-c-o
ipchains -X tcp-s-i
ipchains -X udp-c-o
ipchains -X udp-s-i
ipchains -X tcp-c-i
ipchains -X tcp-s-o
ipchains -X misc-out
ipchains -X misc-in
ipchains -X icmp-in
ipchains -X icmp-out
ipchains -X log-in
ipchains -X log-out
```

```
# Voreinstellung: Pakete ablehnen.
```

```
ipchains -P input  DENY
ipchains -P output REJECT
ipchains -P forward REJECT
```

```
# -----
```

```
# LOOPBACK
```

```
# Keine Einschränkungen für das Loopback-Interface.
```

```
ipchains -A input  -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT
```

```
# -----
# LAN

ipchains -A input -i $LAN_INTERFACE -j ACCEPT
ipchains -A output -i $LAN_INTERFACE -j ACCEPT

# -----
# Benutzerdefinierte Chains

ipchains -N spoofed
ipchains -N tcp-c-o
ipchains -N tcp-s-i
ipchains -N udp-c-o
ipchains -N udp-s-i
ipchains -N tcp-c-i
ipchains -N tcp-s-o
ipchains -N misc-out
ipchains -N misc-in
ipchains -N icmp-in
ipchains -N icmp-out
ipchains -N log-in
ipchains -N log-out

# -----
# EXTERNES INTERFACE: CHAIN: Gefälschte Absenderadressen (spoofed)

# Verbindungen von problematischen Sites ablehnen.

# /etc/rc.d/rc.firewall.blocked enthält eine Liste gesperrter Adressen
# im Format
# ipchains -A spoofed -s <address/mask> -j DENY

# Pakete von einem Absender in der Blockadefliste ablehnen.
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

# -----
# Gefälschte und illegale Adressen ablehnen.
# Gefälschte Pakete sowie völlig illegale Absenderadressen ablehnen.
# Wir wollen auch nicht an fehlerhafte Empfänger senden.

# Gefälschte Pakete von unserer eigenen Adresse ablehnen.
ipchains -A spoofed -s $IPADDR -j DENY -l

# Pakete mit einem privaten Klasse-A-Netz in Absender
ipchains -A spoofed -s $CLASS_A -j DENY
```

Anhang B • Firewall-Beispiele und hilfreiche Skripten 475

```
# Pakete mit einem privaten Klasse-B-Netz in Absender
ipchains -A spoofed -s $CLASS_B -j DENY

# Pakete mit einem privaten Klasse-C-Netz in Absender
ipchains -A spoofed -s $CLASS_C -j DENY

# Pakete von Loopback ablehnen.
ipchains -A spoofed -s $LOOPBACK -j DENY

# Klasse-D-Multicast-Adressen ablehnen.
# Multicast ist nur als Absender illegal; es benutzt UDP.
ipchains -A spoofed -s $CLASS_D_MULTICAST -j DENY -l

# Fehlerhafte Broadcast-Pakete ablehnen.
ipchains -A spoofed -s $BROADCAST_DEST -j DENY -l

# -----
# EXTERNES INTERFACE: CHAIN: Abgehende Pakete von TCP-Clients (tcp-c-o -p tcp)

# HTTP-Client (80)

ipchains -A tcp-c-o -p tcp \
        -d $ANYWHERE http -j ACCEPT

# -----

# NNTP (119) - News als Usenet-Client lesen und schreiben

ipchains -A tcp-c-o -p tcp \
        -d $NEWS_SERVER nntp -j ACCEPT

# -----

# POP (110) - Mailempfang als POP-Client

ipchains -A tcp-c-o -p tcp \
        -d $POP_SERVER pop-3 -j ACCEPT

# -----

# SMTP-Client: Mailversand (25)

ipchains -A tcp-c-o -p tcp \
        -d $ANYWHERE smtp -j ACCEPT
```

476 ————— *Optimierung der Firewall-Regeln*

```
# -----  
  
# AUTH-Client (113)  
  
ipchains -A tcp-c-o -p tcp \  
        -d $ANYWHERE auth -j ACCEPT  
  
# -----  
  
# TELNET (23) - abgehende Verbindungen zu fremden Sites  
  
ipchains -A tcp-c-o -p tcp \  
        --destination-port telnet -j ACCEPT  
  
# -----  
  
# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern  
  
ipchains -A tcp-c-o -p tcp \  
        --destination-port ssh -j ACCEPT  
  
# -----  
  
# FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern  
  
# Datenkanäle im passiven Modus  
  
ipchains -A tcp-c-o -p tcp \  
        --destination-port $UNPRIVPORTS -j ACCEPT  
  
# Datenkanäle im aktiven Modus  
  
ipchains -A tcp-c-o -p tcp ! -y \  
        --destination-port ftp-data -j ACCEPT  
  
# abgehende Verbindung  
  
ipchains -A tcp-c-o -p tcp \  
        --destination-port ftp -j ACCEPT  
  
# -----  
  
# SSL-Client (443)  
  
ipchains -A tcp-c-o -p tcp \  
        -d $ANYWHERE https -j ACCEPT
```

Anhang B • Firewall-Beispiele und hilfreiche Skripten 477

```
# -----
# WHOIS-Client (43)
ipchains -A tcp-c-o -p tcp \
        --destination-port whois -j ACCEPT

# -----
# DNS: TCP-Client (53)
# TCP Anfragen von einem Client sind vom Protokoll erlaubt,
# wenn eine UDP-Anfrage fehlschlägt. Das kommt selten vor.
ipchains -A tcp-c-o -p tcp \
        -d $ANYWHERE domain -j ACCEPT

# -----
# EXTERNES INTERFACE: CHAIN: Ankommende Pakete von TCP-Servern (tcp-s-i -p tcp)
# HTTP-Client (80)
ipchains -A tcp-s-i -p tcp \
        -s $ANYWHERE http -j ACCEPT

# -----
# NNTP (119) - News als Usenet-Client lesen und schreiben
ipchains -A tcp-s-i -p tcp \
        -s $NEWS_SERVER nntp -j ACCEPT

# -----
# POP (110) - Mailempfang als POP-Client
ipchains -A tcp-s-i -p tcp \
        -s $POP_SERVER pop-3 -j ACCEPT

# -----
# SMTP-Client: Mailversand (25)
ipchains -A tcp-s-i -p tcp \
        -s $SMTP_GATEWAY smtp -j ACCEPT
```

478 ————— *Optimierung der Firewall-Regeln*

```
# -----  
  
# AUTH-Client (113)  
  
ipchains -A tcp-s-i -p tcp \  
        -s $ANYWHERE auth -j ACCEPT  
  
# -----  
  
# TELNET (23) - abgehende Verbindungen zu fremden Sites  
  
ipchains -A tcp-s-i -p tcp \  
        --source-port telnet -j ACCEPT  
  
# -----  
  
# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern  
  
ipchains -A tcp-s-i -p tcp \  
        --source-port ssh -j ACCEPT  
# -----  
  
# FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern  
  
# Datenkanäle im passiven Modus  
  
ipchains -A tcp-s-i -p tcp \  
        --source-port $UNPRIVPORTS -j ACCEPT  
  
# abgehende Verbindung  
  
ipchains -A tcp-s-i -p tcp \  
        --source-port ftp -j ACCEPT  
  
# -----  
  
# SSL-Client (443)  
  
ipchains -A tcp-s-i -p tcp \  
        -s $ANYWHERE https -j ACCEPT  
  
# -----  
  
# WHOIS-Client (43)  
  
ipchains -A tcp-s-i -p tcp \  
        --source-port whois -j ACCEPT
```

```
# -----

# DNS: TCP-Client (53)

# TCP-Anfragen von einem Client sind vom Protokoll erlaubt,
# wenn eine UDP-Anfrage fehlschlägt. Das kommt selten vor.

ipchains -A tcp-s-i -p tcp \
        -s $ANYWHERE domain -j ACCEPT

# -----
# EXTERNES INTERFACE: CHAIN: Abgehende Pakete von UDP-Clients (udp-c-o)

# DNS: UDP: Forwards an einen Peer (53)
# -----

ipchains -A udp-c-o -p udp \
        --source-port domain \
        -d $NAMESERVER_1 domain -j ACCEPT

# DNS: UDP-Client (53)
# -----

ipchains -A udp-c-o -p udp \
        --source-port $UNPRIVPORTS \
        -d $ANYWHERE domain -j ACCEPT

# -----

# NTP (123) - Zugriff auf fremde Network Time Server

ipchains -A udp-c-o -p udp \
        --source-port $UNPRIVPORTS \
        -d mein.Time.Server ntp -j ACCEPT

# -----
# EXTERNES INTERFACE: CHAIN: Ankommende Pakete von UDP-Servern (udp-s-i)

# DNS: UDP: Forwards an einen Peer (53)
# -----

ipchains -A udp-c-o -p udp \
        -s $NAMESERVER_1 domain \
        --destination-port domain -j ACCEPT

# DNS: UDP-Client (53)
```

```
ipchains -A udp-s-i -p udp \  
-s $ANYWHERE domain \  
--destination-port $UNPRIVPORTS -j ACCEPT  
  
# -----  
  
# NTP (123) - Zugriff auf fremde Network Time Server  
  
ipchains -A udp-s-i -p udp \  
-s mein.Time.Server ntp \  
--destination-port $UNPRIVPORTS -j ACCEPT  
  
# -----  
# EXTERNES INTERFACE: CHAIN: Ankommende Pakete von TCP-Clients (tcp-c-i -p tcp)  
  
# HTTP-Server (80)  
  
# ankommende HTTP-Anfragen  
  
ipchains -A tcp-c-i -p tcp \  
--destination-port http -j ACCEPT  
  
# -----  
  
# AUTH-Server (113)  
  
# ankommende AUTH-Anfragen  
  
ipchains -A tcp-c-i -p tcp \  
--destination-port auth -j ACCEPT  
  
# -----  
  
# UNPRIVILEGIERTE PORTS  
# Manche Ports würden Protokoll- oder Administrationsschwierigkeiten beschwören  
  
# OpenWindows: ankommende Verbindung  
ipchains -A tcp-c-i -p tcp -y \  
--destination-port $OPENWINDOWS_PORT -j DENY  
  
# X Window: ankommende Verbindung  
ipchains -A tcp-c-i -p tcp -y \  
--destination-port $XWINDOW_PORTS -j DENY -1  
  
# SOCKS: ankommende Verbindung  
ipchains -A tcp-c-i -p tcp -y \  
--destination-port $SOCKS_PORT -j DENY
```


Anhang B • Firewall-Beispiele und hilfreiche Skripten 481

```
# NFS: TCP-Verbindungen
ipchains -A tcp-c-i -p tcp -y \
    --destination-port $NFS_PORT -j DENY -1

# -----
# EXTERNES INTERFACE: CHAIN: Abgehende Pakete von TCP-Servern (tcp-s-o -p tcp)

# HTTP (80)

# ankommende HTTP-Anfragen

ipchains -A tcp-s-o -p tcp \
    --source-port http -j ACCEPT

# -----

# AUTH (113)

# ankommende AUTH-Anfragen

ipchains -A tcp-s-o -p tcp \
    --source-port auth -j ACCEPT

# -----
# EXTERNES INTERFACE: CHAIN: Verschiedene abgehende Pakete (misc-out)

# SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern

ipchains -A misc-out -p tcp \
    --source-port $SSH_PORTS \
    --destination-port ssh -j ACCEPT

# -----

# TRACEROUTE

# Abgehende traceroutes

ipchains -A misc-out -p udp \
    --source-port $TRACEROUTE_SRC_PORTS \
    --destination-port $TRACEROUTE_DEST_PORTS -j ACCEPT

# Ankommende traceroutes vom Internet-Provider.

# (3) Dest_Unreachable, letzte Antwort auf traceroute
```

482 ————— **Optimierung der Firewall-Regeln**

(11) Time_Exceeded, Antwort einer Zwischenstation auf traceroute

```
ipchains -A misc-out -p icmp \  
        --icmp-type port-unreachable -d $MY_ISP -j ACCEPT
```

```
ipchains -A misc-out -p icmp \  
        --icmp-type time-exceeded -d $MY_ISP -j ACCEPT
```

PING

Abgehende Pongs (Ping-Antworten) an vertrauenswürdige Hosts

```
ipchains -A misc-out -p icmp \  
        --icmp-type echo-reply -d $MY_ISP -j ACCEPT
```

EXTERNES INTERFACE: CHAIN: Verschiedene ankommende Pakete (misc-in)

SSH-Client (22) - abgehende Verbindungen zu fremden SSH-Servern

```
ipchains -A misc-in -p tcp ! -y \  
        --source-port ssh \  
        --destination-port $SSH_PORTS -j ACCEPT
```

FTP-Client (20, 21) - abgehende Verbindungen zu fremden Servern

Datenkanäle im aktiven Modus

```
ipchains -A misc-in -p tcp \  
        --source-port ftp-data \  
        --destination-port $UNPRIVPORTS -j ACCEPT
```

NFS: UDP-Verbindungen

```
ipchains -A misc-in -p udp \  
        --destination-port $NFS_PORT -j DENY -1
```

TRACEROUTE

```
# Ankommende traceroutes vom Internet-Provider.

ipchains -A misc-in -p udp \
        -s $MY_ISP $TRACEROUTE_SRC_PORTS \
        --destination-port $TRACEROUTE_DEST_PORTS -j ACCEPT

# -----

# PING

# Ankommende pings von vertrauenswürdigen Hosts.

ipchains -A misc-in -p icmp \
        -s $MY_ISP echo-request -j ACCEPT

# -----

# CHAIN FÜR ANKOMMENDE ICMP-PAKETE

ipchains -A icmp-in -p icmp --icmp-type echo-reply -j ACCEPT
ipchains -A icmp-in -p icmp --icmp-type destination-unreachable -j ACCEPT
ipchains -A icmp-in -p icmp --icmp-type source-quench -j ACCEPT
ipchains -A icmp-in -p icmp --icmp-type time-exceeded -j ACCEPT
ipchains -A icmp-in -p icmp --icmp-type parameter-problem -j ACCEPT

# -----

# CHAIN FÜR ABGEHENDE ICMP-PAKETE

ipchains -A icmp-out -p icmp --icmp-type fragmentation-needed -j ACCEPT
ipchains -A icmp-out -p icmp --icmp-type source-quench -j ACCEPT
ipchains -A icmp-out -p icmp --icmp-type echo-request -j ACCEPT
ipchains -A icmp-out -p icmp --icmp-type parameter-problem -j ACCEPT

# -----

# Pakete filtern

ipchains -A input -i $EXTERNAL_INTERFACE -j spoofed
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp ! -y \
        -d $IPADDR $UNPRIVPORTS -j tcp-s-i
ipchains -A input -i $EXTERNAL_INTERFACE -p udp \
        -d $IPADDR -j udp-s-i
ipchains -A input -i $EXTERNAL_INTERFACE -p icmp \
        -d $IPADDR -j icmp-in
ipchains -A input -i $EXTERNAL_INTERFACE -p tcp \
        -s $ANYWHERE $UNPRIVPORTS \
        -d $IPADDR -j tcp-c-i
```

```
ipchains -A input -i $EXTERNAL_INTERFACE \
        -d $IPADDR -j misc-in
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
        -s $IPADDR \
        -d $ANYWHERE $UNPRIVPORTS -j tcp-s-o
ipchains -A output -i $EXTERNAL_INTERFACE -p tcp \
        -s $IPADDR $UNPRIVPORTS -j tcp-c-o
ipchains -A output -i $EXTERNAL_INTERFACE -p udp \
        -s $IPADDR -j udp-c-o
ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
        -s $IPADDR -j icmp-out
ipchains -A output -i $EXTERNAL_INTERFACE \
        -s $IPADDR -j misc-out

# -----
# CHAIN FÜR FORWARDING UND MASQUERADING

ipchains -A forward -i $EXTERNAL_INTERFACE -s $LAN_ADDRESSES -j MASQ

# -----
# Abgewiesene Pakete protokollieren

ipchains -A input -i $EXTERNAL_INTERFACE -j log-in

ipchains -A output -i $EXTERNAL_INTERFACE -j log-out

# -----

echo "fertig"

exit 0
```

B.4 Spezialskripten

Immer wieder schreibt mir jemand und fragt, ob ich nicht ein Skript habe, das dies oder das tut. Die vier Skripten, die am häufigsten verlangt werden, habe ich hier mit abgedruckt.

B.4.1 Alles erlauben

Das Skript `accept-all` ist ein Hilfsmittel bei der Fehlersuche. Ab und zu müssen Sie ausprobieren, ob ein bestimmter Dienst vielleicht nicht nur deshalb nicht verfügbar ist, weil Ihre Firewall irgendwo einen Fehler enthält.

Eine Beschreibung des `accept-all`-Skripts im Format einer Manual-Page folgt:

Name

`accept-all` – alle Pakete dürfen passieren

Synopsis

`accept-all`

Beschreibung

Löscht alle Firewall-Regeln und setzt die Voreinstellung auf `ACCEPT`. Muss als `root` ausgeführt werden.

Das Skript `accept-all` enthält die folgenden Zeilen:

```
#!/bin/sh

# Alle bestehenden Regeln löschen
ipchains -F input
ipchains -F output
ipchains -F forward

# Voreinstellung auf ACCEPT setzen
ipchains -P input  ACCEPT
ipchains -P output ACCEPT
ipchains -P forward ACCEPT
```

B.4.2 Alles sperren

Das `deny-all`-Skript ist zum Teil ein Hilfsmittel bei der Fehlersuche, zum Teil eine Notbremse, wenn ein neues Firewall-Skript noch Fehler enthält und sich nicht ausführen lässt, und zum Teil ein Panik-Knopf. Manchmal muss man alle Zugriffe aus dem Internet blockieren und in Ruhe von vorn anfangen.

Eine Beschreibung von `deny-all` als Man-Page:

Name

`deny-all` – allen Netzwerkverkehr vom und ins Internet sperren

Synopsis`deny-all`*Beschreibung*

Löscht alle Firewall-Regeln und setzt die Voreinstellung für das externe Netzwerkinterface auf DENY. Muss als root ausgeführt werden.

Das Skript `deny-all` enthält die folgenden Zeilen:

```
#!/bin/sh

LOOPBACK_INTERFACE="lo"
LAN_INTERFACE="eth1"

# Alle bestehenden Regeln löschen
ipchains -F input
ipchains -F output
ipchains -F forward

# Voreinstellung: alle Pakete ablehnen
ipchains -P input  DENY
ipchains -P output REJECT
ipchains -P forward REJECT

ipchains -A input  -i $LOOPBACK_INTERFACE -j ACCEPT
ipchains -A output -i $LOOPBACK_INTERFACE -j ACCEPT

# Zugriff auf den Firewall-Computer aus dem LAN erlauben
ipchains -A input  -i $LAN_INTERFACE -j ACCEPT
ipchains -A output -i $LAN_INTERFACE -j ACCEPT
```

B.4.3 Eine IP-Adresse blockieren

`block-ip` ist praktisch, wenn sich jemand daneben benimmt und sie ihn sofort aussperren wollen, ohne lange im Firewall-Skript herumzueditieren und die Firewall manuell neu zu initialisieren. `block-ip` fügt eine neue Regel in die `input-Chain` ein und hängt diese Regel auch an die Datei mit den gesperrten Adressen an, sodass die Adresse dauerhaft blockiert bleibt.

Eine Beschreibung von `block-ip` als Man-Page:

Name

`block-ip` – Alle ankommenden Pakete von einer Adresse sperren

Synopsis

`block-ip Adresse[/Maske]`

Beschreibung

Fügt am Anfang der input-Chain eine neue DENY-Regel ein. *Adresse* kann ein vollqualifizierter Hostname sein, eine IP-Adresse oder ein Adressbereich in Form von Adresse und Bitmaske.

Wenn die Datei `/etc/rc.d/rc.firewall.blocked` noch nicht existiert, wird sie neu angelegt. Die Firewall-Regel wird ans Ende von `/etc/rc.d/rc.firewall.blocked` angehängt, damit sie permanent in der Liste der gesperrten Adressen bleibt.

Muss als root ausgeführt werden.

Das Skript `block-ip` enthält:

```
#!/bin/sh

EXTERNAL_INTERFACE="eth0"

# Wenn die Sperrdatei noch nicht existiert, legen wir sie an.
if [ ! -f /etc/rc.d/rc.firewall.blocked ]; then
    touch /etc/rc.d/rc.firewall.blocked
fi

# Neue Regel am Anfang der bestehenden input-Chain einfügen
ipchains -i input -i $EXTERNAL_INTERFACE -s $1 -j DENY

# Neue Regel ans Ende der Sperrdatei anhängen
echo ipchains -i input -i $EXTERNAL_INTERFACE -s $1 -j DENY \
    >> /etc/rc.d/rc.firewall.blocked
```

B.4.4 Eine blockierte IP-Adresse wieder zulassen

Wenn Sie eine DENY-Regel umgehend aus der Liste der gesperrten Sites entfernen wollen, ohne das Firewall-Skript zu bearbeiten und die Firewall neu zu initialisieren, ist `unblock-ip` recht handlich. Sie müssen die entsprechende Regel allerdings immer noch per Hand aus `/etc/rc.d/rc.firewall.blocked` löschen.

Es folgt eine Beschreibung von `unblock-ip` als Man-Page:

Name

`unblock-ip` – Eine DENY-Regel aus der Firewall entfernen

Synopsis

`unblock-ip Adresse[/Maske]`

Beschreibung

Löscht eine existierende DENY-Regel aus der input-Chain der Firewall. *Adresse* kann ein vollqualifizierter Hostname sein, eine IP-Adresse oder ein Adressbereich in Form von Adresse und Bitmaske. *Adresse* muss exakt mit dem Parameter übereinstimmen, mit dem die DENY-Regel erzeugt wurde.

Muss als root ausgeführt werden.

Das Skript unblock-ip ist kurz und knapp:

```
#!/bin/sh
```

```
EXTERNAL_INTERFACE="eth0"
```

```
ipchains -D input -i $EXTERNAL_INTERFACE -s $1 -j DENY
```


Anhang C

Glossar



Das Glossar definiert die Begriffe und Abkürzungen, die in diesem Buch verwendet werden.

Abgeschirmter Host

siehe Firewall, abgeschirmter Host

Abgeschirmtes Subnetz

siehe Firewall, abgeschirmtes Subnetz

ACCEPT

Entscheidung einer Firewallregel, dass ein Paket an den nächsten Empfänger weitergegeben werden darf.

ACK

TCP-Flag, das den Empfang eines TCP-Segments bestätigt.

Application Gateway

siehe Proxy, Application Gateway

auth

TCP-Port 113, entspricht dem `identd`-Server, der TCP-Verbindungen Benutzerkennungen zuordnet.

Authentifikation

Vorgang, der entscheidet, ob jemand auch wirklich derjenige ist, der zu sein er vorgibt.

Autorisierung

Vorgang, der entscheidet, welche Dienste und Ressourcen jemand benutzen darf.

Bastion

siehe Firewall, Bastion

Berkeley Internet Name Domain (BIND)

Die Implementierung des DNS-Protokolls aus Berkeley-Unix.

BIND

siehe Berkeley Internet Name Domain

Bitübertragungsschicht

siehe Physikalische Schicht

BOOTP

Ein Bootprotokoll, über das Workstations ohne eigene Festplatte ihre IP-Adresse und die des Boot-Servers herausfinden können. Danach laden sie das Betriebssystem über tftp und booten. BOOTP wurde als Ersatz für RARP entwickelt.

BOOTPC

UDP-Port 68, entspricht den Clients von BOOTP und DHCP.

bootpd

Das Server-Programm für BOOTP.

BOOTPS

UDP-Port 67, entspricht den Servern von BOOTP und DHCP.

Broadcast

Ein IP-Paket, das an alle Interfaces auf dem gleichen Netzwerk adressiert ist.

CERT

siehe Computer Emergency Response Team

CGI

siehe Common Gateway Interface

Chain

Regelliste, die festlegt, welche Pakete über ein Netzwerkinterface ankommen und welche versandt werden dürfen.

Checksumme

siehe Prüfsumme

Choke

siehe Firewall, Choke

chroot

Sowohl ein Programm als auch ein Systemaufruf. Beide definieren für ein gegebenes Verzeichnis, es sei jetzt das Root-Verzeichnis eines neuen Dateisystems. Sie rufen dann ein Programm auf, das in diesem virtuellen Dateisystem abläuft und keinen Zugriff auf den Rest des Systems hat.

Circuit-Gateway-Proxy

Diese Sorte Proxy-Server wird entweder als spezielle Anwendung für den jeweils behandelten Service implementiert, oder als allgemeiner Weiterleitungsmechanismus, der über die konkret verwendeten Protokolle nichts weiß. Das Circuit-Gateway stellt eine virtuelle Verbindung zwischen Client und Server her (siehe Bild C.1). Im Gegensatz zu Proxies auf Anwendungsebene sind die Zwischenschritte

im Verbindungsaufbau für den Benutzer völlig transparent. SOCKS ist so ein Circuit-Level-Proxy.

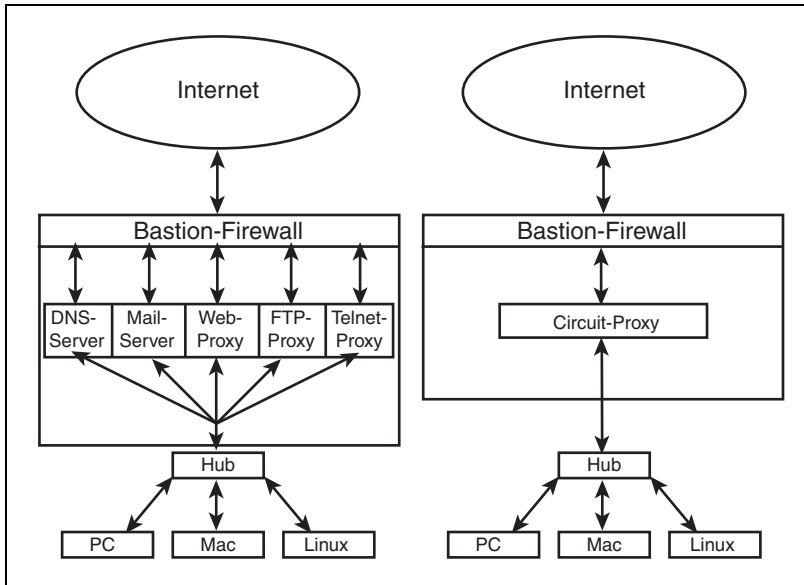


Bild C.1: Proxies: links auf Anwendungsebene, rechts als Circuit-Gateway

Client/Server-Modell

Ein Modell für verteilte Netzwerkdienste. Dabei bietet ein zentrales Programm, der Server, fremden Clients einen Dienst an. Bei diesem Dienst kann es sich z.B. um die Übertragung einer Web-Seite handeln, um den Download einer Datei aus einer Programmsammlung, um eine Datenbankanfrage, den Versand oder Empfang von E-Mail, um irgendeine Berechnung, die mit Daten vom Client ausgeführt wird, oder um die Herstellung von Verbindungen zwischen zwei oder mehr Menschen.

Common Gateway Interface (CGI)

Der Webserver führt CGI-Programme lokal für einen fremden Client aus. Weil CGI-Programme oft Perl-Skripten sind, spricht man auch von CGI-Skripten.

Computer Emergency Response Team (CERT)

Ein Zentrum, das Informationen zusammenstellt und Notfallsituationen im Bereich der Internet-Security vorbeugt. Es entstand nach dem Internet-Worm von 1988 am Software Engineering Institute der Carnegie Mellon University.

Computer Oracle and Password System (COPS)

Diese Programmsammlung überprüft eine große Zahl möglicher Sicherheitslücken auf einem Unix-System.

`control-panel`

Eine Reihe grundlegender Tools zur Systemadministration, die in einem grafischen Interface zusammengefasst sind. Ein neueres und vollständigeres Programm ist `linuxconf`. Die Entsprechung unter SuSE ist `YaST`.

COPS

siehe Computer Oracle and Password System

`crack`

Errät Passworte.

`cron`

Ein Dämon namens `crond` sowie einige Konfigurationsdateien und Skripten, die zusammen dafür sorgen, dass Standardaufgaben des Systems regelmäßig zu bestimmten Zeiten ausgeführt werden.

Dämon

Ein Server für Systemdienste, der ständig im Hintergrund läuft.

DARPA

Defense Advanced Research Projects Agency

Data Link Layer

siehe Sicherungsschicht

Demilitarisierte Zone

Ein Perimeter-Netzwerk mit den öffentlich ansprechbaren Computern, das von einem lokalen, privaten Netzwerk getrennt ist. Das private LAN ist damit vor den unsichereren öffentlichen Servern abgeschirmt.

Denial-of-Service-Angriff

Solch ein Angriff schickt Ihnen unerwartete Daten oder überflutet Ihr System mit Paketen, sodass Ihre Internet-Verbindung unterbrochen oder massiv gestört wird, Ihre Server so stark beansprucht werden, dass sie auf normale Anfragen nicht mehr reagieren, oder – im schlimmsten Falle – Ihr Rechner völlig abstürzt.

`DENY`

Entscheidung einer Firewallregel, dass ein Paket stillschweigend verworfen wird, ohne dass der Absender darüber unterrichtet wird.

DHCP

siehe Dynamic Host Configuration Protocol

`dhcpcd`

Ein Client-Programm für DHCP, das einen DHCP-Server ausfindig macht und eine IP-Adresse anfordert oder ihre Gültigkeit verlängert.

dhcpd

Der DHCP-Server wartet auf Anfragen von Clients, die einen Server oder eine IP-Adresse suchen. Er verlängert gegebenenfalls auch die Gültigkeit vergebener IP-Adressen.

DMZ

siehe Demilitarisierte Zone

DNS

siehe Domain Name Service

Domain Name Service (DNS)

Ein globaler Datenbankservice im Internet. Seine Hauptfunktion ist die Übersetzung von symbolischen Hostnamen in IP-Adressen und umgekehrt.

dual-homed

So nennt man einen Computer mit zwei oder mehr Netzwerkinterfaces. Siehe auch »multi-homed«.

Dynamic Host Configuration Protocol (DHCP)

DHCP vergibt dynamische IP-Adressen und stellt den Clients Informationen über Server und Routing zur Verfügung. DHCP entstand als Ersatz für BOOTP.

Dynamisch zugewiesene Adresse

IP-Adresse, die einem Client durch einen zentralen Server – z.B. einen DHCP-Server – nur vorübergehend zugewiesen wurde. Dynamische Adressen werden typischerweise von Kunden oder Angestellten benutzt und kommen aus einem Pool eingetragener IP-Adressen, der dem Internet-Provider bzw. der Firma gehört.

Ethernet-Frame

In einem Ethernet-LAN werden IP-Datagramme in den so genannten Frames verpackt.

File Transfer Protocol (FTP)

Bezeichnung für das Protokoll und die Programme, die man zur Dateiübertragung zwischen zwei Computern verwendet.

Filter (einer Firewall)

Die Regeln einer Firewall beschreiben ein Paket. Wenn ein Paket mit dieser Beschreibung übereinstimmt, entscheidet die Regel darüber, ob es akzeptiert oder verworfen wird. Zu einem Filter gehören neben Absender- und Empfänger-Adresse sowie Absender- und Empfänger-Port das Protokoll, der TCP-Verbindungszustand und gegebenenfalls der ICMP-Nachrichtentyp.

finger

Ein Programm, das Informationen über Benutzer nachschlägt.

Firewall

Ein Computer, ein Router oder eine Anwendung, der bzw. die Sicherheitsbestimmungen durchsetzt, indem er/sie den Zugang zu System- und Netzwerkressourcen kontrolliert.

Firewall, abgeschirmter Host

Eine Firewall mit einem einzelnen System, bei dem lokale Benutzer sich explizit an der Firewall anmelden müssen, bevor sie von diesem Computer aus auf das Internet zugreifen dürfen. Diese Sorte Firewall betreibt kein Routing zwischen Netzwerken. Neben einem Paketfilter setzt man NAT oder Proxies auf Anwendungsebene ein. Bild C.2 zeigt die Paketfilter-Funktionalität einer Bastion-Firewall in Kombination mit den öffentlichen Servern, die man sonst vielleicht in einem DMZ-Netzwerk unterbringen würde.

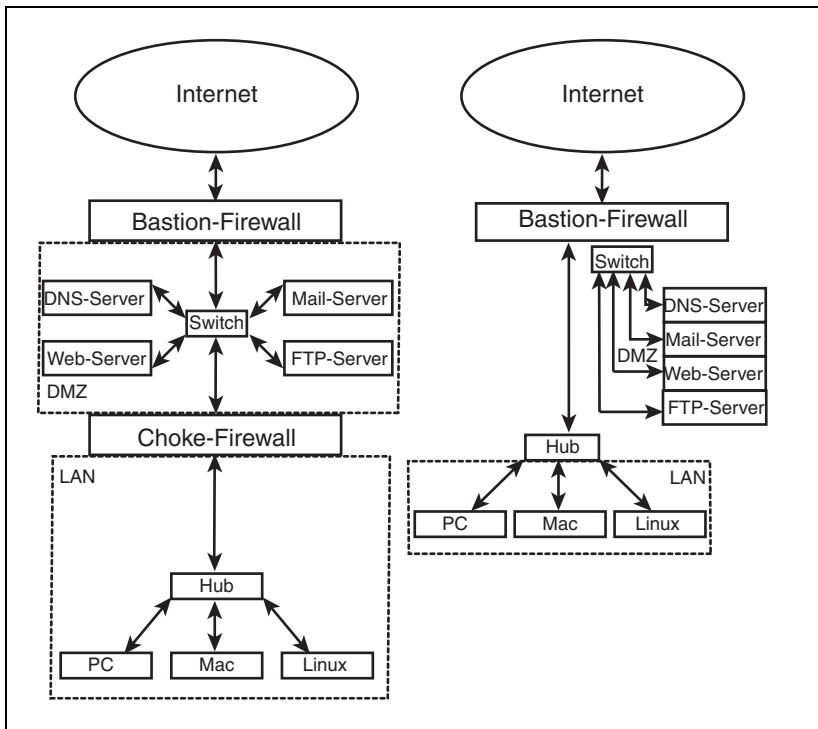


Bild C.2: Zwei Firewall-Architekturen in Form eines abgeschirmten Hosts

Firewall, abgeschirmtes Subnetz

Eine Firewallarchitektur mit einer Bastion-Firewall und einem DMZ-Netzwerk, bei der ein LAN vom direkten Internet-Zugang abgeschirmt ist. Die DMZ mit den öffentlich zugänglichen Servern ist ein separates Netzwerk oder ein vom privaten LAN abgeschirmtes Subnetz. Bild C.3 zeigt die zwei Hauptvarianten dieser Architektur.

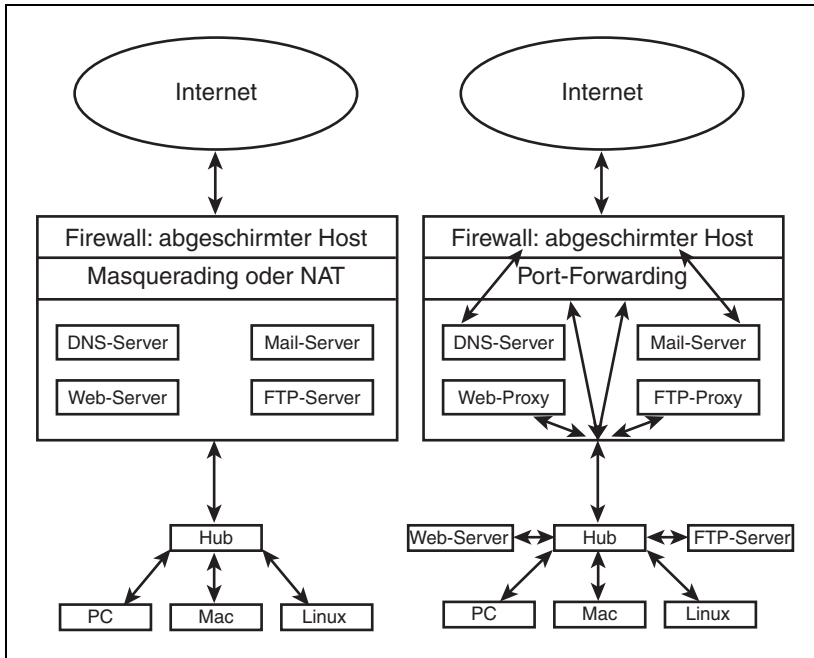


Bild C.3: Zwei Firewall-Architekturen mit abgeschirmten Subnetzen

Firewall, Bastion

Eine Firewall mit zwei oder mehr Netzwerkinterfaces. Sie dient als Übergang zwischen den angeschlossenen Netzwerken, meistens zwischen einem LAN und dem Internet. Weil die Bastion der einzige Verbindungspunkt zwischen den beiden Netzen ist, wird sie bestmöglich abgesichert.

Firewall, Choke

Eine LAN-Firewall mit zwei oder mehr Netzwerkinterfaces, die als Gateway (oder Brückenpunkt) zwischen den angeschlossenen Netzwerken dient. Eine Seite liegt in einer DMZ, einem Perimeter-Netzwerk zwischen der Choke und einer Bastion-Firewall. Die anderen Netzwerkinterfaces stellen Verbindungen zu internen, privaten LANs her.

Firewall, Paketfilter-

Eine auf Netzwerk- und Transportschicht implementierte Firewall, die Netzwerkdaten paketweise filtert und anhand von Informationen aus dem Paket-Header Entscheidungen über das Routing fällt.

Flooding

Ein Denial-of-Service-Angriff, bei dem das Opfer – ein einzelner Computer oder ein ganzes Netzwerk – mit mehr Paketen eines bestimmten Typs überschüttet wird, als es verträgt.

forwarden

Neudeutsch für »weiterleiten«: Ein Paket von einem Computer zu einem anderen weiterleiten, wobei das Paket von einem Netzwerk in ein anderes übergeht.

Fragment

Ein IP-Paket mit einem Stück eines TCP-Segments.

Frame

siehe Ethernet-Frame

FTP

siehe File Transfer Protocol

FTP, anonymes

Ein FTP-Dienst, den jeder beliebige Client benutzen darf.

FTP, authentifiziert

Ein FTP-Dienst, der nur nach Eingabe eines zuvor definierten Benutzernamens und des zugehörigen Passworts zugänglich wird.

Gateway

Ein Computer oder Programm, das Pakete zwischen zwei Netzwerken übermittelt.

Handshake

Beim Aufbau einer TCP-Verbindung erfolgt ein so genannter Handshake: In der ersten Nachricht eines Clients an den Server ist das SYN-Flag als Bitte um Verbindungsaufbau gesetzt. Gleichzeitig wird eine Sequenznummer für die Synchronisation übertragen, die der Client als Ausgangspunkt verwendet, um die folgenden von ihm versandten Pakete zu nummerieren.

Der Server antwortet mit einem Bestätigungsflag (ACK) auf die SYN-Nachricht und setzt seinerseits das SYN-Flag. Er fügt der Nachricht die vom Client erhaltene Sequenznummer plus eins hinzu; damit ist die Client-Nachricht bestätigt. Auch er setzt eine Sequenznummer für die Synchronisation, die der Server als Ausgangspunkt für die Paketnummerierung verwenden wird.

Der Client bestätigt mit einem ACK das SYN-ACK des Servers. Er antwortet mit der vom Server erhaltenen Sequenznummer plus eins und bestätigt damit ihren Empfang. Die Verbindung ist damit hergestellt.

`hosts.allow` **und** `hosts.deny`
Konfigurationsdateien der `tcp_wrapper`.

HOWTO

Neben den Manual-Pages zusätzliche Dokumentation, unter Linux zu den verschiedensten Themen in mehreren Formaten und vielen Sprachen verfügbar. Für die Koordination der HOWTO-Dokumente zeichnet das Linux Documentation Project verantwortlich.

HTTP

siehe HyperText Transfer Protocol

Hub

Ein Repeater, der mehrere Netzwerksegmente physikalisch miteinander verbindet, die physikalische Ausdehnung eines Netzwerkes erweitert oder Netzwerksegmente verschiedener physikalischer Typen zusammenführt.

HyperText Transfer Protocol (HTTP)

Wird von Web-Servern und -Browsern verwendet.

IANA

Internet Assigned Numbers Authority

ICMP

siehe Internet Control Message Protocol

identd

Server für das `auth`-Protokoll zur Benutzeridentifizierung.

ifstatus

Prüft die Konfiguration der Netzwerkschnittstellen und zeigt alle Interfaces an, die im Debugging-Modus oder im *Promiscuous Mode* sind, d.h. auch Pakete mithören, die nicht für den eigenen Computer bestimmt sind.

IMAP

siehe Internet Message Access Protocol

inetd

Eine Art »Super-Server«. `inetd` wartet auf verschiedenen Ports auf Anfragen. Sobald eine eintrifft, startet es einen passenden Server, der die Anfrage bedient.

inetd.conf

Konfigurationsdatei von `inetd`.

innd

Server für Usenet News.

Internet Control Message Protocol (ICMP)

Protokoll für die Übermittlung von Status- und Kontrollnachrichten auf der Netzwerkschicht.

Internet Message Access Protocol (IMAP)

Ein Protokoll zum Abholen von Mail von einem entsprechenden Server.

Internet Relay Chat (IRC)

Schriftliche elektronische Kommunikation zwischen Einzelpersonen und Gruppen in Echtzeit.

IP

Internet Protocol.

IP-Adresse

Eine eindeutige numerische Bezeichnung für ein bestimmtes Netzwerk oder eine bestimmte Schnittstelle auf einem Netzwerk. Diese Software-Adresse ist direkt in einen symbolischen, bedienerfreundlichen Host- oder Netzwerknamen übersetzbar. Die IP-Adresse eines Netzwerkinterfaces ist einer oder mehreren Hardware-Adressen zugeordnet.

IP-Datagramm

Ein Paket der Netzwerkschicht von IP.

ipchains

Programm zur Administration der aktuellen Implementierung der IPFW-Firewall.

IPFW

IP-Firewall-Mechanismus.

ipfwadm

Programm zur Administration der IPFW-Firewall, das vor Einführung von ipchains eingesetzt wurde.

IRC

siehe Internet Relay Chat

ISP

Internet-Service-Provider.

Klasse einer Netzwerkadresse

Es gibt insgesamt fünf Klassen von Netzwerkadressen. Eine IPv4-Adresse besteht aus 32 Bits. Der Adressbereich wird über die vier signifikantesten Bits davon (MSB) in die Klassen A bis E unterteilt. Die Klasse A enthält 126 Netze, jedes mit über 16 Millionen Rechnern. Zur Klasse B gehören 16 000 Netze mit jeweils

bis zu 64 000 Hosts. Klasse C umfasst zwei Millionen Netze mit bis zu 254 Rechnern. Klasse D ist den Multicast-Adressen vorbehalten, Klasse E experimentellen Zwecken.

klogd

Protokolldämon des Kernels; er nimmt Fehler- und Statusmeldungen des Kernels entgegen und schreibt sie in Zusammenarbeit mit `syslogd` in die entsprechenden Protokolldateien.

LAN

Local Area Network, lokales Netz.

linuxconf

siehe control-panel

localhost

Symbolischer Name aus `/etc/hosts`, mit dem häufig das Loopback-Interface eines Computers bezeichnet wird.

Loopback-Interface

Eine besondere Netzwerkschnittstelle auf Softwarebasis, über die das System lokale Pakete an die eigene Maschine ausliefert, ohne dass die Hardware-Schnittstelle und der zugehörige Treiber benutzt werden müssten.

Manual-Page

»Handbuchseite«. Standardformat für Unix-Dokumentation. Manual-Pages existieren z.B. für praktisch alle Benutzer- und Systemkommandos sowie für System- und Bibliotheksaufrufe, Gerätetypen und Dateisysteme.

Masquerading

Vorgang, bei dem die Absenderadresse eines Paketes mit der IP der Firewall bzw. des Gateways ersetzt wird, wodurch die IP-Adresse des LANs verborgen bleibt. Nach außen wirkt es so, als käme das Paket vom Gateway, nicht von einem Computer im LAN. Bei ankommenden Antworten von einem fremden Server wird der Vorgang umgekehrt: Die Empfängeradresse des Pakets – die IP-Adresse der Firewall – wird durch die Adresse des Clients im internen LAN ersetzt. IP-Masquerading bezeichnet man allgemeiner auch als *Network Address Translation* oder kurz NAT.

Maximum Transmission Unit (MTU)

Die größte aufgrund der Eigenheiten des zugrunde liegenden Netzwerkes zulässige Paketgröße.

MD5

Kryptographischer Prüfsummen-Algorithmus, mit dessen Hilfe man Datenintegrität über digitale Unterschriften, die so genannten *Message Digests*, absichert.

MTU

siehe Maximum Transmission Unit

Multicast

Ein Paket mit einer Multicast-Adresse der Klasse D als Empfänger. Multicast-Clients sind bei den Routern registriert und erhalten Pakete, die an eine bestimmte Multicast-Adresse gerichtet sind.

multi-homed

So nennt man einen Computer mit zwei oder mehr Netzwerkinterfaces. Siehe auch »dual-homed«.

named

Der DNS-Nameserver.

Nameserver, primärer

Autoritativer Nameserver für eine Domain oder eine Zone aus dem Domainbereich. Der Server verwaltet eine vollständige Datenbank der Hostnamen und IP-Adressen für die jeweilige Zone.

Nameserver, sekundärer

Sicherheitskopie oder Helfer für einen primären Nameserver.

NAT

siehe Network Address Translation

netstat

Sammelt aus diversen Kernel-Tabellen Informationen zum Netzwerkstatus zusammen.

Network Address Translation (NAT)

Vorgang, bei dem die Absenderadresse eines Paketes mit der IP der Firewall bzw. des Gateways ersetzt wird, wodurch die IP-Adresse des LANs verborgen bleibt. Nach außen wirkt es so, als käme das Paket vom Gateway, nicht von einem Computer im LAN. Bei ankommenden Antworten von einem fremden Server wird der Vorgang umgekehrt: Die Empfängeradresse des Pakets – die IP-Adresse der Firewall – wird durch die Adresse des Clients im internen LAN ersetzt.

Network File System (NFS)

Zur gemeinsamen Nutzung von Dateisystemen auf vernetzten Computern.

Network Information Service (NIS)

Zur zentralen Verwaltung von Benutzeraccounts und Informationen über die Computer eines Netzwerks.

Network News Transfer Protocol (NNTP)

Datenübertragungsprotokoll von Usenet News.

Network Time Protocol (NTF)

Z.B. von `xntpd` und `ntdate` benutzt.

Netzwerkschicht

Im Deutschen auch *Vermittlungsschicht* genannt, englisch *network layer*. Die dritte Schicht des OSI-Referenzmodells. Sie repräsentiert die Kommunikation zwischen zwei Computern, z.B. Routing und Auslieferung eines IP-Datagramms von Ihrem Computer als Absender zu einem fremden Server als Empfänger. Im TCP/IP-Referenzmodell entspricht dem die zweite Schicht, die *Internet-Schicht*.

NFS

siehe Network File System

NIS

siehe Network Information Service

nmap

Ein neueres Hilfsmittel zur Sicherheitsüberwachung von Netzwerken. Ein Port-scanner, der viele der heute eingesetzten Scan-Techniken beherrscht.

NNTP

siehe Network News Transfer Protocol

NTP

siehe Network Time Protocol

ntdate

Stellt eine Verbindung zu ein oder mehreren Zeitservern her und aktualisiert die Systemuhr.

OSI-Referenzmodell

(*Open Systems Interconnection*.) Ein von der International Organization for Standardization (ISO) entwickeltes Netzwerkmodell mit sieben Schichten, das bei der Definition von Netzwerkstandards helfen sollte.

OSPF

Open Shortest Path First: Ein Routing-Protokoll für TCP/IP, das eines der heute am häufigsten benutzten Protokolle ist. Eingesetzt z.B. im Routing-Dämon `gated`.

Paket

Ein IP-Datagramm.

Paketfilter

siehe Firewall, Paketfilter

Paketflut

siehe Flooding

PATH

Diese Umgebungsvariable legt fest, in welchen Verzeichnissen und in welcher Reihenfolge die Shell ausführbare Programme sucht.

Peer-to-Peer

Kommunikationsmodus zwischen zwei Servern. Oft – aber nicht immer – unterscheidet sich das Peer-to-Peer-Protokoll von dem Protokoll, mit dem Server und Client kommunizieren.

Physikalische Schicht

Im Deutschen auch »Bitübertragungsschicht« genannt, englisch *physical layer*. Die erste Schicht des OSI-Referenzmodells. Sie steht für das physikalische Medium, über das die elektronischen Signale zwischen zwei benachbarten Netzwerkgeräten übermittelt werden, z.B. Kupferdraht, Fiberglas, Paketfunk oder Infrarot. Im TCP/IP-Referenzmodell zählt sie mit zur ersten Schicht, der Subnetz-Schicht.

PID

Prozess-ID, eine systemweit eindeutige numerische Bezeichnung für einen Prozess, meist assoziiert mit dem entsprechenden Eintrag in der Prozesstabelle.

ping

Einfaches Tool zur Netzwerkanalyse; es bestimmt, ob ein anderer Computer erreichbar ist und antwortet. `ping` schickt eine ICMP-echo-request-Nachricht; der andere Computer antwortet mit einem ICMP-echo-reply.

Policy

Die Policy einer Firewall-Chain – egal, ob `input`, `output` oder `forward` – gibt an, was mit einem Paket passiert, wenn keine der Regeln aus der Liste passt.

Bei einer Policy von `ACCEPT` werden alle Pakete angenommen, wenn keine Regel zutrifft. Die meisten Regeln sind daher vom Typ `DENY` und definieren Ausnahmen, die nicht akzeptiert werden sollen.

Eine Policy von `DENY` (oder auch `REJECT`) hingegen verwirft alle Pakete ohne passende Regel. Die meisten Firewall-Regeln sind also `ACCEPTs` und geben explizit an, welche Pakete doch passieren dürfen.

POP

Das *Post Office Protocol* dient dem Abholen von Mail von einem entsprechenden Server.

Port

Numerischer Bezeichner für einen bestimmten Netzwerk-Kommunikationskanal in TCP oder UDP. Verantwortlich für Port-Zuweisungen ist die IANA. Manche Ports sind durch einen offiziellen Standard einem ganz spezifischen Anwendungsprotokoll zugeordnet. Andere gehören nur aus Konvention zu einem

bestimmten Service. Wieder andere sind nicht zugewiesen und werden von Clients dynamisch benutzt. Man unterscheidet:

- *Privilegierte* Ports im Bereich von 0 bis 1023. Viele davon sind durch internationale Standards bestimmten Protokollen zugewiesen. Zugang zu privilegierten Ports erfordert `root`-Rechte.
- *Unprivilegierte* Ports von 1024 bis 65535. Einige von diesen werden von ganz bestimmten Programmen benutzt. Ein Client darf jeden Port aus diesem Bereich für den Aufbau einer Verbindung zu einem Server verwenden.

Portscan

Eine Probe von einigen oder allen Ports eines Computers. Meist werden Ports gescannt, zu denen bekannte Sicherheitslücken existieren.

portmap

Ein RPC-Manager, der für einen Client eine RPC-Service-Nummer in den tatsächlich benutzten Port des jeweiligen Servers übersetzt.

Probe

Senden eines Pakets an einen einzelnen Port eines Computers. Die Probe soll herausfinden, ob der Empfänger in irgendeiner Weise reagiert.

Proxy

Programm, das für ein anderes Programm eine Netzwerkverbindung aufbaut und unterhält. Damit dient es auf Anwendungsebene der Weiterleitung von Daten zwischen Client und Server. Client und Server kommunizieren nicht direkt miteinander. Der Proxy erscheint dem Server als Client, dem Client als Server. Man unterscheidet Proxies auf Anwendungsebene von Circuit-Gateways (siehe auch Bild C.1).

Proxy, Application Gateway (Proxy auf Anwendungsebene)

Ähnlich einem abgeschirmten Host als Firewall, auf System- und auf Anwendungsebene. Nur der Computer mit dem Application-Gateway selbst verfügt über einen direkten Internetzugang. Pakete werden niemals automatisch durch den Computer mit diesem Proxy weitergeleitet. Alle Internetzugriffe erfolgen über den Proxy. Externe wie interne Zugriffe sind nur bis zu diesem Gateway erlaubt. Lokale Benutzer müssen sich entweder auf dem Computer mit dem Application-Gateway einloggen und das Internet von dort aus benutzen, oder sie verbinden sich mit dem Gateway und authentifizieren sich ihm gegenüber.

Die meisten Proxy-Implementierungen bestehen aus separaten Anwendungen für jeden unterstützten Service. Der Proxy-Server versteht das spezifische Kommunikationsprotokoll der jeweiligen Anwendung. Jede Proxy-Anwendung erscheint dem Client-Programm als Server, dem echten Server hingegen als Client. Speziell geschriebene Client-Programme bzw. entsprechend konfigurierte Clients greifen nicht direkt auf den fremden Server zu, sondern auf den Proxy. Dieser stellt dann

für den Client die Verbindung zum eigentlichen Server her, ersetzt dabei aber die Absender-Adresse des Clients durch seine eigene Adresse.

Beispiele für Proxies auf Anwendungsebene sind der Web-Proxy von Apache oder die anwendungsspezifischen Proxies aus dem TIS-Firewall-Toolkit.

Proxy, Circuit-Gateway

siehe Circuit-Gateway-Proxy

Prüfsumme

Diese Zahl entsteht durch einen Algorithmus, der jedes Byte einer Datei oder eines Paketes berücksichtigt. Wenn sich die Datei ändert, oder wenn das Paket bei der Übertragung beschädigt wird, und man erneut eine Prüfsumme berechnet, stimmt sie nicht mehr mit dem Original überein. So lässt sich die Änderung bemerken.

QoS

Quality of Service.

RARP

siehe Reverse Address Resolution Protocol

REJECT

Entscheidung einer Firewallregel, dass ein Paket verworfen wird, wobei der Absender eine ICMP-Fehlermeldung über den Vorgang erhält.

Resolver

Client-Anteil von DNS. Der Resolver ist ein Teil der Systembibliotheken, die mit Netzwerkprogrammen verlinkt werden. Die zugehörige Konfigurationsdatei heißt `/etc/resolv.conf`.

Request for Comment (RFC)

(»Bitte um Kommentar«.) Über die Internet-Society oder die Internet Engineering Task Force IETF publizierte Notiz. Einige RFCs werden zu Standards. Sie behandeln meistens ein Thema, das sich auf das Internet oder die TCP/IP-Protokolle bezieht.

Reverse Address Resolution Protocol**RARP**

Ordnet der Hardware-MAC-Adresse eines Clients die korrekte IP-Adresse zu.

RFC

siehe Request for Comment

RIP

Routing Information Protocol: ein älteres Routing-Protokoll, das aber nach wie vor im Einsatz ist, besonders in großen LANs. Unterstützt wird es beispielsweise von `routed`.

RPC

Remote Procedure Call.

Regel

siehe Firewall und Filter

Runlevel

Ein Konzept zur Beschreibung von Bootvorgängen und Systemzuständen. Der Terminus kommt aus System V UNIX. Normalerweise arbeitet ein System auf einem der Runlevel 2, 3 oder 5. Runlevel 3 ist der normale Systemzustand im Mehrbenutzermodus. Level 2 ist identisch, wobei aber keine NFS-Dienste aktiv sind. Runlevel 5 entspricht 3, wobei zusätzlich Login und Hostauswahl über den Display-Manager von X-Windows möglich sind.

SATAN

Das »Security Administrator Tool for Analyzing Networks« hilft bei der Identifikation von Sicherheitslücken in der Netzwerkkonfiguration.

Segment, TCP

Eine TCP-Nachricht.

setgid

Ein Programm, das bei der Ausführung die Gruppenkennung des Programmeigentümers annimmt und nicht mit der Gruppenkennung des aufrufenden Prozesses läuft.

setuid

Ein Programm, das bei der Ausführung die Benutzerkennung des Programmeigentümers annimmt und nicht mit der Benutzerkennung des aufrufenden Prozesses läuft.

Shell

Ein Befehlsinterpreter unter Unix, z.B. sh, ksh, bash, csh.

Sicherungsschicht

Die zweite Schicht des OSI-Referenzmodells. Zu ihr gehört die Signalübertragung zwischen zwei benachbarten Netzwerkgeräten, z.B. die Übertragung eines Ethernet-Frames von Ihrem eigenen Computer zu Ihrem Router. (Im TCP/IP-Referenzmodell zählt diese Funktionalität noch zur ersten Schicht, der Subnetz-Schicht.)

Simple Mail Transport Protocol (SMTP)

Protokoll für den Austausch von E-Mail zwischen Mail-Servern sowie zwischen Clients und Servern.

Simple Network Management Protocol (SNMP)

Protokoll zur Verwaltung von Netzwerkgeräten über das Netzwerk.

Skript

Ein ausführbarer ASCII-Shellskript, z.B. eine Datei mit sh-, csh-, bash-, ksh- oder perl-Befehlen.

smrsh

sendmail Restricted Shell, eine Shell mit begrenzten Rechten, die speziell für sendmail entwickelt wurde.

SMTP

siehe Simple Mail Transport Protocol

SNMP

siehe Simple Network Management Protocol

Socket

Eindeutiger Ausgangs- oder Zielpunkt einer Netzwerkverbindung, definiert als Paar einer IP-Adresse mit einem bestimmten TCP- oder UDP-Port.

SOCKS

Ein beliebtes Circuit-Gateway-Proxy von NEC.

Spoofing

Fälschung der Absenderadresse eines IP-Pakets.

SSH

Ermöglicht gut authentifizierte und verschlüsselte Netzwerkverbindungen.

SSL

Secure Socket Layer, ein Protokoll für verschlüsselten Datenaustausch. SSL wird am häufigsten von Webservern und -browsern für den Austausch persönlicher Daten im Rahmen von e-Commerce eingesetzt.

statische IP-Adresse

Permanent zugewiesene, fest eingestellte IP-Adresse. Der Begriff ist unabhängig davon, ob die Adresse öffentlich registriert wurde oder einer privaten Adressklasse entstammt.

strobe

Ein Einfacher TCP-Portscanner.

Subnetz

Adressschema, bei dem durch die Verwendung von IP-Adressmasken eine einzelne Netzwerkadresse intern in mehrere verschiedene Netzwerke unterteilt wird. Gegenüber dem Internet wirkt dieser Adressbereich wie ein einzelnes Netzwerk, intern hingegen verfügt man über mehrere Netzwerke oder LANs. Dies erreicht man, indem man einige der Host-Bits der IP-Adresse als Teile der Netzwerk-Bits interpretiert. Z.B. könnte man den Adressbereich 192.168.30.0 – Netmask 255.255.255.0/24 – durch das höchste Bit des vierten Bytes in zwei interne Netz-

werke unterteilen – Netmask 255.255.255.128/25. Man erhält zwei interne Adressbereiche mit jeweils 126 verwendbaren IP-Adressen.

Subnetz-Schicht

Die erste Schicht des TCP/IP-Referenzmodells. Sie repräsentiert das physikalisch für den Austausch elektrischer Signale zwischen benachbarten Netzwerkgeräten benutzte Medium, z.B. Kupferdraht, Fiberglas, Pakettfunk oder Infrarot. Daneben beinhaltet der Begriff auch den logischen Datenaustausch zwischen zwei benachbarten Netzwerkgeräten, z.B. die Übertragung eines Ethernet-Frames von Ihrem Computer zu Ihrem externen Router.

SUNRPC

Port 111. Er wird vom portmap-Dämon verwendet, der eine Anfrage nach einem RPC-Dienst in den tatsächlich vom jeweiligen Server verwendeten Port übersetzt.

SYN

Bitte um Synchronisation, Flag einer TCP-Verbindung. SYN wird in der allerersten Nachricht gesetzt, die ein Programm beim Verbindungsaufbau mit einem anderen Programm sendet.

syslog.conf

Konfigurationsdatei des Protokolldämonen syslogd.

syslogd

Protokolldämon des Systems. Er nimmt über den Systemaufruf syslog() Fehler- und Statusmeldungen von anderen Programmen entgegen.

TCP

siehe Transmission Control Protocol

tcp_wrapper

Autorisierungsschema, das den Zugriff auf lokale Dienste durch fremde Rechner kontrolliert.

tcpd

Programm, das den tcp_wrapper-Dienst realisiert.

TCP/IP-Referenzmodell

Informelles Modell für Netzwerkkommunikation. Es entstand, als TCP/IP Ende der 70er, Anfang der 80er Jahre zum *de facto*-Standard für den Datenaustausch zwischen Unix-Maschinen über das Internet wurde. Das TCP/IP-Referenzmodell ist kein formelles, akademisches Ideal, sondern basiert darauf, was tatsächlich von Herstellern und Entwicklern für das Internet benutzt wurde.

TFTP

siehe Trivial File Transfer Protocol

tiger

Eine Sammlung aus Skripten und C-Programmen. tiger sucht nach Sicherheitslücken, die einem Eindringling root-Zugang ermöglichen könnten.

TOS

Type of Service. Ein Feld im IP-Header, das ursprünglich eine Präferenz bezüglich des Routings andeuten sollte.

traceroute

Bestimmt den Weg, den ein Paket von einem Computer über das Netz zu einem anderen Rechner nimmt.

Transmission Control Protocol (TCP)

Protokoll für verlässliche und fortlaufende Verbindungen zwischen zwei Programmen.

Transportschicht

Die vierte Schicht des OSI-Referenzmodells. Sie bezeichnet die Kommunikation zwischen zwei Programmen, z.B. die Übertragung eines Paketes von einem Client-Programm zum Server. Im TCP/IP-Referenzmodell entspricht dem die dritte Schicht, die ebenfalls »Transportschicht« heißt. Allerdings beinhaltet die Transportschicht des TCP/IP-Modells zusätzlich auch die Aufgaben der Sitzungsschicht von OSI, die für die korrekte Reihenfolge und passende Synchronisation der Nachrichten sorgt.

tripwire

Erstellt und verwaltet eine Datenbank von MD5-Signaturen für die Dateien und Verzeichnisse eines Systems. Damit soll erkannt werden, wenn Dateien unerlaubt ergänzt, gelöscht oder modifiziert werden.

Trivial File Transfer Protocol (TFTP)

Protokoll zum Download eines Bootprogramms in einen Computer oder Router ohne eigenes Bootmedium. Es handelt sich dabei um eine UDP-basierte, vereinfachte Form von FTP.

TTL

Die *Time-to-Live* eines IP-Paketes steht im Header. Sie gibt an, wie oft das Paket auf dem Weg zum Empfänger noch von einem Router zum nächsten weitergegeben werden darf.

UDP

siehe User Datagram Protocol

Unicast

Ein IP-Paket, das von Punkt zu Punkt, von einem Netzwerkinterface zu einem anderen versandt wird.

User Datagram Protocol**UDP**

Protokoll zur Übertragung einzelner Meldungen zwischen Programmen, wobei nicht sichergestellt ist, ob Pakete überhaupt ankommen und ob sie in der richtigen Reihenfolge bleiben.

UUCP

UNIX-to-UNIX Copy Protocol, ein altes Protokoll zur Dateiübertragung zwischen zwei Computern.

Vermittlungsschicht

siehe Netzwerkschicht

WAIS

Wide Area Information Service, heute einfach »Suchmaschine« genannt.

WAREZ

Sammlung von Raubkopien.

WWW

World Wide Web.

X-Windows

Grafisches Benutzerinterface von Unix.

xntpd

NTP-Zeitserver. Stellt Clients die aktuelle Systemzeit zur Verfügung und tauscht Zeitinformationen mit anderen Servern aus.

YaST

siehe control-panel

Zone-Transfer

Übertragung der Host- und Adress-Datenbank eines autoritativen DNS-Servers, die sich auf einen zusammenhängenden Teil einer Domain bezieht, auf einen sekundären Nameserver.

Der Autor

Robert L. Ziegler erhielt von der University of Wisconsin-Madison ein Vordiplom in Psychologie und studierte danach Deutsch und Philosophie bis kurz vor dem Abschluss. Nach einigen Ausflügen im Hinblick auf Ausbildung und Karriere entschloss er sich, sein Hobby zum Beruf zu machen, und erwarb einen Magister in Informatik, ebenfalls an der University of Wisconsin-Madison.

Nach der Uni wurde Bob einer von zwei Unix-Systemprogrammierern in einer Firma, die einen Mini-Supercomputer entwickelte. Neben der Arbeit an Einzelprozessorsystemen erstellte er eine Multiprozessorversion von BSD 4.3. Danach arbeitete er als Unix-Kernelprogrammierer für Forschungsfirmen in der Gegend um Boston.

Als Linux verfügbar und eine Standleitung ins Internet für Privatpersonen bezahlbar wurde, verwirklichte sich ein Traum, der Bob seit 1982 verfolgt hatte: Er konnte zu Hause einen eigenen Unix-Server in einem LAN aufbauen. Seine Bemühungen, dieses System gegenüber dem Internet abzusichern, wurden bald zu einer Leidenschaft. Er bietet über das WWW kostenloses Firewall-Design für Linux an und pflegt eine beliebte FAQ-Liste zu Firewalls und LANs, damit die Leute ihre Linux-Rechner schneller absichern können.

Bob ist heute leitender Ingenieur bei Nokia. Er entwirft und entwickelt Firewalls für die Ipsilon-Produktfamilie.

Stichwortverzeichnis

!

0.0.0.0 99
127.0.0.1 53
255.255.255.255 99
- 401
(400, 403
) 400
.plan 142
.rhost 383, 402
.rhosts 133, 326
/bin/false 372
/etc/dhcpd.conf 367
/etc/ftpaccess 360, 361
/etc/ftpconversions 360
/etc/ftpgroups 360
/etc/ftphosts 360, 361
/etc/ftpusers 361
/etc/group 328, 372
/etc/hosts.allow 75, 133, 216, 218, 329,
361, 497
/etc/hosts.deny 75, 133, 218, 330, 497
/etc/hosts.equiv 326
/etc/inetd.conf 69, 80, 200, 497
/etc/inittab 70
/etc/issue 86, 374
/etc/issue.net 334, 374
/etc/mail/access 336
/etc/mail/relay-domains 336
/etc/named.conf 340, 341, 347, 348, 354,
356
/etc/ntp.conf 368, 369
/etc/ntp/step-tickers 368
/etc/passwd 324, 372
/etc/password 325
/etc/ppp/ip-up 157
/etc/ppp/ip-up.local 157
/etc/pump.conf 159
/etc/rc.d 69, 70

/etc/rc.d/rc.firewall 158, 159
/etc/rc.d/rc.local 86
/etc/rc.d/rc.local 157
/etc/resolv.conf 337, 341, 351, 504
/etc/securetty 327
/etc/security/access.conf 82
/etc/sendmail.cf 337
/etc/services 31
/etc/shadow 325
/etc/ssh2/ssh2_config 335
/etc/ssh2/sshd2_config 335
/etc/ssh_config 335
/etc/sshd_config 335
/etc/sysconfig/network 167
/etc/sysctl.conf 167
/etc/syslog.conf 270, 311, 312, 375, 507
/proc/kmsg 308
/var/log 311
/var/log/maillog 312
/var/log/messages 311
/var/log/secure 312
/var/log/spooler 312
~/rhosts 326

A

abgeschirmter Host 44, 164, 494
abgeschirmtes Subnetz 162, 184, 495
Absender-Port 54
Abtastversuch 55
ACCEPT 489, 502
Accounts, System 373
ACK 37, 38, 39, 55, 66, 67, 91, 108, 489,
496
AF_INET 293
AF_UNIX 293
aktives ftp 135, 222
alert, syslog-Priorität 313
ALL 330

allow-query 349, 356, 359
allow-transfer 349, 357, 359
allow-update 349, 357, 359
amd 70, 75
American Registry for Internet
Numbers 393
anonymes ftp 363
Anwendungsschicht 29
any/0 92
Apache 72, 310
APNIC 393
Application Gateway 44, 489
Application Layer siehe Anwendungs-
schicht
ARIN 393
arpwatch 70
Asia Pacific Network Information
Centre 393
atd 310
ATM 305
auth 85, 120, 200, 286, 288, 294, 489, 497
– auf der Bastion 201
– auf der Choke 202
– syslog-Facility 312
Authentifizierung 324, 489
authpriv, syslog-Facility 312
autobuse 321
autofs 71
automount 70, 71
Autorisierung 327, 489
autoritativ, Nameserver 114

B

BackOrifice 319
Backup-Nameserver 114
Bastion 162, 163, 183, 489, 495
– Firewall 26, 88, 154, 162
Bcast 305
bdf flush 311
benutzerdefinierte Chains 188, 471
Berkeley Internet Name Domain
(BIND) 489

Berkeley Internet Name Domain
Service 337
biff 318
BIND siehe Berkeley Internet Name
Domain
Bitübertragungsschicht 29
Black Hat 20
BOOTP 84, 490
bootparamd 71
BOOTPC 490
bootpd 84, 490
BOOTPS 490
bootps 318
Broadcast 53, 54, 98, 490
– Adressen 189, 367
Brute-Force 325
Buffer Overflow 62
bytes 281, 283

C

Carnegie Mellon University 491
Center for Education and Research in Infor-
mation Assurance and Security 394
CERIAS 394
cfinger 85
CGI siehe Common Gateway Interface
cgiwrap 370
Chains 45, 47, 490
– benutzerdefinierte 471
chargen 81, 317
Checksumme 490
Choke 162, 163, 183, 184, 490, 495
chroot 333, 490
– und ftpd 363
CIAC 394
CIDR 92
Circuit
– Gateway-Proxy 371, 490, 491
Circuit Gateway 44
Class A IP-Adressen, private 52, 97, 188
Class B IP-Adressen, private 52, 97, 188
Class C IP-Adressen, private 52, 97, 188
Class D IP-Adressen 52, 189
Class E IP-Adressen 52, 189

Client/Server-Modell 491
 COAST 394
 COMMAND 310
 Common Gateway Interface (CGI) 491
 – Skripten 72, 370
 Computer Emergency Response Team
 (CERT) 393, 394, 490, 491
 Computer Incident Advisory
 Capability 394
 Computer Oracle and Password
 System siehe COPS
 comsat 83
 control-panel 69, 304, 492
 Coordination 400
 COPS 381, 492
 crack 381, 492
 crit, syslog-Priorität 313
 cron 492
 – syslog-Facility 312
 crond 310, 384, 492
 CU-SeeMe 168, 258
 – Bastion
 DMZ-Seite 260
 extern 258
 – Choke 261
 – Proxy-Modul für Masquerading 258

D

daemon
 – Account 373
 – syslog-Facility 312
 Dämon 30, 492
 DARPA siehe Defense Advanced Research
 Projects Agency
 Darstellungsschicht siehe OSI-Referenz-
 modell
 Data Link Layer siehe Sicherungsschicht
 Datagramm 28
 daytime 81
 debug, syslog-Priorität 313
 Debugging 296
 default-lease-time 367
 Defense Advanced Research Projects Agen-
 cy (DARPA) 492

Demilitarisierte Zone 163, 492
 Denial-of-Service-Angriff 59, 391, 492
 DENY 50, 51, 95, 291, 492, 502
 destination 279, 280, 282, 284
 destination-unreachable 103, 105, 145,
 191, 285, 287
 DES-Verschlüsselung 381
 DHCP 53, 65, 71, 84, 147, 149, 158, 164,
 271, 490, 492
 – Choke als Server 271
 – Nachrichtentypen 147
 – Server, Konfiguration 366
 DHCPACK 148, 271
 dhcpcd 64, 147, 271, 310, 366, 492, 493
 dhcpcd.conf 367
 DHCPDECLINE 148, 271
 DHCPDISCOVER 147, 148, 271
 DHCPINFORM 148
 DHCPNAK 148, 271
 DHCPPOFFER 147, 148, 271
 DHCPRELEASE 148, 271
 DHCPREQUEST 148, 271
 discard 81
 DMZ 163, 183, 493
 DNS 53, 74, 80, 114, 194, 489, 493
 – Anfrage über UDP 114
 – auf der Bastion 196
 – auf der Choke 199
 – Client 116
 – Konfiguration 337
 – Resolver 116
 – Zone-Transfer 114
 DNS-Server
 – auf der Bastion 195
 – auf der Choke 199
 domain 285, 288, 318
 domain-name 367
 domain-name-servers 367
 DoS 59
 Drossel 162, 183
 dtalk 83
 dual-homed 162, 493
 Dynamic Host Configuration Protocol siehe
 DHCP

E

echo 81, 317
 Echoanforderung 61
 echo-reply 103, 106, 192, 502
 echo-request 61, 103, 106, 192, 306, 502
 E-Mail, Zugang durch die Firewall 122
 emerg, syslog-Priorität 313
 Empfänger-Port 55
 err, syslog-Priorität 313
 error, syslog-Priorität 313
 ESTABLISHED 308
 Ethernet 305
 – Frame 28, 493
 exec 82, 318
 externe Sicherheit 18

F

Facilities 312
 – für syslog 312
 Fälschen von IP-Adressen 61, 96, 188
 false 372
 FedCIRC 395
 Federal Computer Incident Response
 Capability 395
 Fehlersuche 275, 296
 File Transfer Protocol (FTP) 109, 493
 Filter 493
 Filtern
 – nach Absender-IP 51, 65
 – nach Absender-Port 54, 66
 – nach Empfänger-IP 54, 65
 – nach Empfänger-Port 55, 66
 – nach TCP-Status-Flags 55, 66
 FIN SENT 308
 FIN WAIT 308
 finger 54, 66, 85, 142, 243, 318, 493
 – auf der Choke 244
 – Server 244
 Firewall 26, 44, 494
 – Initialisierung 93
 – Löschen alter Regeln 94, 187
 – Regeln, Rangfolge 179, 180
 FIRST 394

Flags 307

– Anzeige durch ipchains -L 279, 280, 281,
 283
 – Verbindungszustand 55
 Flooding 496
 Flushing 94
 Flusskontrolle 103, 190
 Fontserver 310
 Foreign Address 294
 Forum of Incident Response and Security
 Teams 394
 forward 179
 forward first 348
 forward only 356
 forward-Chain 289
 forwarden 496
 forwarders 348, 356
 Forwarding 155, 167, 178, 277
 – DNS 117
 fragmentation-needed 103, 104, 105
 Fragmente 32, 64, 496
 Fragmentierung 64, 305
 ftp 72, 81, 135, 136, 222, 294, 317
 – Account 373
 – anonymes 363
 – Firewalling mit SOCKS 371
 – Konfiguration 359
 – syslog-Facility 312
 FTP siehe File Transfer Protocol
 ftpaccess 360, 361
 FTP-Client
 – Bastion 226
 – Choke 224, 231
 ftpconversions 360
 ftpd
 – Option -a 360
 – Option -i 360
 – Option -l 360
 – Option -o 360
 – tcp_wrapper 361
 ftpgroups 360
 ftphosts 360, 361

FTP-Server

- auf der Bastion 223
- auf der Choke 227
- in der DMZ 229
- ftpusers 361

G

- games-Account 373
- gated 62, 72, 501
- Gateway 26
 - SMTP 123
- gdm-Account 373
- gefälschte IP-Adressen 96, 188
- gopher 82, 144, 246, 247
 - Account 373
- group 328, 372

H

- halb-offene Verbindung 37
- Handshake 496
- hosts.allow 75, 133, 216, 218, 329, 361, 497
- hosts.deny 75, 133, 218, 330, 497
- hosts.equiv 133, 326
- HOWTO 497
- HTTP 139, 236, 497
 - Bastion als Server 233
 - Choke als Client 234, 241
 - Server in der DMZ 239
- httpd 72, 310
- HTTPS 232
- Hub 497
- HWaddr 305
- Hypertext Transfer Protocol siehe HTTP

I

- IANA 30, 108, 497
- ICMP 32, 33, 61, 62, 103, 187, 189, 498, 502
 - Konvention für Tabellen 113, 194
 - Masquerading 189, 193, 289
- ICQ 68
- identd 85, 120, 200, 286, 288, 294, 489, 497

- auf der Bastion 201
- auf der Choke 202
- Identität 324
- IDS 380
- ifconfig 304
 - Option -a 304
- ifname 282, 284
- ifstatus 382, 497
- IGMP 100
- IMAP 83, 122, 123, 125, 127, 203, 318, 498
- IMAP-Server 129
 - auf der Bastion 206
 - in der DMZ 212, 213
- in.telnetd 311
- inet 72
- inet addr 305
- inetd 69, 72, 80, 294, 310, 497
- inetd.conf 80, 200, 383, 497
- INET-Domain 307
- info, syslog-Priorität 313
- init 70, 308, 309, 310
- inittab 70
- innd 73, 498
- I-Node 308
- inode 308
- input 178, 179
 - Chain 284
- Instant Messenger 83
- Institute 401
- Internet 400
- Internet Assigned Numbers Authority siehe IANA
- Internet Control Message Protocol siehe ICMP
- Internet Group Management Protocol 100
- Internet Message Access Protocol siehe IMAP
- Internet Protocol 28, 32, 498
- Internet Relay Chat siehe IRC
- Internet-Domain-Sockets 293
- Internet-Schicht 28, 29, 501
- Internet-Worm 491
- Intrusion Detection System 380

IP siehe Internet Protocol
 IP-Adressen 498
 – private 52
 ipchains 19, 88, 90, 93, 188, 410, 498
 – Optimierung 471
 – Option -C 290
 – Option -L 278
 – Option -l 314
 – Option -n 279
 – Option -v 281
 – Regeln anzeigen 278
 IP-Datagramm 498
 IP-Fälschung 61
 IPFW 32, 88, 498
 ipfwadm 19, 188, 431, 498
 – Optimierung 462
 ipmasqadm 181
 ipmasqadm ipportfw 210
 ipmasqadm portfw 209, 249
 IP-Masquerading für das LAN 272
 ipportfw masq 181
 ip-up 157
 ip-up.local 157
 IPX-Netzwerkschicht 73
 IRC 83, 109, 168, 253, 498
 – Bastion
 DMZ-Seite 255
 extern 254
 – Choke 257
 ISP 498
 issue 86, 374
 issue.net 334, 374

K

kern, syslog-Facility 312
 kflushd 309, 310
 Klasse einer Netzwerkadresse 498
 Klasse-A-IPs, private 52, 97, 188
 Klasse-B-IPs, private 52, 97, 188
 Klasse-C-IPs, private 52, 97, 188
 Klasse-D-IPs 52, 189
 Klasse-E-IPs 52, 189
 klogd 78, 308, 310, 499

kmsg 308
 kpiod 309, 310
 ksh 311
 kswapd 309, 310

L

LAN 499
 – Dienste, private 267
 – Zugang, einfacher Weg 154
 link 318
 linuxconf 73, 304, 499
 LISTEN 308
 listen-on 349, 357
 LiveVideo 168
 LOCAL 330
 Local Address 294
 Local Area Network siehe LAN
 localhost 53, 499
 logcheck 321
 login 82, 311, 319
 Loopback 53, 98, 187
 – Interface 499
 – Traffic zulassen 95
 lp-Account 373
 lpd 73
 lpr, syslog-Facility 312

M

mail, syslog-Facility 312
 maillog 311
 Mail-Relaying 336
 Manual-Page 499
 mark 282, 284
 mars-nwe 73
 Mask 305
 Masquerading 155, 165, 167, 168, 178,
 180, 189, 272, 499
 – CU-SeeMe 258
 – IRC 255
 – Module 168
 – QuickTime 251
 – RealAudio 251
 – von internen Servern 181

Maximum Transmission Unit siehe MTU
 max-lease-time 367
 mcserv 74
 MD5 382, 499
 – Hashes 325
 mdrecoveryd 309, 310
 Message Digests 499
 messages 311
 Midnight Commander 74
 mingetty 309, 311
 mount 71, 83, 319
 mountd 319, 376, 380
 mscan 56
 MTU 35, 305, 499
 Multicast 99, 500
 – Adressen 52, 189
 multi-homed 500

N

Nachrichtentyp, ICMP, Konvention für
 Tabellen 113, 194
 Name Switch Cache Daemon 75
 named 74, 116, 194, 308, 310, 340, 500
 named.boot 340
 named.conf 340, 341, 347, 348, 354, 356
 named-bootconf.pl 340
 Nameserver
 – auf der Bastion 195, 196
 – auf der Choke 199
 – autoritativer 114
 – Backup 114
 – öffentliche und private 196
 – primärer 114, 500
 – sekundär 114, 500
 NAT 499, 500
 National Institute of Standards and
 Technology 395
 NCP 73
 netbios-dgm 318
 netbios-ns 318
 netbios-ssn 318
 NetBus 319
 netfs 74

netstat 85, 297, 307, 317, 500
 – Anzeige offener Ports 292
 – Anzeige konventionen 295
 – Option -A 292
 – Option -a 292
 – Option -A unix 307
 – Option -n 292
 – Option -p 292
 NetWare 74
 network 74
 Network Address Translation 499, 500
 Network File System siehe NFS
 Network Information Service siehe NIS
 Network Layer siehe Netzwerkschicht, Ver-
 mittlungsschicht
 Network News Transfer Protocol siehe
 NNTP
 Network Time Protocol siehe NTP
 Network Time Service 266
 Netzwerkschicht 28, 29, 46, 501
 News 129, 214, 318
 news, syslog-Facility 312
 news-Account 373
 News-Server 73
 NFS 69, 70, 71, 74, 75, 111, 164, 319, 500
 nfs 75
 nfsfs 75
 NIRPNET 393
 NIS 71, 75, 79, 164, 326, 500
 NIST 395
 nmap 296, 501
 NNTP 500
 – für die Choke 216
 – Server in der DMZ 214
 nntp 65, 129, 214, 286, 318
 notice, syslog-Priorität 313
 NS 340
 nscan 56
 nsd 75
 ntalk 83, 319
 NTP 34, 35, 150, 164, 266, 368, 501
 – Client und Server auf der Choke 268
 – Konfiguration 368

- Server auf der Bastion 267
- Server, öffentliche 369
- ntp 318
- ntp.conf 368, 369
- ntpd 78
- ntpddate 78, 150, 269, 368, 369, 501

O

- Open Shortest Path First 501
- Open Systems Interconnection siehe OSI
- Open Windows 109
- openwin 319
- operator-Account 373
- opt 279, 280, 281, 283
- Optimierung der Firewall-Regeln 451
- OSI
 - Referenzmodell 27, 501
- OSPF 72, 501
- output 178, 179
- Chain 286
- outsize 282, 284

P

- Packet marking 282
- Paket 28, 32, 501
- Paketfilter 43, 44, 45, 496, 501
- panic, syslog-Priorität 313
- parameter-problem 103, 104, 190, 191, 285, 287
- PARANOID 330
- passives ftp 135, 222
- passwd 324, 372
- password 325
- PATH 502
 - Variable 373
- Path 308
- PC Anywhere 319
- pcanywhere 319
- Peer-to-Peer 502
 - DNS 117
- Perimeter 163, 184
- Netzwerk 183
- Physical Layer siehe Bitübertragungsschicht

- physikalische Schicht 502
- PID 309, 502
- PID/Program name 294, 308
- ping 61, 103, 106, 189, 192, 305, 319, 502
- Ping of Death 61
- ping-Flooding 61
- pkts 281, 283
- plan 142
- Platzhalter für tcp_wrapper 330
- Policies 47, 88, 91, 95, 157, 187, 502
- POP 53, 122, 123, 125, 126, 203, 502
- pop-2 83
- POP3 65
- pop-3 83, 294, 318
- popd 294, 365
- POP-Server 128
 - auf der Bastion 205
 - in der DMZ 211, 212
 - Konfiguration 365
- Port-Forwarding 182
- portmap 71, 74, 75, 108, 327, 331, 503, 507
- Ports 29, 30, 502
 - häufig gescannte 317
 - privilegierte 30, 503
 - unprivilegierte 30, 503
 - schützen 108
- ports 279, 280, 282, 284
- Portscan 56, 503
 - gezielt 57
 - komplett 56
- Portumleitung 181
- port-unreachable 104, 105
- Post Office Protocol siehe POP
- postgres-Account 373
- postgresql 75

PPP

- Firewall-Installation 157
- Presentation Layer siehe Darstellungsschicht
- primär, Nameserver 114
- printer 319
- Prioritäten für syslog-Meldungen 313

private Dienste 267
private IP-Adressen 52, 97, 188
privilegierte Ports 30, 503
Probe 503
Promiscuous Mode 70, 382
prot 279, 280, 281, 283
Proto 293, 307
Protokolle, Anzeige durch ipchains -L 279,
280, 281, 283
Protokollierung auf andere Rechner 277,
375
Proxy 141, 162, 164, 232, 503
– Anwendungsebene 491, 503
– Circuit-Gateway 490, 491
– Firewall 44, 371
– für ankommende Verbindungen 182
– transparent 181, 371
Proxy für HTTP 238
– auf der Choke 242
– Bastion als Server 234
– Choke als Client 235, 242
– Server in der DMZ 240
Prozess-ID 309
Prüfsumme 504
ps 297
– Option a 308
– Option u 309
– Option x 308
ps ax 308
PTR 340
pump 147, 159
pump.conf 159
pwconv 325

Q

QoS 282, 504
Quake 168, 262
– Bastion
DMZ-Seite 264
extern 263
– Choke als Client 266
– Server auf der Bastion 265

Quality of Service 282, 504
QuickTime siehe RealAudio

R

RAID 310
range 367
Rangfolge von Firewall-Regeln 179, 180
RARP 84, 490, 504
rc 70
rc.firewall 158, 159
rc.local 86
rc.local 157
rc.sysinit 70
rcp 164
Real Time Streaming Protocol siehe RTSP
Real Time Transport Protocol siehe RTTP
RealAudio 68, 168, 249, 371
– Bastion
DMZ-Seite 251
extern 249
– Choke 252
Recv-Q 293
Redirect 62
redirect 103, 319
RefCnt 307
REJECT 50, 51, 95, 291, 502, 504
– bei auth/identd 121
Relay 123
relay-domains 336
Relaying 336
Remote Procedure Call siehe RPC
Request for Comment siehe RFC
Réseaux IP Européens 393
resolv.conf 341, 351, 504
Resolver 74, 116, 337, 504
Reverse Address Resolution Protocol siehe
RARP
Reverse-Lookup 329
rexec 82
RFC 504
rhost-Authentifizierung 326
rhosts 133
RIP 72, 76, 504

- RIPE 393
- R-Kommandos 164
- rlogin 82, 164, 326
- rlytest 336
- root-Account 327
- route 319
- routed 62, 64, 72, 76, 504
- router 367
- Routing Information Protocol siehe RIP
- RPC 68, 71, 74, 75, 505
- und tcp_wrapper 331
- rpc.rusersd 76
- rpc.rwalld 76
- rpcbind 331
- rsh 82, 164, 326
- rstatd 76
- RTSP 249
- RTt 249
- Runlevel 69, 505
- Manager 69
- runtime 76, 164
- ruserd 76
- rwalld 76
- rwho 76, 164
- rwhod 64, 76

- S**
- Samba 74, 77
- SATAN 56, 382, 505
- Scan 56
- gezielt 57
- komplett 56
- secure 311
- Secure Shell siehe SSH
- Secure Socket Layer siehe SSL
- securetty 327
- Security 401
- security, syslog-Facility 312
- Segment 28, 505
- sekundär, Nameserver 114
- sendmail 77, 124, 125, 127, 128, 203, 207, 308, 311, 335
- Send-Q 293
- Sequenznummern 37
- Server 30
- Service not available 33
- Service Or Host Not Available 34
- services 31
- Session Layer siehe Sitzungsschicht
- setgid 332, 505
- setuid 332, 505
- shadow 325
- Shadow-Passwörter 324
- Shell 505
- shell 82, 319
- Sicherungsschicht 29, 492, 505
- Simple Mail Transport Protocol siehe SMTP
- Simple Network Management Protocol
siehe SNMP
- Sitzungsschicht 508
- auch OSI-Referenzmodell 29
- smb 77
- smrsh 337, 506
- SMTP 77, 122, 123, 125, 126, 127, 128, 203, 505
- Clients, Konfiguration der Choke 204
- Gateway 123
- Konfiguration 335
- Server in der DMZ 208
- smtp 286, 288, 318
- SMTP-Server
- auf der Bastion 204
- in der DMZ 207, 208
- smurf 107, 108
- SNMP 77, 164
- snmp 318
- snmpd 77
- Socket 36, 293, 307, 506
- Sockettypen 293
- SOCKS 30, 168, 371, 491, 506
- Bezugsquelle 372
- Server 110
- socks 319
- source 279, 280, 282, 284
- source-querch 103, 104, 190, 285, 287
- Source-Routing 63

Spoofing 506
 spooler 311
 SPX-Transportschicht 73
 SQL 319
 Squid 78
 squid-Account 373
 sscan 56, 57, 317, 319
 SSH 506
 ssh 54, 66, 81, 131, 133, 134, 218, 296, 317
 – Bezugsquellen 334
 – Installation 335
 – Konfiguration 334
 ssh_config 335
 ssh-Client
 – Bastion 220
 – Choke 220
 sshd 86, 308, 311
 sshd_config 335
 ssh-Server
 – Bastion 219
 – Choke 221
 SSL 139, 140, 232, 237, 506
 – Bastion als Server 233
 – Choke als Client 235, 241
 – Server in der DMZ 239
 STAT 310
 State 294, 308
 stateless 34, 61
 statische IP-Adresse 506
 step-tickers 368
 strobe 56, 295, 506
 su 312
 – einschränken 328
 subnet 367
 – mask 367
 Subnetz 169, 170, 506
 – Schicht 28, 29, 507
 SUNRPC 507
 sunrpc 83, 318
 swatch 322
 SYN 36, 37, 55, 66, 91, 108, 496, 507
 SYN RECV 308
 SYN SENT 308
 SYN-Cookies 61, 96

SYN-Flooding 60, 61
 sysctl.conf 167
 syslog 78, 269, 270, 319, 384
 – Facilities 312
 – Konfiguration 312
 – Prioritäten 313
 – syslog-Facility 312
 syslog.conf 270, 311, 312, 375, 507
 syslogd 64, 78, 308, 310, 311, 507
 – Option -r 375
 systat 85, 317
 Systemaccounts 373

T

talk 82, 83, 319
 target 279, 280, 281, 283
 TCP 28, 32, 35, 36, 37, 38, 39, 68, 508
 – Dienste auf unprivilegierten Ports 109
 – SYN-Flooding 60
 TCP/IP 46
 – Referenzmodell 27, 507
 tcp_wrapper 81, 294, 328, 497, 507
 – Platzhalter 330
 – und ftpd 361
 tcpd 328, 507
 telnet 54, 66, 72, 81, 131, 132, 216, 294, 318
 – Konfiguration 334
 telnetd 334
 Testpakete für ipchains 290
 TFTP 508
 tftp 84, 139, 490
 tftpd 318
 tiger 383, 508
 TIME 310
 time 81
 timed 64, 164
 time-exceeded 103, 105, 145, 191, 192, 285, 287
 Timeout 191
 Time-to-Live siehe TTL
 Todesping 61
 Token Ring 305
 TOS 281, 282, 283, 508

tosa 281, 283
 tosx 281, 283
 traceroute 103, 105, 145, 146, 189, 191,
 319, 508
 Transmission Control Protocol siehe TCP
 transparenter Proxy 181, 371
 Transport Layer siehe Transportschicht
 Transportschicht 28, 29, 46, 508
 tripwire 383, 508
 Trivial File Transfer Protocol siehe TFTP
 TTL 103, 191, 339, 508
 TTY 309
 Tunnel, ssh 133
 Type 307
 Type-of-Service 281, 282, 283, 508

U

UDP 28, 32, 34, 35, 61, 68, 145, 258, 509
 – Dienste auf unprivilegierten Ports 111
 – Firewalling mit SOCKS 371
 – Flooding 61
 – Weiterleitung 371
 Umleitung, ICMP 62
 Unicast 508
 UNIX-Domain 307
 – Sockets 293
 unprivilegierte Ports 30, 503
 – schützen 108
 update 311
 Usenet 73, 129, 214
 usepeerdns 158
 User Datagram Protocol siehe UDP
 user, syslog-Facility 312
 UUCP 509
 uucp 83, 312, 319
 – Account 373
 – syslog-Facility 312

V

vacation 337
 Vermittlungsschicht 28, 29, 501
 Virtual Private Network(VPN) 133

W

WAIS 145, 247, 248, 509
 wall 64
 WAREZ 392, 509
 warn, syslog-Priorität 313
 warning, syslog-Priorität 313
 Web-Proxy 141, 238
 – auf der Choke 242
 – Bastion als Server 234
 – Choke als Client 235, 242
 – Server in der DMZ 240
 wheel 328
 who 319
 whois 143, 144, 245, 246
 Wide Area Information Server siehe WAIS
 World Wide Web (WWW) 139, 232

X

X11 319
 xdm 69
 xdmcp 318
 xfs 78, 308, 310
 xfs-Account 373
 xntpd 78, 150, 164, 266, 268, 368, 369,
 501, 509
 xterm 93
 X Window 110
 – bei der Fehlersuche 276
 – Server 30

Y

YaST 304, 509
 ypbind 79
 yppasswdd 79
 ypserv 79

Z

Zone 338
 – demilitarisierte 492
 – Transfer 114, 509