
millin

Wolfgang Barth

Das Firewall-Buch

Grundlagen, Aufbau und Betrieb sicherer Netzwerke
mit Linux

3., aktualisierte und erweiterte Auflage

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über `http://dnb.ddb.de` abrufbar.
ISBN 3-89990-128-2

© 2004 Nicolaus Millin Verlag GmbH, Poing (http://www.millin.de)
Umschlaggestaltung: Fritz Design GmbH, Erlangen
Gesamtlektorat: Nicolaus Millin
Fachlektorat: Peter Albrecht, Dieter Bloms, Anke Boernig, Roman Drahtmüller, Matthias Eckermann, Michael Hasenstein, Fabian Herschel, Marc Heuse, Lars Knoke, Björn Lotz, Wolfgang Rosenauer, Sascha Wessels, Stefan Werden
Satz: LaTeX
Druck: Kösel, Krugzell
Printed in Germany on acid free paper.

Inhaltsverzeichnis

Vorwort zur dritten Auflage **1**

1 Ziel dieses Buches **3**

2 Wozu braucht man Firewalls? **7**
2.1 Der Begriff „Firewall“ 7
2.2 Was ein Firewall kann … 8
2.3 … und was ein Firewall nicht kann 9
2.4 Grundwerte der IT-Sicherheit 10
2.4.1 Vertraulichkeit 10
2.4.2 Verfügbarkeit 11
2.4.3 Integrität 12
2.5 Zusammenfassung 12

3 Security Policy **13**
3.1 Security Policy und Firewalls 14
3.2 Gefährdungpotentiale 14
3.2.1 Externe, aktive Angriffe 15
3.2.2 Interne, aktive Angriffe 17
3.2.3 Autonome Einheiten 18
3.3 Sicherheitspolitik 21
3.3.1 Sicherheitsziele 21
3.4 Sicherheitskonzept 26
3.4.1 Organisatorisches 26
3.4.2 Infrastruktur 29
3.4.3 Einzelne Systeme 29
3.4.4 Netzwerke 31
3.5 Zusammenfassung 33

4 Grundlagen des Firewalldesigns **35**
4.1 TCP/IP – Netzwerkprotokoll für das Internet . . . 36

4.1.1 Das OSI-Referenzmodell 36
4.1.2 Historische Entwicklung 38
4.1.3 DoD-Protokollfamilie (TCP/IP) 39
4.1.4 Firewallsysteme und OSI-Ebenen 40
4.1.5 IP 40
4.1.6 TCP 42
4.1.7 UDP 46
4.1.8 ICMP 47
4.2 Paketfilterung 48
4.2.1 Statische Paketfilterung 49
4.2.2 Dynamische Paketfilterung 50
4.3 Proxy-Systeme 51
4.4 Firewallumgebungen 53
4.4.1 Entmilitarisierte Zone: das Grenznetz 53
4.4.2 Bastion Hosts 54
4.4.3 Dual Homed Hosts 56
4.5 Private IP-Adressen 57
4.5.1 Proxies 59
4.5.2 Network Address Translation/Masquerading . . 60
4.5.3 Transparente Proxies 61
4.6 Zusammenfassung 62

5 Paketfilterung und Network Address Translation 63
5.1 Kernel vorbereiten 64
5.1.1 Kernel 2.4 64
5.1.2 Kernel 2.6 66
5.1.3 Modularer oder statischer Kernel? 68
5.2 Paketfilterung mit `iptables` 69
5.2.1 Ablaufdiagramm des Linux-Kernelfilters . . . 70
5.2.2 Grundoperationen für Filterregeln 71
5.2.3 Matching-Optionen für Filterregeln 71
5.2.4 Targets – Was soll die Filterregel bewirken? . . 81
5.2.5 Default Policy 87
5.2.6 Regelketten anzeigen, entleeren, überwachen . . 88
5.2.7 Weitere Variationen von Filterregeln 90
5.2.8 Filterregeln testen 92
5.2.9 Filterregeln sichern/restaurieren 92
5.3 Network Address Translation 92
5.3.1 Einsatz von NAT: Schema 93
5.3.2 Source-NAT 95
5.3.3 Destination-NAT 96
5.4 Netfilter Erweiterungen: Patch-o-Matic 97
5.5 Zusammenfassung 101

6 Proxies **103**
6.1 HTTP-Proxy: Squid 107
6.1.1 Basis-Konfiguration 110
6.1.2 Inbetriebnahme 113
6.1.3 Access Control Lists 118
6.1.4 Authentifikation 120
6.1.5 Transparenter Proxy 122
6.1.6 Redirector 124
6.2 Filterung für Squid: SquidGuard 125
6.2.1 Konfiguration von SquidGuard 127
6.2.2 Anpassung von Squid 134
6.2.3 Pflege der Sperrlisten 135
6.3 Viren scannen am Proxy 136
6.3.1 Apache::ProxyScan 141
6.3.2 Apache2 Modul: `mod_clamav` 144
6.4 SuSE Proxy Suite: `ftp-proxy` 149
6.4.1 Inbound-Connections 150
6.4.2 Startarten für den `ftp-proxy` 158
6.4.3 Outbound-Connections 159
6.4.4 Transparenter FTP-Proxy 159
6.5 SOCKS 160
6.5.1 SOCKS V4 im Vergleich zu SOCKS V5 161
6.5.2 Implementierungen von SOCKS 162
6.5.3 Dante SOCKS-Server 163
6.5.4 Clients mit SOCKS (Socksifying) 170
6.6 TIS Firewall Toolkit 174
6.6.1 Bestandteile des TIS-FWTK 175
6.7 Ausblick 176

7 Internetdienste und Firewalls **179**
7.1 Häufig benutzte Internetdienste 180
7.1.1 E-Mail: SMTP/POP3 180
7.1.2 Domain Name Service (DNS) 188
7.1.3 Terminal-Session: Telnet 196
7.1.4 r-Kommandos 198
7.1.5 Secure Shell: sicherer Ersatz für `rsh/telnet` . . 201
7.1.6 File Transfer Protocol (FTP) 207
7.1.7 World Wide Web: HTTP 217
7.1.8 HTTP via SSL/TLS 226
7.1.9 `ping` und weitere ICMP-Messages 229
7.1.10 Usenet-News: NNTP 234
7.1.11 Zeitsynchronisierung im Netzwerk: NTP . . . 238
7.1.12 Finger 240

7.1.13 RPC-Kandidaten: NFS, NIS/YP 240
7.1.14 X Window System 242
7.2 Internetdienste starten 243
7.2.1 Start als permanenter Daemon 243
7.2.2 Start über den `inetd`-Daemon 245
7.3 Internetdienste abschalten 246
7.3.1 Start über `/etc/init.d` 246
7.3.2 Start über `inetd` 247
7.4 Internetdienste kontrollieren 247
7.4.1 Der `tcpwrapper` 248

8 Firewalls bauen – einzelner Rechner **251**
8.1 Absicherung eines einzelnen Rechners 251
8.1.1 Benutzerverwaltung/Login 252
8.1.2 Unnötige Dienste abschalten 254
8.1.3 Sichere Konfiguration verbleibender Dienste . . 256
8.1.4 Kontrolle über offene Ports 257
8.2 Einzelner Rechner mit direktem Internetzugang . . . 259
8.2.1 Überflüssige Dienste abschalten 260
8.2.2 Paketfilterung mit fester IP-Adresse 263
8.2.3 Skript starten und gesetzte Regeln protokollieren . 280
8.2.4 Auswertung der Protokolle 284
8.2.5 Dynamische IP-Adressen 289
8.2.6 Variante für dynamische Filterregeln 297

9 Firewalls bauen – Netzwerk **301**
9.1 Internetzugang über Router/Paketfilter 301
9.2 Spezialbehandlung für DSL 314
9.3 Einfacher Firewall 315
9.3.1 Paketfilterung 319
9.3.2 Proxy-Konfiguration (SOCKS) 336
9.3.3 Caching-Only Nameserver 340
9.3.4 Feste IP-Adresse 345
9.4 Firewall mit Grenznetz 345
9.4.1 Paketfilterung 347
9.4.2 Caching Web-Proxy 360
9.4.3 Webserver 363
9.4.4 FTP-Server 365
9.4.5 Transparente Proxies 371
9.5 Firewall mit Dual Homed Bastion Host 372
9.5.1 Paketfilterung für „proxy“ 374
9.5.2 Paketfilterung für „gate“ 383
9.5.3 Dreistufiger Firewall 389

9.5.4 Parallele Dual Homed Bastion Hosts 389
9.6 Zusammenfassung 391

10 Maintenance: Firewalls installieren, betreiben, überwachen 393
10.1 Vor der Inbetriebnahme 394
10.1.1 Physikalische Sicherheit 394
10.1.2 Partitionierung/Mounts 396
10.1.3 Auswahl der zu installierenden Pakete . . . 399
10.1.4 Planung von regelmäßigen Updates 403
10.1.5 Datensicherung, Worst-Case-Recovery . . . 405
10.1.6 Plazierung des Firewallskriptes 406
10.2 Regelmäßige Wartung 409
10.3 Intrusion Dectection 411
10.3.1 Einführung 411
10.3.2 Snort 414
10.3.3 Informationsquellen zum Thema IDS 423
10.4 Integritätsprüfung auf Dateiebene 423
10.4.1 Bordmittel: RPM 424
10.4.2 Tripwire 426
10.4.3 weitere Tools 434
10.5 Logfile-Überwachung 435
10.5.1 syslog 437
10.5.2 `logcheck` 442
10.5.3 `logsurfer` 447
10.6 Netzwerkanalyse 462
10.6.1 `tcpdump` 464
10.6.2 `ngrep` 469
10.6.3 Portscanning: `nmap` 473
10.6.4 Nessus 479
10.7 Port-Überwachung 485
10.7.1 `scanlogd` 485
10.7.2 PortSentry 486
10.8 Was tun, wenn es doch passiert ist... 487
10.9 Zusammenfassung 490

11 Virtuelle Private Netzwerke (VPN) 491
11.1 IPsec-Grundlagen 492
11.1.1 Sicherheitsprotokolle 493
11.1.2 Transport- und Tunnelmode 494
11.1.3 Authentifikation, automatischer Schlüsselaustausch . 495
11.2 IPsec unter Linux 497
11.2.1 Kernel 2.4 oder 2.6? 497
11.2.2 FreeS/WAN, strongSwan, Openswan 498

11.3 Installation von strongSwan 499
11.4 Einsatzszenarien 503
11.5 Konfiguration von strongSwan 505
11.5.1 Erzeugen von RSA-Schlüsseln für die Authentifikation 505
11.5.2 `/etc/ipsec.conf` 507
11.5.3 LAN-to-LAN-Kopplung 508
11.5.4 Initiierung der Verbindung 512
11.5.5 Router zwischen den beiden IPsec-Gateways . . 517
11.5.6 Road Warrior 518
11.5.7 Zwei Gateways mit dynamischer IP-Adresse . . 520
11.6 Opportunistic Encryption OE 525
11.7 Policy Groups 529
11.8 X.509-Zertifikate 533
11.8.1 Erzeugung eigener Zertifikate 535
11.8.2 Konfiguration von strongSwan 543
11.8.3 IPsec Policies über Zertifikate 546
11.9 Network Address Translation und IPsec 547
11.9.1 NAT vor dem IPsec-Tunnel 547
11.9.2 NAT auf dem IPsec-Gateway 548
11.9.3 NAT zwischen den beiden IPsec-Gateways . . 549
11.10VPN-Gateways und Firewalling 550
11.10.1 Filterung auf einem Firewall zwischen VPN-Gateways 551
11.10.2 Filterung auf dem VPN-Gateway 551
11.10.2.1 Kernel 2.4 551
11.10.3 VPN-Gateway im dreistufigen Firewall . . . 555

12 VPN-Gateway für Windows-Clients 557
12.1 Mit Windows ans Internet: Safety first! 559
12.2 Import der Zertifikate unter Windows 561
12.3 Natives IPsec 568
12.3.1 strongSwan-Konfiguration 568
12.3.2 Dynamische IP-Adresse am Gateway 571
12.3.3 Konfiguration des Windows-Clients 572
12.4 IPsec/L2TP 577
12.4.1 L2TP-Protokollstack 578
12.4.2 Konfiguration des VPN-Gateways 579
12.4.3 Windows-Client: private Verbindung über VPN . 584
12.4.4 Troubleshooting 589

A Dynamisches DNS für das Internet-Gateway 593

B TCP/IP: Ausgewählte Kapitel 597
B.1 IP 597

B.2 TCP 598
B.3 UDP 600
B.4 ICMP 600
B.5 IP-Fragmentierung 602
B.6 ausgewählte TCP/UDP-Portnummern 602

C Runtime-Kernelkonfiguration **607**
C.1 IP Forwarding 607
C.2 IP Spoof Protection 608
C.3 TCP SYN-Cookies 608
C.4 ICMP Options 608
C.5 Weitere, Device-abhängige Kernelparameter 609

D Die Secure Shell: SSH **611**
D.1 Serverinstallation und -konfiguration 612
D.1.1 Hostkey erzeugen 612
D.1.2 Starten des `sshd` 613
D.1.3 Systemweite Konfigurationsdateien 613
D.2 User-Konfiguration 615
D.2.1 Erzeugen des DSA/RSA-Schlüssels 615
D.2.2 Schlüsselverteilung, Logins einrichten 616
D.2.3 Userkonfigurierbare Dateien 617
D.3 User-Authentifikation mit SSH 619
D.3.1 SSH-Agent 621

E Unterschiede zwischen `iptables` und `ipchains` **625**
E.1 Paket-Routing: Unterschiede 626
E.2 Änderungen von `ipchains` auf `iptables` 629

F Informationsquelle Internet **633**
F.1 SUSE LINUX 633
F.2 Debian 634
F.3 Security-Links 634
F.3.1 Allgemeine Security-Links 634
F.3.2 Linux Security-Links 635
F.3.3 Intrusion Detection 635
F.4 Verschiedenes 636
F.4.1 Linux Online 636
F.4.2 Linux und ISDN 636
F.4.3 Homepage des Autors 636

Vorwort zur dritten Auflage

Kernel 2.4 oder 2.6?

Diese Frage stellt sich immer öfter, je mehr sich Kernel 2.6 zu einem stabilen Standard-Kernel entwickelt. Ob man die Kernelversion 2.6 auch im Firewallbereich einsetzt, hängt von verschiedenen Fragestellungen ab.

Während die 2.4er Linie weitestgehend ausgereift ist und stabil läuft, erfährt der Kernel 2.6 heute (Sommer 2004) noch permanente Änderungen. Auf Rechnern mit sehr hoher Last und vielen gleichzeitig laufenden Prozessen hat die 2.6er Serie deutliche Vorteile aufgrund des stark verbesserten Schedulings. Am ehesten macht sich dieser Vorteil bemerkbar, wenn User interaktiv auf dem System arbeiten. Für einen Firewall dagegen macht das nur wenig Unterschied, hier ist es wichtig, daß der Kernel möglichst stabil läuft.

Für die Paketfilterung kommt bei beiden Kernelversionen das Netfilter-Framework zum Einsatz, das der Administrator über das Programm `iptables` konfiguriert und bedient. Beim Einsatz von Proxies spielt die Kernelversion ebenfalls keine Rolle. Solange man nicht irgendwelche designbedingte Grenzen des 2.4er sprengen will (maximal 4 Prozessoren, Device-Limits im Kernel wie maximal 64 PPP-Interfaces etc.), kann man also genauso gut bei Version 2.4 bleiben.

Beim Einsatz von IPsec allerdings gibt es wesentliche Unterschiede. Hier hängt es davon ab, was man genau erreichen will. Für den Kernel 2.6 gibt es im Zusammenhang mit IPsec derzeit einige Einschränkungen. Im Abschnitt 11.2 wird auf die Unterschiede näher eingegangen. Es gibt derzeit verschiedene Bemühungen, bestimmte Features bei Einsatz von Kernel 2.6 wieder auf das Niveau von Kernel 2.4 zu heben, das Ende ist hier noch nicht abzusehen.

Kurz zusammengefaßt könnte man sagen: wer kein IPsec einsetzt, kann im Prinzip frei zwischen 2.4 und 2.6 wählen. Beim Einsatz von IPsec ist derzeit auch weiterhin der Einsatz von 2.4 in vielen Fällen ratsam, bis der Ausgang der Entwicklungen um IPsec mit Kernel 2.6 abzusehen ist.

Was ist neu in der dritten Auflage?

In Kapitel 5 wurde der Abschnitt „Kernel vorbereiten“ der Weiterentwicklung des Kernel entsprechend um Kernel 2.6 ergänzt. Das Proxy-Kapitel 6 wurde um das Thema „Viren scannen beim Websurfen“ erweitert, ein Thema, dem in letzter Zeit eine immer größere Bedeutung beigemessen wird.

Bedingt durch die Einstellung der Weiterentwicklung von FreeS/WAN wurde Kapitel 11 überarbeitet. Beschrieben wird jetzt die Konfiguration mit dem FreeS/WAN-Nachfolger strongSwan. Dabei wurde das Thema VPN-Gateways mit dynamischen IPs ergänzt und das leidige Thema Zwangstrennung der Verbindung durch den Provider berücksichtigt.

Hinzugekommen ist ein neues Kapitel 12 zur Anbindung von Windows-Clients über IPsec an ein Linux-Gateway. Hier werden zwei Alternativen Im Detail beschrieben, einmal die Anbindung über natives IPsec, zum anderen die von Microsoft favorisierte Variante IPsec/L2TP.

Informationsquellen

Zu diesem Buch gibt es auch eine Mailingliste, die Sie bei Fragen zu allen Themen, die in diesem Buch vorkommen, nutzen sollten:

`http://listi.jpberlin.de/mailman/listinfo/firewallbuch`

Weitere Informationen zum Buch, Ergänzungen und Fehlerkorrekturen finden Sie auf der Homepage des Autors:

`http://linux.swobspace.net`

Linz, im Oktober 2004 *Wolfgang Barth*

Kapitel 1

Ziel dieses Buches

Dieses Buch beschäftigt sich mit dem Bau eines Firewalls mit frei (hier im Sinne von „kostenlos“) verfügbaren Mitteln aus dem Internet. Betriebssystem-Plattform ist das ebenfalls frei verfügbare, im Source-Code erhältliche und gut dokumentierte Unix-Derivat Linux.

Ziel dieses Buches ist, für Sicherheitsfragen zu sensibilisieren, Firewalls als Bestandteil von Sicherheitsinfrastrukturen kennenzulernen und die Funktionsweise von Firewalls zu verstehen. Darüber hinaus sollten Sie nach der Lektüre in der Lage sein, eine eigene Sicherheitspolitik zu definieren und einen dazu passenden Firewall mit freien Werkzeugen aufzubauen und zu betreiben.

Das Thema „Selbstbau“ bei Firewalls wird heftig diskutiert. Zum einen wird die Auffassung vertreten, daß bei „Open-Source“- oder „Public-Domain“-Programmen niemand die Verantwortung für Programmfehler trage und die Herkunft der eingesetzten Software oft zweifelhaft sei (zum Beispiel [Poh98]). Andere halten dagegen, daß bei kommerziellen Systemen niemand außer dem Hersteller die Funktionalität und Anfälligkeit des gekauften Firewalls kenne. Ein kommerzieller Firewall kostet zudem Geld und kommt daher kaum für private Anwendungen in Betracht. Bei hohen Ansprüchen an eine sichere Umgebung übersteigt der Aufwand für die Realisierung eines kommerziellen Systems zudem leicht den Aufwand für den Selbstbau ([CZ96]).

Die Entscheidung „kaufen oder selbst bauen“ kann Ihnen niemand abnehmen. Der Autor ist jedoch der festen Überzeugung, daß Sie eine sachgerechte Entscheidung erst dann treffen können, wenn Sie sich intensiv mit dem befaßt haben, was sich hinter dem kurzen Stichwort „Firewall“ verbirgt. Im übrigen läßt sich der Aufwand, den ein Selbstbau mit sich bringt, ohnehin am besten abschätzen, indem man es einfach einmal versucht: learning by doing.

Um überprüfen zu können, ob ein Firewall Ihren Anforderungen genügt, müssen Sie zunächst diese Anforderungen definieren. Auch dabei hilft Ihnen dieses

Buch. Ein Kapitel widmet sich ausschließlich dem Thema „Sicherheitspolitik“. Wenn Sie Ihre Anforderungen klar bestimmt haben, müssen Sie zudem einschätzen können, ob ein Firewall (sei dieser nun kommerziell oder nicht) diese Anforderungen auch erfüllt. Hier ist ein vertieftes Verständnis für die Funktionalität von Firewalls unbedingt notwendig.

Wenn Sie sich am Ende dieses Buches dennoch für ein kommerzielles Firewallsystem entscheiden, dann auch, weil Ihnen dieses Buch eine Fülle von Informationen an die Hand gibt, die eine solche Entscheidung erleichtern. Damit wäre dann auch ein Ziel dieses Buches erreicht: „Firewalls verstehen lernen“.

Wenn Sie sich aber intensiv mit den vorgestellten Varianten auseinandergesetzt und diese in der Praxis umgesetzt haben, werden Sie vermutlich feststellen, daß das entworfene System Ihren Sicherheitsansprüchen genügt und damit die zusätzliche Auseinandersetzung mit kommerziellen Systemen überflüssig sein dürfte.

Möglicherweise gehören Sie jedoch zu der Gruppe von Anwendern, die beim Einsatz freier (nun im Sinne von frei verfügbar und einsetzbar) Software aus dem Internet ein „ungutes Gefühl“ hat. Hier sei daher eine Lanze für freie Software gebrochen: Freie Software ist im Quellcode erhältlich und wird in der Regel von einer großen Schar freiwilliger Entwickler gepflegt. Die Verfügbarkeit des Quellcodes führt dazu, daß Fehler schneller erkannt und behoben werden als bei kommerziellen Systemen, und die rasche Fehlerbeseitigung gerade bei sicherheitsrelevanter Software ist allemal besser als das Totschweigen kaum bekannter Bugs. Auch wenn sich kommerzielle Hersteller immer häufiger bemühen, Öffentlichkeit in Bezug auf vorhandene Probleme herzustellen, und eine Fülle von Patches liefern, ist das keine Garantie dafür, daß Sicherheitslücken nicht wochen- oder monatelang in Hackerkreisen kursieren, bevor das Problem überhaupt entdeckt wird. Bei freier Verfügbarkeit des Quellcodes ist die Wahrscheinlichkeit unentdeckter Lücken wesentlich geringer. Über mangelnden Support kann sich bei freier Software ebenfalls niemand beklagen. Es ist durchaus üblich, daß für freie Software innerhalb weniger Stunden oder Tage ein Patch verfügbar ist (einschließlich Problemlösungen im Linux-Kernel für neue Angriffsvarianten), während bei kommerziellen Systemen weitaus mehr Zeit vergeht; kommerzieller Support ist zudem kostenpflichtig.

Das „Bundesamt für Sicherheit in der Informationstechnik“ (BSI) empfiehlt daher auch, freie Software in die Überlegungen bei der Auswahl sicherheitskritischer Software mit einzubeziehen, und nennt vor allem die Möglichkeit der eigenen Source-Code Validierung als Vorteil (`http://www.bsi.de/literat/doc/-fuhrberg.htm`). An anderer Stelle wird dort auch die freie Verfügbarkeit von Source-Code als Qualitätsmerkmal bezeichnet (`http://www.bsi.bund.de/-bsi-cert/webserv.htm`).

Natürlich gibt es auch Argumente für den Einsatz kommerzieller Werkzeuge. Wenn Sie ganz spezielle Anforderungen haben, kann es sein, daß diese nur ein kommerzieller Firewall erfüllt. Hersteller von Firewalls investieren viel Geld und Zeit in erweiterte Sicherheitsmechanismen. Diese Ressourcen stehen der Entwickler-Gemeinde freier Software nicht immer zur Verfügung.

Allerdings sind nicht immer alle Mechanismen kommerzieller Produkte transparent. Sie müssen damit rechnen, daß Sie später nicht alles auf einem kommerziellen Produkt verstehen oder gar ändern können, sich das kommerzielle Produkt also genau für den Zweck einsetzen läßt, für den es geplant war, aber eine spätere Korrektur problematisch bis unmöglich sein kann.

Wenn Sie sich für den Einsatz von kommerziellen Produkten entscheiden, aber trotzdem großen Wert darauf legen, den gesamten Firewall möglichst gut kontrollieren und steuern zu können, sollten Sie in Erwähnung ziehen, das kommerzielle Produkt in Kombination mit Open-Source-Werkzeugen einzusetzen. Dann ist es leichter, Vorgänge, die Sie auf dem kommerziellen Produkt nicht verstehen, über die Open-Source-Werkzeuge zu kontrollieren. Meistens bedeutet das den Einsatz zusätzlicher Hardware für die Open-Source-Werkzeuge, da sich diese in vielen Fällen zusammen mit kommerziellen Produkten nicht auf demselben System einsetzen lassen (etwa dann, wenn das kommerzielle Produkt ein eigenes, speziell angepaßtes Betriebssystem mitbringt).

Zunächst müssen Sie für den Selbstbau eines Firewalls nicht viele Voraussetzungen mitbringen. Wenn Sie über eine Linux-Installation verfügen und damit auch Ihren Internetanschluß realisieren, können Sie einige Verfahren direkt umsetzen. Sie sollten es sogar.

Wenn Sie eine komplexere Firewallstruktur einsetzen wollen (oder müssen), aber bisher wenig Erfahrung mit der Realisierung gesammelt haben, sollten Sie dieses Buch Schritt für Schritt durcharbeiten und dabei auch die praktischen Lösungsbeispiele der Reihe nach implementieren, um Übung mit den eingesetzten Verfahren zu bekommen. Dann fällt es Ihnen auch leichter, abseits der vorgeschlagenen Lösungen eigene Varianten zu entwickeln, die optimal auf Ihren Bedarf zugeschnitten sind.

Der Aufbau dieses Buches

Zunächst werden Sie im Kapitel „Wozu braucht man Firewalls?“ in die Thematik eingeführt. Sie werden erfahren, daß ein Firewall in erster Linie ein Konzept, weniger ein Stück Hardware ist, auch wenn es manchmal so aussieht. Wozu ein Firewall dient – und was er *nicht* leisten kann – wird hier beschrieben.

Das Kapitel „Security Policy“ macht Sie mit den Grundkonzepten der IT-Sicherheit vertraut und bietet Ihnen Handwerkszeug, um eine eigene Sicherheitspolitik festzulegen, denn schließlich möchten Sie Daten und Computersysteme schüt-

zen. Haben Sie erst einmal Ihre Sicherheitspolitik umrissen, können Sie einen Firewall gezielt als Bestandteil Ihrer IT-Sicherheit einsetzen.

Anschließend führt ein Kapitel in die „Grundlagen des Firewalldesigns“ ein. Sie lernen die Bausteine von Firewalls, verschiedene Firewallkonzepte und damit eine Vielzahl neuer Begriffe kennen.

Im Kapitel „Paketfilterung und Network Address Translation“ geht es bereits in die Praxis: mit Bordmitteln von Linux auf einer sehr tiefen Ebene, nämlich im Kernel selbst (dies ist außerordentlich wichtig für ein stabiles und sicheres System), lassen sich alle Anforderungen an einen Paketfilter abdecken. Network Address Translation, im Linux-Bereich ist eher die Untermenge „Masquerading“ bekannt, wird ebenfalls beschrieben.

Zu einem Firewall gehören neben Paketfiltern auch Application-Level-Gateways, die sich mit sogenannten „Proxies“ realisieren lassen. Die wichtigsten freien Proxies aus dem Internet werden hier vorgestellt.

Bevor es an den eigentlichen Aufbau geht, noch ein paar Überlegungen zu den häufigsten Anwendungen im Kapitel „Internetdienste und Firewalls“. Sie finden dort einige gängige Vorschläge, wie Sie verschiedene Anwendungen mit Ihrem Firewall umsetzen können.

Nach so vielen Grundlageninformationen wird es Zeit für praktische Lösungsbeispiele. In den beiden Kapiteln „Firewalls bauen“ finden Sie verschiedene Abschnitte: zunächst den einfachen Fall eines direkt mit dem Internet verbundenen Rechners, auf dem einige grundlegende Security-Regeln umgesetzt werden. Darauf aufbauend, beschäftigt sich der zweite Abschnitt mit einem Router mit Firewallfunktion, der ein lokales Netz mit dem Internet verbindet. Im zweiten Teil dieses Abschnitts wird der Router über eine zweite Netzwerkkarte erweitert, um ein Grenznetz zu implementieren. Der dritte Abschnitt ist dann ein klassischer, zweistufiger Firewall mit einem echten Grenznetz zwischen zwei Routern: schon etwas anspruchsvoll und aufwendig, aber sehr modular anwendbar. Den Abschluß bildet dann ein Ausblick auf Hochsicherheitsanforderungen.

Wenn Sie es bis dahin geschafft haben, haben Sie sich ganz bestimmt eine Atempause verdient. Allerdings sollten Sie sich damit noch nicht zufrieden geben: Einen Firewall zu implementieren ist eine Sache, ihn zu betreiben, zu warten und zu überwachen eine andere, mindestens ebenso wichtige. Darauf geht ausführlich das Kapitel „Maintenance: Firewalls installieren, betreiben, überwachen“ ein.

Linz, im Juli 2003 *Wolfgang Barth*

Kapitel 2

Wozu braucht man Firewalls?

2.1 Der Begriff „Firewall“

Das englische Wort „*Firewall*“ läßt sich mit „Brandschutzmauer“ übersetzen. Dieses Bild ist treffend, aber auch wiederum nicht ganz. Eine Brandschutzmauer soll das Übergreifen eines Brandes von einem Bereich (z. B. einem Gebäudeteil) auf einen angrenzenden verhindern. Aufgabe einer Brandschutzmauer ist es also, nichts hindurch zu lassen. Andererseits hat eine Brandschutzmauer manchmal auch Löcher (in Form von Brandschutztüren zum Beispiel). Die Löcher in einer Brandschutzmauer lassen den Verkehr solange ungehindert passieren, bis ein Gefahrenfall (Brand) eintritt. Dann schließen die Türen hermetisch ab.

Der hundertprozentige Schutz vor den Gefahren aus dem Internet ist nur gegeben, wenn gar keine Verbindung besteht. Es gibt viele Firmen, die sich den Aufwand zum Betreiben eines Firewalls gar nicht leisten können. Die Lösung kann dann mitunter sein, sogenannte Standalone-PCs aufzustellen, die zwar eine Verbindung zum Internet, aber keine Verbindung zum internen Netzwerk haben. Datentransfer von und zu diesen PCs (z. B. über Disketten) ist streng verboten. Manchmal ist das praktikabel – effektiv und kostengünstig ist es allemal. Wenn Sie die absolut sichere Brandschutzmauer suchen, dann benötigen Sie also keinen Firewall, sondern nur eine Schere für Ihre bisherige Internetanbindung.

Was also ist ein Firewall genau? Sie möchten Daten mit anderen austauschen, egal, ob über Internet oder Extranet. Gleichzeitig soll aber auch sichergestellt werden, daß nur die geforderten Anwendungen und der damit verbundene Datenaustausch realisiert werden. Andere Daten sollen keinesfalls in das eigene Netzwerk gelangen oder dieses verlassen.

In diesem Sinne ist ein Firewall eher vergleichbar mit einer Pforte mit strenger Zugangskontrolle (und eben weniger mit einer Brandschutzmauer). Ein Firewall trennt einen öffentlichen Bereich von einem privaten Bereich des Netzwerks ab.

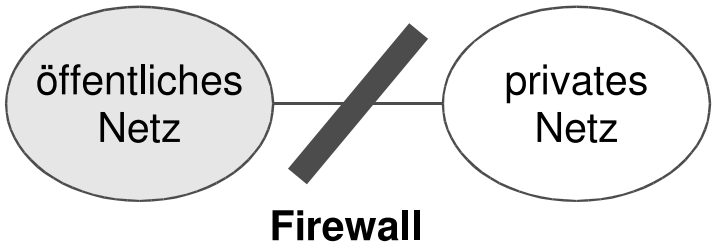

Abbildung 2.1: Firewall: Trennung öffentlicher und privater Bereiche

Der Einsatz eines Firewalls bedeutet bedingte Durchlässigkeit. Dort, wo Durchgangsverkehr gestattet ist, öffnet man ein Loch in der Brandschutzmauer und postiert einen Pförtner. Dort, wo kein Durchgang notwendig ist, mauert man die Brandschutzmauer zu, benötigt dann aber auch keinen Pförtner bzw. keinen Firewall.

Der Pförtner – um im Bild zu bleiben – kontrolliert die Passanten nach definierten Kriterien. In Hochsicherheitsbereichen nicht nur die Person selbst, sondern auch Inhalte von Aktentaschen, Kofferräumen. Möglicherweise untersucht er sogar auf Strahlung. Üblicherweise assoziiert man mit einer Pforte eher die Einlaßkontrolle, aber auch die umgekehrte Richtung kann kontrolliert werden, in einigen Umgebungen muß sie das sogar.

Ein Firewall ist ein Konzept für die Verbindung zwischen einem öffentlichen und einem nicht-öffentlichen Bereich (s. Abbildung 2.1). Ein Firewall kann nur aus einem einzigen Stück Hardware bestehen, muß es aber nicht. Der Begriff Firewall kann auch eine ganze Infrastruktur mit mehreren Servern, Routern und anderen Komponenten beschreiben.

2.2 Was ein Firewall kann . . .

Ein Firewall kontrolliert eine Passage zwischen dem öffentlichen und dem nicht-öffentlichem Teil eines Netzwerks und registriert den Verkehr zwischen diesen:

- *Durchsetzen von Sicherheitsbestimmungen*: Eine Firma kann ihren Mitarbeitern per Anordnung verbieten, bestimmte Daten aus dem Firmengebäude mit nach Hause zu nehmen. Einige mögen sich auch daran halten… Daß sich möglichst viele daran halten ist nur zu erreichen, indem die Anordnung auch kontrolliert wird. Der Firewall ist hier der Pförter bzw. die Radarfalle. Er kann verhindern, daß Daten über das Netzwerk (E-Mail, FTP oder ähnliches) nach draußen gelangen. Er kann natürlich nur solche Sicherheitsbestimmungen durchsetzen, die sich auf den Einsatzbereich eines Firewalls, das Netzwerk, beziehen.

- *Protokollierung*: Ein Firewall protokolliert auch den laufenden Datenverkehr. So lassen sich später Vorgänge nachvollziehen und beurteilen. Bei einem Einbruch in einem Wohnhaus hilft die Spurensicherung bei der Beurteilung. Ist tatsächlich eingebrochen worden oder wurde es nur erfolglos versucht? Es folgt eine Bestandsaufnahme, ob evtl. etwas fehlt.
  Mit Daten ist das wesentlich schwieriger. Ob Daten gelöscht wurden, ist schon weitaus schwieriger auszumachen, ob ggf. Daten kopiert wurden, kann ohne Spurensicherung am Firewall überhaupt nicht beurteilt werden. Vielleicht wurden auch Datenbestände manipuliert. Aussagekräftige Protokolle helfen sehr bei der Frage, nach welchen Schäden man eigentlich suchen soll.

2.3 . . . und was ein Firewall nicht kann

Ein Firewall erfüllt nicht all Ihre Sicherheitsanforderungen, er ist nur ein Baustein dazu. Ein Firewall hat auch Grenzen:

- *Ein Firewall schützt nur Verbindungen, die über ihn laufen*: Am Beispiel der Pforte ist das leicht zu verstehen. Wenn Ihr Hochsicherheitstrakt noch einen Hintereingang hat, ist die Zugangssicherheit nur so hoch, wie die schwächere der beiden Pforten. Es nützt Ihnen gar nichts, den Haupteingang immer sicherer zu machen, wenn Sie nicht in gleichem Maße den Hintereingang sichern. Es muß auch nicht immer eine Türe sein: ein Fenster ist ebenfalls ein Sicherheitsrisiko. Mit anderen Worten: Der beste Firewall schützt Sie nur auf dem vorgegebenen Weg vor Datenmißbrauch. Wenn Sie aber gleichzeitig zulassen, daß Mitarbeiter unkontrolliert Modems benutzen dürfen, Disketten oder andere Datenträger mit nach Hause nehmen können, hilft Ihnen der Firewall an dieser Stelle nicht weiter. Sie müssen zusätzliche Maßnahmen ergreifen.
- *Angriffe von innen*: Ein Firewall wird ja nur implementiert, wenn ein gewisser Datenverkehr zugelassen werden soll, in der Regel von innen nach außen. Das bedeutet, daß es sogenannte erlaubte Verbindungen gibt. Wenn sich nun jemand diese erlaubten Kanäle zunutze macht, kann der Firewall umgangen werden. Gerade die steigende Sicherheit von Firewalls führt zu neuen Angriffsszenarien: trojanische Pferde setzen sich im internen Netzwerk fest und nutzen die geringe Restdurchlässigkeit eines Firewalls, um ihn zu umgehen. Sie müssen zusätzlich zum Firewall Maßnahmen ergreifen, die solche Attacken vermeiden, erkennen und abstellen helfen.
- *Untreue Mitarbeiter*: Ein Firewall schützt nicht allein vor Mitarbeitern, die bewußt versuchen, Kontrollmechanismen zu umgehen. Erlaubte Verbindungen können mit einigem Aufwand auch zum Datenexport mißbraucht werden. Wenn eine Verbindung prinzipiell erlaubt ist, kann ein Firewall besten-

falls protokollieren. Sie brauchen andere Mechanismen (z. B. eine Dienstanweisung, Zugangsbeschränkungen intern), die zusammen mit den Protokollen zu einem wirksamen Schutz führen (Androhung von Sanktionen).

- *Hacker im eigenen Netzwerk*: Der Firewall trennt den internen vom öffentlichen Bereich ab. Sitzt der Angreifer bereits im internen Netzwerk (ein Mitarbeiter etwa), schützt er Sie nicht vor Angriffen auf im Netzwerk erreichbare Rechner. Sie benötigen zusätzlich zum Firewall weitere Mechanismen, die Angriffe transparent machen können (und Anordnungen wie etwa Dienstanweisungen, die es Ihnen erlauben, aus festgestellten Attacken auch Konsequenzen ziehen zu können). Und Sie kommen trotz des Firewalls nicht umhin, gefährdete Rechner zusätzlich zu härten (d. h., Rechner gegen Attacken unempfindlicher zu machen). Möglicherweise kommen Sie auch zu dem Schluß, daß Sie Ihr internes Netzwerk in verschiedene Bereiche aufteilen, die zusätzlich durch Firewalls voneinander getrennt werden.

2.4 Grundwerte der IT-Sicherheit

Wenn Sie an einer gesichterten Pforte um Einlaß bitten, werden Sie automatisch nach verschiedenen Kriterien überprüft.

Zunächst hat jemand entschieden, daß es schützenswerte Güter im Unternehmen gibt. Im Gegensatz zu öffentlichen Bereichen, zu denen jedermann ungehindert Zutritt hat, gibt es hier eine Grenze. Die Frage lautet: Was, warum und wovor soll etwas geschützt werden? Definiert werden zu schützende Werte; diese heißen im Bereich der IT-Sicherheit: *Vertraulichkeit*, *Verfügbarkeit* und *Integrität*.

Die Entscheidung, etwas schützen zu wollen, ist bereits Teil einer Sicherheitspolitik. Sicherheitspolitik umfaßt das gesamte Spektrum von der Entscheidung, etwas zu schützen, bis zur einzelnen Anweisung, wie das geschehen soll. Das Thema Sicherheitspolitik ist Gegenstand des Kapitels 3.

2.4.1 Vertraulichkeit

Vertraulichkeit ist wohl der Begriff, an den man beim Stichwort IT-Sicherheit am ehesten denkt. Vertraulichkeit bedeutet, daß Daten nicht in unbefugte Hände geraten dürfen. Beispiele für vertrauliche Daten:

- *Personenbezogene Daten*: Schutz dieser Daten wird unter dem Stichwort „Datenschutz“ zusammengefaßt. Eine Verletzung des Datenschutzgeheimnisses kann ganz unterschiedliche Auswirkungen haben: Verlust des Ansehens einer Person, wirtschaftliche Schäden, weil z. B. jemand aufgrund von bekanntgewordenen medizinischen Daten nirgends mehr eingestellt wird etc. Allen gemeinsam ist, daß der Verlust der Vertraulichkeit personenbezogener Daten fast immer der Person schadet, zu der diese Daten gehören.

Der allgemein gebrauchte Begriff „Datenschutz“ in seiner gesetzlich geregelten Form zielt immer auf den Schutz von personenbezogenen Daten ab.

- *Betriebsgeheimnisse*: Betriebliche Geheimnisse können für eine Firma einen erheblichen Wert darstellen. Erfindungen sichern mitunter einen Technologievorsprung von mehreren Jahren, manchmal hängt das Überleben eines Unternehmens davon ab. Aber auch einfachere Interna, wie zum Beispiel die Angebotslage für ein Großprojekt, können das Geschäftsergebnis erheblich beeinflussen. Ein in falsche Hände geratenes Angebotsfax ermöglicht es der Konkurrenz, durch Unterbieten zum Zuge zu kommen.
- *Zugangsmechanismen*: Informationen über Sicherheitsstrukturen (über den Firewall etwa) oder gar User-Kennungen und Passwörter erleichtern einen Angriff von außen erheblich oder ermöglichen ihn sogar erst.

2.4.2 Verfügbarkeit

Wie verfügbar sind Ihre Systeme und Ihre Daten? Verfügbar sind Daten und Dienste dann, wenn Sie innerhalb einer zuvor festgelegten Zeitspanne (zum Beispiel Bürozeit, 5x8 h, 7x24 h etc.) im Bedarfsfalle ohne Einschränkung ordnungsgemäß erreichbar sind. Einige Beispiele für Verfügbarkeiten bzw. Einschränkungen:

- *Systemverfügbarkeit*: Gemessen wird in der Regel die Zeit ungeplanten Ausfalles pro Jahr. Gezielte Wartungsintervalle, zu denen ein Administrator Arbeiten vornimmt und das System gezielt herunterfährt, sind dabei ausgenommen.
  Eine Angabe von 99,999% Verfügbarkeit bedeutet einen ungeplanten Ausfall von 5 Minuten insgesamt pro Jahr. Das ist normalerweise nur mit zertifizierter Soft- und Hardware zu erreichen.
- *Netzwerkverfügbarkeit*: beschreibt ebenfalls die ungeplanten Ausfallzeiten pro Jahr, bezogen auf das Netzwerk.
  Bespiel: Der Server steht im Nachbargebäude, draußen im Hof beschädigt ein Bagger bei Erdarbeiten das Glasfaserkabel. Mangels Backup-Leitung über einen anderen Weg oder redundanter Systeme stehen die Daten am Arbeitsplatz nicht zur Verfügung.
- *Höhere Gewalt*: Ereignisse, die nicht vorhersehbar sind und nicht durch menschlichen Eingriff verursacht werden. Fahrlässigkeit darf ebenfalls nicht die Ursache sein. Hierzu zählen Blitzschlag, Brand, Wassereinbruch, Erdbeben und andere.
  Beispiel: im Rechnerraum bricht ein Brand aus, oder ein in der Decke verlaufendes Wasserrohr wird undicht. Die komplette Hardware muß ersetzt werden. Möglicherweise stehen Daten für mehrere Tage oder Wochen nicht zur Verfügung.

- *Sabotage*: Verschiedene bekannte Angriffsmechanismen über das Internet versuchen erst gar nicht, in ein System einzudringen, sondern es einfach außer Gefecht zu setzen. Diese Angriffsart nennt man auch „Denial of Service“. Dies können direkte Angriffe sein, aber z. B. auch ein Trojanisches Pferd. Vor einiger Zeit legte ein solches die komplette Infrastruktur eines amerikanischen Providers für mehr als 18 Stunden lahm. Kundenproteste und Schadenersatzforderungen waren die Folge.

2.4.3 Integrität

Als Frage formuliert: Sind Ihre Daten und Ihre Systeme noch das, was sie vorgeben zu sein? Sind sie korrekt, unverändert?

- *Systemintegrität*: bezeichnet die korrekte Funktionsweise eines Systems. Die Systemintegrität ist verletzt, wenn das System – etwa der Firewall – nicht mehr das tut, wofür es konzipiert und in Betrieb genommen wurde.
  Beispiel: Ein Eindringling in ein Rechnersystem erlangt über eine Sicherheitslücke Administratorrechte und legt sich selbst einen Account an, mit dem er selbst nach Schließen der Sicherheitslücke weiterhin Zugang zum System hat. Darüber hinaus installiert er ein Trojanisches Pferd, das ihm auch nach Entfernen des zusätzlichen Accounts eine weitere Zugangsmöglichkeit offenhält.
- *Datenintegrität*: Unversehrtheit der Daten, das heißt, die Daten enthalten das, was sie normalerweise sollen. Werden die Daten mißbräuchlich verändert, ist die Datenintegrität verletzt.
  Beispiel: Ein Virus macht sich im System breit und ersetzt z. B. einen bestimmten String durch einen anderen (zum Beispiel „Kreditor“ durch „Debitor“). Solange Sie den Virus nicht kennen, wissen Sie auch nicht, welchen Schaden er anrichtet. Makroviren für Office-Applikationen lassen sich mit wenig Aufwand von weniger wohl gesinnten Zeitgenossen sehr einfach in Viren umbauen, die zum Beispiel in einer Tabellenkalkulation willkürlich und zufällig einzelne Feldinhalte ändern.

2.5 Zusammenfassung

Ein Firewall ist ein Element in einem umfassenden Sicherheitskonzept zwischen öffentlichem und nicht-öffentlichen Bereich eines Netzwerks. Er hat die Aufgabe der Zugangskontrolle und Zugangsprotokollierung. Er kann (muß aber nicht) aus Hard- und Software bestehen. Er ist zwar ein wichtiger Baustein innerhalb dieses Konzepts, muß aber durch weitere Maßnahmen im Rahmen einer Sicherheitspolitik ergänzt werden.

Kapitel 5

Paketfilterung und Network Address Translation

Die klassische Paketfilterung entscheidet über den Verbleib der Pakete, verändert jedoch nicht deren Inhalte. Mitunter setzen Paketfilter fragmentierte Pakete wieder zusammen, doch auch hier ändert sich der Inhalt nicht.

Network Address Translation (NAT) – der Name sagt es ja schon – verändert dagegen Adressen. Im Linux Kernel sind jedoch Network Address Translation und Paketfilterung sehr nahe beieinander implementiert und werden auch mit denselben Werkzeugen konfiguriert.

Die Paketfilterung unter Linux erfolgt bereits im Kernel selbst. Im Vergleich zu Softwarepaketen, die oberhalb des Kernel arbeiten und als eigenständige Prozesse laufen, die ihrerseits wiederum mit dem Kernel kommunizieren müssen, ist das von Vorteil. Einem potentiellen Angreifer bieten sich dadurch weniger Möglichkeiten zur Manipulation. Firewallsoftware, die oberhalb der Betriebssystemebene als normales Programm läuft, kann noch so sicher implementiert sein. Wenn das Betriebssystem selbst bereits Mängel und Sicherheitslücken aufweist, bleibt es angreifbar.

Ein Nachteil ist sicherlich, daß eine in den Kernel integrierte Lösung schwerlich auf andere Betriebssysteme portiert werden kann.

Um mit Linux Pakete filtern zu können, benötigt man zwei Dinge:

- Einen entsprechend vorbereiteten Kernel, der neben der Firewall-Unterstützung auch andere Sicherheitsfeatures und Routing unterstützen soll.
- Ein Werkzeug, mit dem man die Regeln für die Paketfilterung auf Kernelebene setzen und bearbeiten kann. Gerade wegen der Kernelnähe muß man für jede Kernelversion das richtige Werkzeug wählen: `iptables` für Kernel 2.4 oder Kernel 2.6, `ipchains` für Kernel 2.2, und `ipfwadm` für Kernel 2.0.

Die Codebasis ab Kernelversion 2.4 wurde gründlich überholt und „ausgemistet“. Zwar werden die Mechanismen von `ipchains` und `ipfwadm` durch ladbare Module auch noch in Kernel 2.4 unterstützt (aber nicht mehr in Kernel 2.6), wegen des besseren Konzeptes und der Zukunftssicherheit werden wir hier jedoch nur auf `iptables` eingehen, damit wird Kernel 2.4 oder 2.6 vorausgesetzt.

Auf die Unterschiede zwischen `iptables` und `ipchains` geht der Anhang E ein. Ergänzende Informationen zu diesem Buch, insbesondere zu `ipchains`, sind im Internet zu finden unter `http://www.swobspace.de/`.

Die unterschiedlichen Mechanismen dürfen auf keinen Fall miteinander gemischt werden. Wer unter Kernel 2.4 mit `iptables` arbeitet, darf nicht das Modul für `ipchains` laden.

5.1 Kernel vorbereiten

Um von den Filterfähigkeiten des Kernel Gebrauch machen zu können, müssen die zum Firewalling gehörenden Teile einkompiliert sein. Was die Netzwerkfähigkeit und den Netfilter-Code angeht, sind die Unterschiede in der Konfiguration zwischen Kernel 2.4 und 2.6 nicht sehr groß. Allerdings ist die Konfiguration von Version 2.6 wesentlich strukturierter, so daß man vielleicht am Anfang etwas suchen muß, bis man die gewünschten Einstellungen findet. Die Konfiguration für Kernel 2.4 wird hier klassisch über `make menuconfig` (ASCII-Menue) durchgeführt, für die Konfiguration von Kernel 2.6 verwenden wir die Qt-basierte Oberfläche mit `make xconfig`.

5.1.1 Kernel 2.4

Zur Konfiguration von Kernel 2.4 wird im Verzeichnis `/usr/src/linux` das Programm `make` aufgerufen:

```
\# make menuconfig
```

Neben der generellen Netzwerk-Unterstützung unter `General setup` werden folgende Einstellungen aus dem Menü `Networking options` benötigt:

```
<*> Packet socket
[*] Network packet filtering (replaces ipchains)
[*] Network packet filtering debugging
...
[*] TCP/IP networking
...
[*] IP: TCP syncookie support (disabled per default)
  IP: Netfilter Configuration  --->
```

Das Submenu `Netfilter Configuration` faßt alle für die Netfilter-Architektur möglichen Parameter zusammen.

```
<*> Connection tracking (required for masq/NAT)
<M>   FTP protocol support
<*> IP tables support (required for filtering/masq/NAT)
<M>   limit match support
<M>   MAC address match support
<M>   netfilter MARK match support
<M>   Multiple port match support
<M>   TOS match support
<M>   Connection state match support
<*>   Packet filtering
<M>     REJECT target support
<*>   Full NAT
<M>     MASQUERADE target support
<M>     REDIRECT target support
<*>   Packet mangling
<M>     TOS target support
<M>     MARK target support
<*>   LOG target support
<M>   TCPMSS target support
<M> ARP tables support
<M>   ARP packet filtering
```

Die einzelnen Einstellungen werden in

`/usr/src/linux/Documentation/Configure.help`

näher beschrieben. Diese zu jedem Eintrag entsprechende Hilfe erhält man auch im Konfigurationsmenü über die Auswahl `Help`.

Die wesentlichen Einstellungen daraus sind:

- `Network packet filtering`
  Schaltet grundsätzlich das Netfilter Framework ein. Ist nötig, sobald der Rechner als Paketfilter arbeiten soll oder NAT durchgeführt werden muß.
- `IP: TCP syncookie support`
  ist eine Abwehrmaßnahme gegen SYN flooding, eine Denial-of-Service-Attacke. Auf dem äußeren Router/Filter unbedingt verwenden. Allerdings muß der SYN-Cookie Support nach dem Booten aktiviert werden:
  ```
  echo "1" > /proc/sys/net/ipv4/tcp_syncookies
  ```
- `IP tables support (required for filtering/masq/NAT)`
  Der eigentliche Kern der Paket-Manipulation im Kernel 2.4. Darunter finden sich eine Reihe von Funktionalitäten als einzelne Module implementiert. Im Gegensatz zu früheren Kernels sind auch schon nichttriviale Targets

wie `REJECT`, `MASQUERADE` und `REDIRECT` sowie Logging in eigene Module ausgelagert worden. Das hält den eigentlichen Kernelcode klein und übersichtlich und erleichtert die Erweiterung.
Unter dem Punkt `IP tables support` werden sich bei Weiterentwicklung des Kernel sicher immer wieder Veränderungen ergeben.

- `Connection tracking (required for masq/NAT)`
  In einer eigenen Tabelle werden ausgehende Pakete und Verbindungen gelistet, um eingehende Pakete besser einer bestehenden Verbindung zuordnen zu können. Das ermöglicht erst NAT, erlaubt aber auch eine dynamische Paketfilterung (Modul `Connection state match support`).

Bereits während der Entwicklung des 2.2er Kernel wurden verschiedene, zunächst einkompilierte Flags in das `/proc`-Filesystem aufgenommen und können nun während der Laufzeit des Kernel konfiguriert werden. Da viele dieser Einstellungen nicht direkt unter den Oberbegriff „Paketfilter“ fallen, aber dennoch unterstützend auf einem Firewall eingesetzt werden können und auch sollen, sind die Optionen zur Runtime-Konfiguration im Anhang C gesondert zusammengefaßt.

Der Kernel wird übersetzt und installiert mit:

```
# make dep && make install
# make modules && make modules_install
```

5.1.2 Kernel 2.6

Zur Konfiguration von Kernel 2.6 ruft man im Verzeichnis `usr/src/linux` das Programm `make` auf:

```
# make xconfig
```

Daraufhin wird ein graphisches Fenster geöffnet, das grob aus drei Teilen besteht: links eine Art Navigationsleiste, rechts oben die Auswahl der Optionen, und rechts unten gleich die Online-Hilfe zum ausgewählten Parameter. Der Übersicht halber wird nachfolgend aber immer nur der für die Paketfilterung gerade relevante Ausschnitt dargestellt. Bei den Optionen stellt ein leeres Kästchen eine abgewählte Option dar, mit einem Häkchen versehen wird die Option fest einkompiliert. Ein Punkt im Kästchen bedeutet, daß die Option als Modul übersetzt wird.

Abbildung 5.1 zeigt den Einstieg für die Netzwerk- und Netfilter-Konfiguration. Links wählt man „Networking support“ aus, rechts läßt sich dann unterhalb des Eintrages „Networking support“ das Untermenu „Networking options“ öffnen.

In dem Menübaum rechts (Abbildung 5.2) erhält man neben dem Eintrag „Network packet filtering“ auch noch weitere, wichtige Optionen. Selbstverständlich sollte

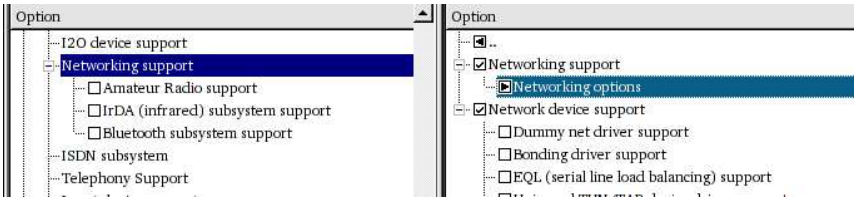

Abbildung 5.1: `make xconfig`: Networking support (Kernel 2.6)

- „TCP/IP networking“ angekreuzt sein, ebenso
- „Packet socket“,
- „Unix domain sockets“ und
- „TCP syncookie support“.

Für natives IPsec im Kernel 2.6 (siehe auch Abschnitt 11.3) werden folgende Optionen benötigt:

- „PF_KEY sockets“,
- „AH transformation“,
- „ESP transformation“,
- „IPComp transformation“, und
- „IPsec user configuration interface“.

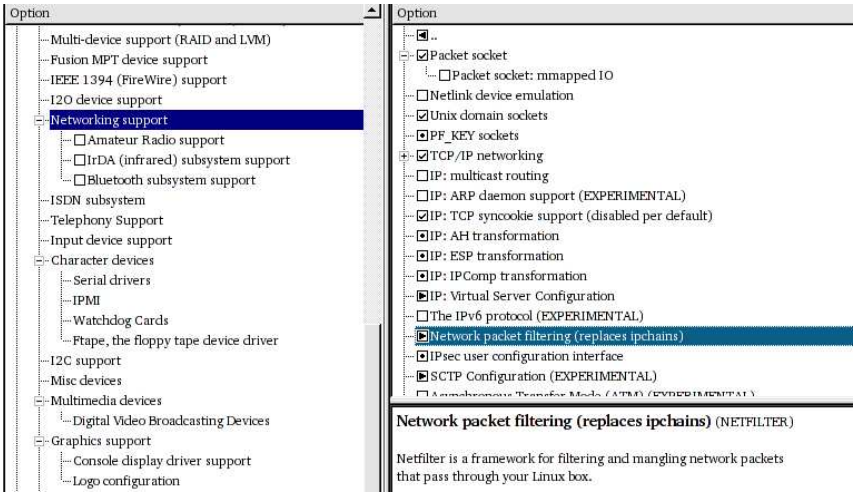

Abbildung 5.2: `make xconfig`: Networking options (Kernel 2.6)

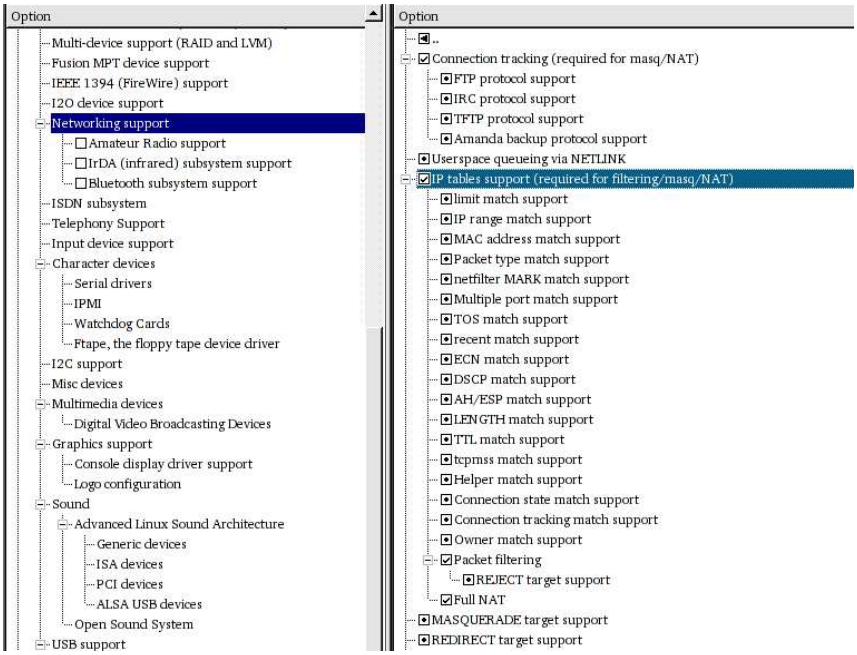

Abbildung 5.3: `make xconfig`: Network packet filtering (Kernel 2.6)

Abbildung 5.3 zeigt schließlich den unter „Network packet filtering“ aufgerissenen Menübaum an.

Nach vollendeter Konfiguration sollte man „Speichern“ nicht vergessen, bevor man das graphische Tool wieder verläßt. Der Kernel wird wieder übersetzt und installiert mit:

```
# make install
# make modules && make modules_install
```

5.1.3 Modularer oder statischer Kernel?

Im vorangegangenen Abschnitt wurde der übliche Weg gewählt, einige Kernelbestandteile als Modul zu übersetzen, um damit flexibel reagieren zu können. Der Kernel ist beim Systemstart relativ klein, benötigte Teile werden in Form von Modulen einfach nachgeladen.

An dieser Stelle soll nicht verschwiegen werden, daß ein modularer Kernel Hackern eine zusätzliche Möglichkeit bietet: gelingt es dem Eindringling, sich Zugang zum System und Root-Rechten zu schaffen, ist er in der Lage, ebenfalls ein eigenes Modul zusätzlich zu den restlichen Modulen zu laden. Ein bösartiges

Modul kann Hackeraktivitäten unsichtbar machen. Sucht man auf so einem System, verschleiert es die Existenz von Backdoors oder anderen, unerwünschten Dingen. Als Kernelmodul kann es auf einer sehr tiefen Ebene agieren und Werkzeugen, die nach Einbruchsspuren suchen, nach oben hin bereits eine heile Welt vorgaukeln.

Will man in letzter Konsequenz jede Möglichkeit eines solchen Stealth-Moduls ausschließen, kann man den Kernel statisch – also ohne modulare Unterstützung – übersetzen. Unter dem Menüpunkt „Loadable module support“ findet sich der entsprechende Eintrag:

```
[ ] Enable loadable module support
```

Für einen rein statischen Kernel ohne modulare Unterstützung darf dieser Punkt nicht angewählt sein.

Ein statischer Kernel hat allerdings auch Nachteile: alle benötigten Funktionen müssen fest einkompiliert werden. Bei jeglicher Änderung muß der Kernel neu übersetzt werden. Damit der Kernel nicht zu groß wird, sollte man alle nicht benötigten Bestandteile außen vor lassen. Ein bißchen Erfahrung im Backen von Linux-Kerneln ist für den Bau eines statischen Kernel schon vonnöten.

5.2 Paketfilterung mit `iptables`

Der Kernel arbeitet in der Paketfiltertabelle mit Listen von Regeln. Diese Listen bezeichnet man als „*Firewall Chain*“, oder auch nur einfach „*Chain*“. Eine Firewall Chain ist ein zusammengehörender Satz von Regeln, die nacheinander abgearbeitet werden. Im folgenden soll für den englischen Begriff „Chain“ die Bezeichnung „*Regelkette*“ verwendet werden. Permanent (d. h. bereits beim Systemstart) vorhanden sind drei Regelketten: die `INPUT` Chain, die `FORWARD` Chain und die `OUTPUT` Chain, vorausgesetzt, Firewalling ist in den Kernel einkompiliert.

Wie Glieder in einer Kette können Filterregeln jederzeit in eine Regelkette eingefügt oder aus dieser entfernt werden. Mit dem Tool `iptables` ist es auch möglich, zur leichteren Verwaltung und besseren Lesbarkeit eigene Regelketten zu erstellen und zu bearbeiten.

Wenn ein Paket über ein Interface hereinkommt (das kann eine Ethernetkarte, aber auch jedes andere Interface wie ISDN oder das Loopback-Interface sein), durchläuft es einen vorgezeichneten Weg, der aus mehreren Regelketten und einigen weiteren Schritten besteht (siehe Abschnitt 5.2.1).

In jeder Regelkette werden die einzelnen Regeln der Reihe nach abgearbeitet und geprüft, ob eine Regel auf das aktuelle Paket zutrifft oder nicht. Trifft eine Regel auf das Paket zu, wird – vereinfacht gesagt – das Paket akzeptiert oder abgelehnt. An dieser Stelle wird dann die jeweilige Kette verlassen, es sei denn, die Regel

dient ausschließlich der Protokollierung. Deshalb ist die Reihenfolge bei der realen Implementierung von Regelketten äußerst wichtig: trifft eine allgemeinere Regel bereits weiter oben in der Kette zu, kommt eine möglicherweise speziell für einen besonderen Fall gedachte Regel gar nicht mehr zur Anwendung.

5.2.1 Ablaufdiagramm des Linux-Kernelfilters

Der Kernel untersucht zunächst das eingehende Paket auf die Zieladresse, um zu entscheiden, wohin das Paket geliefert werden soll (Routing Decision, siehe Abbildung 5.4). Ist es für den lokalen Host bestimmt, wird das Paket an die `INPUT` Chain weitergeleitet. Wenn das Paket dort durchgelassen wird, landet es bei einem lokalen Prozeß.

Ist das Paket nicht für den lokalen Rechner bestimmt, wird das Paket unter zwei Bedingungen an die `FORWARD` Chain weitergereicht: der Kernel muß prinzipiell Forwarding eingeschaltet haben, und der Kernel muß wissen, wie das Paket geroutet werden soll (zum Paket passender Eintrag in der Routing Tabelle). Treffen beide Bedingungen zu, geht das Paket in die `FORWARD` Chain, und wird es dort durchgelassen, verläßt das Paket wieder den Rechner. Trifft mindestens eine der beiden Bedingungen nicht zu, wird das Paket einfach verworfen.

Natürlich kann der Rechner auch selbst Pakete versenden. Diese werden in der `OUTPUT` Chain geprüft und – falls Prüfung bestanden – direkt an das Output-Interface weitergereicht.

Gegenüber der Verwendung von `ipchains` hat sich Forwarding auch für den Anwender deutlich vereinfacht (die anderen Verbesserungen sind nicht ganz so transparent). Für ein Forwarding wird jetzt nur noch die `FORWARD` Chain durchlaufen, vorher waren alle drei permanenten Regelketten betroffen. Damit verrin-

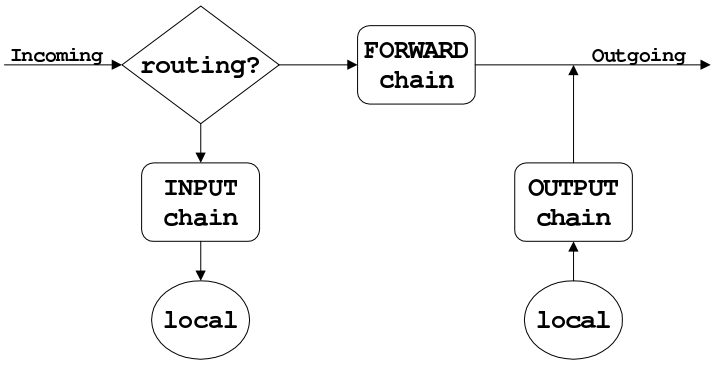

Abbildung 5.4: Grundschema der Firewall-Regelketten (Firewall Chains) im Kernel 2.4

gert sich die Anzahl der notwendigen Regeln für eine bidirektionale Verbindung von sechs auf zwei.

Filterregeln lassen sich ganz grob in drei Bestandteile zerlegen:

- die *Grundoperation*, also Einfügen einer Regel, Regelketten anlegen und löschen etc.
- *Matching-Optionen* entscheiden, ob die Regel überhaupt auf ein Paket zutrifft oder nicht. Und schließlich
- *Targets*, die beschreiben, was mit einem Paket geschehen soll, wenn die Regel schließlich zutrifft.

5.2.2 Grundoperationen für Filterregeln

- `iptables -A chain rule ...`
  fügt eine Filterregel am Ende der Regelkette ein (Append)
- `iptables -D chain rule ...`
  löscht die passende Filterregel (Delete)
- `iptables -D chain position`
  löscht die Regel an Position `position` (Delete)
- `iptables -R chain position rule ...`
  ersetzt die Regel an Position `position` mit der neuen Regel (Replace)
- `iptables -I chain position rule ...`
  fügt die Regel an Position `position` ein (Insert)
- `iptables -F [chain]`
  löscht alle Regeln aus der Regelkette `chain` (Flush)
- `iptables -P chain`
  ändert die Default Policy der Regelkette (Policy)
- `iptables -L [chain]`
  zeigt alle Regeln der ausgewählten Regelkette (List)
- `iptables -Z [chain]`
  löscht alle Zähler der Regelkette (Zero)
- `iptables -E old-chain-name new-chain-name`
  benennt eine Regelkette um (Exchange)

5.2.3 Matching-Optionen für Filterregeln

Matching-Optionen geben an, ob die Regel auf ein Paket zutrifft oder nicht.

Protokolle

`iptables -A chain` **`-p proto ...`**

Mögliche Angaben für `proto` sind `TCP`, `UDP`, `ICMP`. Groß-/Kleinschreibung spielt hier keine Rolle. Es kann auch die Protokollnummer angegeben werden. Diese Nummern sind allerdings wenig bekannt, so daß sich aus Gründen der Lesbarkeit empfiehlt, TCP/UDP/ICMP auszuschreiben.

Beispiel:

```
iptables -A INPUT -p TCP ...
```

Quell-/Zieladresse

`iptables -A chain` **`[-s source-ip] [-d destination-ip]`**

beschränkt die Filterregel auf die angegebenen Quell-/Zieladressen. Mögliche Angaben für `source-ip/destination-ip` sind:

- Full Name: `www.example.com`
- IP-Adresse: `127.0.0.1`
- Gruppe von IP-Adressen (short): `192.168.1.0/24`
- Gruppe von IP-Adressen (full): `192.168.1.0/255.255.255.0`
- Spezialfall beliebige IP-Adresse: `0/0`. Normalerweise kann man die IP-Adresse auch weglassen.

Beispiel:

```
iptables -A INPUT -s 192.168.1.2 -d 192.168.5.3 ...
```

Auswahl von Interfaces

```
iptables -A INPUT [-i in-iface]
iptables -A FORWARD [-i in-iface] [-o out-iface]
iptables -A OUTPUT [-o out-iface]
```

`in-iface/out-iface` kann jedes Interface sein. Im Augenblick existierende Interfaces können mit `ifconfig` abgefragt werden. Eingehende Pakete besitzen nur ein Input-Interface, ausgehende ausschließlich ein Output-Interface. Mit anderen Worten: eine Regel mit `-o` trifft in einer Input-Chain nie zu.

Generell können auch Interfaces angegeben werden, die noch gar nicht existieren, sondern später konfiguriert werden. Das ist insbesondere für nichtpermanente Verbindungen über ISDN oder Modems wichtig. So können schon beim Systemstart Filterregeln gesetzt werden, obwohl das Interface erst viel später initialisiert wird.

```
squidGuard -u -c /etc/squidguard.conf -C all
```

Auch wenn SquidGuard letztenendes auf die `.db`-Files zugreift, wird im Konfigurationsfile immer der Filename ohne diese Endung angegeben.

Zum Schluß noch ein Hinweis auf die Permissions der Datenbankfiles. Squid muß diese unter dem User, unter dem Squid läuft, lesen können (auch die Verzeichnisse dazu). Kann der „Squid-User" die Datenbankfiles nicht lesen, schaltet SquidGuard auf Durchzug: keine Anfrage von Squid wird ausgewertet, alles ist sozusagen erlaubt.

Zum Testen von SquidGuard übergibt man einen String, der dieselben Informationen enthält, die Squid liefern würde:

```
echo http://foo/bar 192.168.1.71/- - GET | \
         /usr/bin/squidGuard -c /etc/squidguard.conf
```

liefert entweder eine Redirect-URL im Falle einer Ablehnung, ansonsten gar nichts zurück. Zur Syntax des Strings für den Redirector siehe Abschnitt 6.1.6.

6.3 Viren scannen am Proxy

Jeder Download von Software oder anderen Dateien aus dem Internet birgt die Gefahr, daß sich darunter infizierte Dateien befinden können, die – geöffnet und auf der lokalen Workstation ausgeführt – zu einer Infektion des Systems führen und sich von dort aus im lokalen Netzwerk ausbreiten.

Man sollte dabei aber nicht aus den Augen verlieren, daß derzeit eine Kontamination per E-Mail mit Abstand die größte Infektionsquelle darstellt. Es nützt daher wenig, sich ausschließlich auf eine Absicherung gegen eine Download-Infektion zu konzentrieren. Der wichtigste Schritt zu einem sicheren Netzwerk ist daher nach der Absicherung der Internetverbindung durch einen Firewall die Einrichtung einer Schutzmaßnahme gegen E-Mail-Viren. Eine bewährte – wenn auch nicht die einzige – Methode ist `amavisd-new`[2], der auch gleich mit dem SPAM-Filter Spamassassin[3] verbunden werden kann. Die Absicherung von E-Mail über `amavisd-new` und Spamassassin wird ausführlich beschrieben in „Das Postfix-Buch", zweite Auflage[4], Open Source Press, von Peer Heinlein [Hei04].

Trotzdem: die Infektionsgefahr beim Websurfen ist durchaus vorhanden, so daß man die nächsten Seiten nicht gleich überblättern sollte, falls noch kein E-Mail-Virenschutz vorhanden ist.

[2] `http://www.ijs.si/software/amavisd`
[3] `http://www.spamassassin.org`
[4] die erste Auflage des Postfix-Buches erschien 2002 bei SuSE Press

Grundprinzip und Lösungsansätze

Über eines muß man sich von Anfang an im Klaren sein, wenn man den Datenstrom des Websurfers nach Viren hin untersucht: es kostet nicht gerade wenig Rechenzeit, und damit Systemperformance. Ein gescannter Datenstrom bietet nicht die gleiche Zugriffsgeschwindigkeit wie ein direkter, ungefilterter Zugriff auf das Internet.

Allerdings kann man sich Gedanken darüber machen, wie man die Performance-Einbrüche begrenzen kann:

- *Leistungsfähige Hardware*: kann die Performance deutlich steigern.
- *Nur neue Dateien scannen*: speichert man bereits geprüfte Dateien zwischen („Caching"), müssen nur noch unbekannte Dateien geprüft werden, was bei wiederholtem Zugriff auf die gleichen Dateien einen erheblichen Performance-Gewinn bedeutet. Es ist allerdings möglich, daß eine mit einem brandneuen Virus infizierte Datei, der zum Zeitpunkt des Downloads dem Virenscanner noch nicht bekannt war, im Cache verbleibt.
- *Sichere Dateien nicht scannen*: Dateien oder Dateitypen, die als sicher eingestuft werden, werden vom Virenscan ausgeschlossen. Das birgt prinzipiell die Gefahr in sich, daß über eine Fehleinschätzung, was eine „sichere Datei" ist, sich doch eine infizierte Datei einschleicht. Deshalb sollte man die Liste der „sicheren Dateien" klein halten. Eine Überprüfung auf den MIME-Typ liefert generell eine wesentlich höhere Treffersicherheit als eine Überprüfung auf die nackte Dateiendung.

In der Praxis kombiniert man in der Regel alle drei Möglichkeiten, dabei bringt sicher der Einsatz eines Caching-Proxy vor einer Scan-Einheit den meisten Gewinn, denn bereits gespeicherte Dateien können direkt geliefert werden, der Download sowie die Entscheidung, ob es sich um eine „sichere Datei" handelt und der Virenscan wegfallen kann, muß erst gar nicht getroffen werden. In Abbildung 6.5 ist das Grundprinzip für den Virenscan beim Websurfen dargestellt.

Der Anwender greift von seiner Workstation aus zunächst auf dem Caching-Proxy zu. Bereits dort abgelegte Dateien sind mit lokaler Netzwerkgeschwindig-

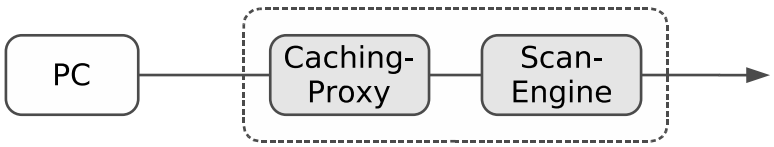

Abbildung 6.5: Grundprinzip für den Proxyscan

keit direkt verfügbar, eine Verzögerung durch Download oder Virenscan entfällt hier.

Der Proxy wird so eingestellt, daß er nicht vorhandene Dateien immer über die Scan-Engine holt und keine andere Wege einschlägt. Die Scan-Engine überprüft zunächst, ob es sich um einen Datei-Typ handelt, der als „sicher“ gilt (in der Regel die MIME-Typen `text/html`, `text/plain`, `image/gif`, `image/jpeg` und `image/png`). Ist das der Fall, entfällt der Virenscan. Alle anderen Dateien werden auf Viren untersucht, bevor diese an den Browser weitergereicht werden.

In den meisten Fällen bietet es sich an, Caching-Proxy und Scan-Engine auf dem gleichen Rechner unterzubringen. Insbesondere dann, wenn die Scan-Engine in einer Kombination mit dem Webserver Apache eingesetzt wird, läßt sich die Scan-Engine so konfigurieren, daß ein Zugriff nur von `localhost` aus möglich ist.

Virenscan und Browsergeduld

Findet der Scanner einen Virus, ist es sinnvoll und nützlich, den Anwender zu informieren. Nur ist das nicht so einfach. Angenommen, der Anwender lädt eine gezipptes Archiv herunter. Der Webserver meldet dem Browser den MIME-Typ `application/zip`. Bei anderen Dateien eben den passenden MIME-Typ. Findet der Scanner einen Fehler, muß er eine Info an den Anwender senden, die dieser im Browser lesen kann und nicht abspeichert, also etwa eine HTML-Seite. Eine HTML-Seite hat aber den MIME-Typ `text/html`. Der MIME-Typ wird im HTTP-Header vor der eigentlichen Datei übertragen. Die Scan-Engine muß also zuerst wissen, ob die Datei in Ordnung ist, bevor sie entscheidet, ob der Anwender die Datei bekommt, oder eine Fehlermeldung. Der Download muß also abgeschlossen und die Datei vollends gescannt sein. In der Zwischenzeit wartet der Browser ohne eine Antwort, was bei großen Dateien oft zu lange dauert. Der Browser denkt, die Verbindung ist abgebrochen, und beendet die Verbindung zum Proxy.

Die Lösung für das Timeout-Problem ist ein Kompromiß, den man akzeptieren muß, weil es keine andere Lösung gibt. Um den Timeout zu verhindern, sendet die Scan-Engine in regelmäßigen Abständen ein paar Bytes an den Anwender, damit der Browser nicht die Geduld verliert. Findet die Scan-Engine allerdings später einen Virus, gibt es keine Möglichkeit mehr, den Anwender zu benachrichtigen. Die Verbindung wird einfach abgebrochen.

Idealerweise setzt man daher eine Scan-Engine ein, die erst nach dem Ablauf einer bestimmten Zeit die ersten Datenbytes sendet. Solange das nicht geschehen ist, bleibt der Scan-Engine immer noch die Möglichkeit, den Anwender über einen vorhandenen Virus zu informieren. Setzt man dann noch einen Browser wie Mozilla ein, der längere Wartezeiten jenseits einer Minute in Kauf nimmt,

kann man mit dem Kompromiß ganz gut leben. Bei einer leistungsfähigen Internetanbindung und nicht allzugroßen Dateien ist das recht praktikabel.

Übersicht über die unterschiedlichen Lösungen

Die nachfolgende Übersicht erhebt keinen Anspruch auf Vollständigkeit. Über die Homepages von Squid und dem OpenAntiVirus-Projekt findet sich Linkseiten auf weitere Projekte:

`http://www.squid-cache.org/related-software.html`
`http://www.openantivirus.org/projects.php`

Mit Ausnahme des eigentlichen Virenscanners sollte die aufgeführte Software im wesentlichen „frei“ im Sinne des Open Source-Begriffes sein. Eine genaue Auskunft bietet die jeweilige Homepage der beschriebenen Software.

- *squid-vscan*
  `http://www.openantivirus.org/`
  `squid-vscan` ist ein Patch für Squid 2.3. Zum Einsatz kommt der OpenAntiVirus Scanner (OAV). Benötigt wird neben Squid 2.3 der `squid-vscan`-Patch und der OpenAntiVirus Scanner.
  Der Vorteil liegt in der direkten Integration in Squid, allerdings ist diese Lösung stark veraltet. Weder ist OpenAntiVirus auf dem neuesten Stand, noch gibt es einen Patch für neuere Squid-Versionen. Die Aktivitäten bezüglich des Patches scheinen eingestellt zu sein.
- *Internet Content Adaption Protocol ICAP*
  `http://www.i-cap.org/home.html`
  Das Internet Content Adaption Protocol ICAP (RFC 3507) ist ein relativ neues Protokoll, das von verschiedenen Herstellern von Virenscannern aufgegriffen wurde. Squid läßt sich dabei als Caching-Proxy und ICAP-Client einsetzen, als ICAP-Server dient ein kommerzieller Virenscanner, der dieses Protokoll unterstützt. Für Squid 2.5 oder neuer ist ein Patch erhältlich, der unter `http://squid.sourceforge.net/icap/` heruntergeladen werden kann (vermutlich ab 3.0 nicht mehr erforderlich). Squid wird damit zum ICAP-Client. Als ICAP-Server setzt man einen kommerziellen Virenscanner mit ICAP-Serverfunktionalität ein.
  ICAP wird zunehmend von Herstellern von Antivirus-Software unterstützt, dadurch hat man eine fast freie Wahl des Antivirus-Scanners. Der ICAP-Client für Squid wird offiziell vom Squid-Team gepflegt und in eine der nächsten Releases integriert.
- *Viralator*
  `http://viralator.sourceforge.net/`
  Viralator besteht aus einem Skript und einem Redirector, über den Squid die gewünschte URL an das Skript weiterreicht. Das Viralator-Skript lädt die

Datei in ein eigenes Verzeichnis, die Datei wird dort nach Viren gescannt. Ist die Datei infiziert, erhält der User eine Warnmeldung, ist die Datei „sauber“, wird ein HTTP-Redirect an den Browser gesendet, der dann die gescannte Datei von der neuen Lokation – nämlich dem Scanverzeichnis – herunterladen kann. Für den Zugriff per HTTP auf das Scanverzeichnis wird Apache benötigt.
Als Redirector kommt primär Squirm zum Einsatz, eine Konfiguration mit SquidGuard ist zwar ebenfalls möglich, aber schlechter dokumentiert und weniger getestet. Der Download findet über `wget` statt.
Vorteil dieser Lösung ist sicher die breite Unterstützung vieler Virenscanner, nachteilig dagegen ist die sehr aufwendige Konfiguration. Außerdem lassen sich Dateitypen nur nach der Endung unterscheiden.

- *`Apache::ProxyScan`*
  `http://search.cpan.org/dist/Apache-ProxyScan/`
  Apache::ProxyScan ist ein Perlmodul, das sich in den Proxy-Modus bei Apache einklinkt. Dazu wird Apache als Parent Cache zu Squid eingesetzt. Das Modul wurde für Apache 1.3 entwickelt, Apache 2.0 ist vom Autor des Moduls nicht getestet. Im Prinzip können alle unter Linux lauffähigen Virenscanner verwendet werden, da der Virenscanner über ein Skript aufgerufen wird, das man selbst entsprechend anpassen kann.
  Die Konfiguration von Apache::ProxyScan ist relativ einfach. Das Modul ist in der Lage, neben einer Prüfung auf Dateiende auch eine Prüfung auf den MIME-Typ vorzunehmen, was mehr Sicherheit bietet.
  Nachteilig ist vielleicht, daß Apache im Proxy-Modus verwendet werden muß. Der Proxy-Modus läßt sich relativ einfach als Mailrelay und für andere Zwecke mißbrauchen, wenn man die Konfiguration nicht entsprechend absichert.

- *Apache2 und `mod_clamav`*
  `http://software.othello.ch/mod_clamav/`
  `http://www.clamav.net`
  Das Modul `mod_clamav` wird über den bei Apache2 neuen Filtermechanismus eingebunden. `mod_clamav` benutzt entweder direkt die Library des Virenscanners Clamav (dann muß kein externes Programm aufgerufen werden), oder kontaktiert den Clamav-Daemon `clamd`.
  Apache2 wird hier wieder im Proxy-Modus als Parent für Squid betrieben, das heißt, bei der Konfiguration von Apache muß man entsprechende Sorgfalt walten lassen, um einen Mißbrauch als offenes Relay vorzubeugen.
  `mod_clamav` benötigt die bei Apache2 mitgelieferten `mod_proxy*`-Module, die je nach Installation bei der Konfiguration erst noch hinzugefügt werden müssen. Außerdem wird die Clamav-Library `libclamav1` benötigt. `mod_clamav` wird selten von der Distribution mitgeliefert und muß daher meistens selbst übersetzt werden. Durch die direkte Einbindung in Apache

entfällt der Aufruf externer Skripte. Die Erkennung von „sicheren Dateien“ über MIME-Typen, URLs und Pattern ist möglich und funktioniert sehr gut. Als Nachteil mag vielleicht erscheinen, daß das Modul ausschließlich auf den Open Source Antiviren-Scanner Clamav beschränkt ist.

- *Apache2 und* `mod_vscan`:
  Gleiches Prinzip wie bei `mod_clamav`, es wird eben nur statt Clamav der ebenfalls freie Scanner OpenAntiVirus (OAV) benutzt.

In den nachfolgenden beiden Abschnitten wird die Konfiguration zum Virenscan beim Webzugriff mit dem Tandem Squid und Apache beschrieben, zum einen für das Perlmodul Apache::ProxyScan, zum anderen für das Apache-Filtermodul `mod_clamav`.

Warnhinweis: ein im Proxy-Mode betriebener Apache ist leicht zu mißbrauchen. Es muß daher unbedingt sichergestellt werden, daß Apache nur für den angestrebten Zweck genutzt werden kann. Zum einen läßt sich das über den Socket erreichen. Man erlaubt über die Listen-Direktive nur Zugriffe von `localhost` *aus. Zum zweiten sollte man zusätzlich über Paketfilterung dafür sorgen, daß von keinem externen Host aus der Proxy-Port von Apache erreichbar ist.*

6.3.1 Apache::ProxyScan

Zunächst wird Apache 1.3 einschließlich des Perl-Modules `mod_perl` benötigt. Beide sind bei allen gängigen Linux-Distributionen vorhanden, `mod_perl` muß aber meist explizit für die Installation ausgewählt werden.

Das Perl-Modul `Apache::ProxyScan` ist in keiner Distribution vorhanden und muß daher über das CPAN (Comprehensive Perl Archiv Network) geholt und installiert werden. Dazu gibt es zwei Möglichkeiten: die manuelle Installation oder die CPAN-Variante.

Manuelle Installation

Die Module finden sich unter `http://www.cpan.org`.

Als User `root` Archiv auspacken, übersetzen und installieren:

```
% cd /usr/local/src/Perl
% tar xvzf Apache-ProxyScan-0.31.tar.gz
% cd Apache-ProxyScan-0.31/
% perl Makefile.PL
% make ; make check ; make install
```

Gegebenenfalls fehlende Perl-Module sollten von der Distribution nachinstalliert werden, soweit vorhanden, andernfalls kann man diese auf den hier beschriebenen Weg natürlich ebenfalls nachinstallieren.

Kapitel 11

Virtuelle Private Netzwerke (VPN)

Unterschiedliche Standorte lassen sich heute über zwei verschiedene Klassen von Netzwerken verbinden: private Netzwerke und öffentliche Netzwerke. Ein privates Netzwerk entsteht, indem man auf Leitungstypen zurückgreift, die ausschließlich vom eigenen Unternehmen benutzt werden, etwa Wählleitungen wie ISDN oder Festverbindungen. Man mietet sich von Providern Leitungen und erstellt eine eigene Infrastruktur (mittels Routern), die auch selbst betrieben werden muß.

Die andere Möglichkeit ist, auf bereits bestehende, öffentlich benutzbare Infrastrukturen zurückzugreifen. Um zwei Standorte über das Internet verbinden zu können, benötigt man im Prinzip nur zwei offizielle IP-Adressen, um mit der jeweiligen Gegenstelle am anderen Standort kommunzieren zu können.

Das Internet als Transportweg hat gegenüber dem privaten Netzwerk mehrere Vorteile:

- Durch die gemeinsame Nutzung von ein und demselben Leitungsweg durch viele Benutzer ist eine Verbindung über das Internet in der Regel viel kostengünstiger.
- Das Internet ist an sich als vielmaschiges Netzwerk redundant ausgelegt. Bei einem privaten Netzwerk muß man Redundanzen aufwendig selbst herstellen.
- Der Löwenanteil der Verwaltung und des Betriebs des Netzwerkes wird von Providern erledigt.

Allerdings haben öffentliche Netzwerke wie das Internet auch Nachteile:

- Der Transportweg ist nicht sicher. Es besteht die Möglichkeit, daß andere den Datenverkehr mitlesen.

- Feste Bandbreiten können normalerweise nicht garantiert werden (außer, wenn alle Leitungen von ein und demselben Provider zur Verfügung gestellt werden).

Verschlüsselt man den Datenverkehr über die öffentlichen Netze und sorgt für eine vernünftige Authentifikation der Gegenstellen, entsteht ein „Virtual Private Network“ VPN. In den meisten Fällen überwiegen die Vorteile eines VPNs, insbesondere der Kostenvorteil, deshalb breitet sich die VPN-Technologie rasch aus.

Grundlegende Eigenschaft eines „echten VPN“ ist die völlige Unabhängigkeit von den unteren OSI-Schichten 1-3. Es spielt keine Rolle, wie die IP-Pakete transportiert werden, es gibt keinen festen Weg.

Es gibt andere Technologien, die auch zu den VPN-Technologien gezählt werden oder mit denen sich virtuelle private Netzwerke erstellen lassen. Ihnen ist aber die fehlende Unabhängigkeit auf den unteren OSI-Layern gemeinsam. Der Provider legt die Route fest, die ein Paket nimmt, und verhindert so den Zugriff von „außen“ auf das virtuelle private Netzwerk. Zu diesen Technologien gehören unter anderem das Protokoll L2F (Layer 2 Forwarding Protocol) oder das L2TP (Layer 2 Transport Protocol). Aber auch Frame Relay läßt sich von Providern für die Erstellung eines VPNs einsetzen. Wir gehen aber auf solche Technologien nicht näher ein.

Für ein IP-basiertes VPN gibt es verschiedene Möglichkeiten, Daten verschlüsselt zu übertragen. Das anerkannt sichere und sehr weit verbreitete IPsec (IP Security) ist auf fast allen Plattformen – auch auf hardware-basierten Routern – verfügbar. Man kann IPsec als Standard für IP-basierte VPNs betrachten. Dieses Kapitel beschränkt sich daher auf auf die Grundlagen und Anwendung von IPsec.

Nach einem kurzen Überblick über die Grundlagen gehen wir auf die Installation von IPsec unter Linux ein. Verwendet wird das frei verfügbare, auf FreeS/WAN basierende strongSwan.

11.1 IPsec-Grundlagen

Aufgabe von IPsec ist die Herstellung von Vertraulichkeit und Integrität von IP-Verbindungen. Das eigentliche IPv4-Protokoll besitzt keine Mechanismen, die die beiden Grundwerte der IT-Sicherheit garantieren können. Grob betrachtet besteht IPsec aus vier Bereichen:

- *Sicherheitsprotokolle*: die Basis von IPsec bilden die beiden Sicherheitsprotokolle „Authentication Header (AH)“ und „Encapsulating Security Payload (ESP)“.

- *Security Association (SA)* : Verbindungen werden über Security Associations abgebildet. Jedes Sicherheitsprotokoll benötigt seine eigene Security Association. SAs sind zeitlich begrenzt. Nach Ablauf des Gültigkeitszeitraumes müssen diese wieder neu verhandelt werden.
- *Schlüsselmanagement*: Für die Authentifikation und Verschlüsselung wird mit verschiedenen Schlüsseln gearbeitet. Neben der Möglichkeit, Schlüssel auch manuell auszutauschen, gibt es das „Internet Key Exchange (IKE)-Protokoll“, das den automatischen Schlüsselaustausch erlaubt.
- *Verschlüsselungsalgorithmen*: IPsec kennt eine Reihe verschiedener Algorithmen, die aber nicht immer alle implementiert sein müssen. Von der Qualität des Algorithmus hängt letztenendes auch die Sicherheit der Verbindung ab.

11.1.1 Sicherheitsprotokolle

Die beiden Sicherheitsprotokolle „Authentication Header (AH)“ und „Encapsulating Security Payload (ESP)“ erfüllen unterschiedliche Aufgaben und können einzeln unabhängig voneinander, aber auch zusammen eingesetzt werden.

- *Authentication Header (AH)*: will man den Absender eindeutig identifizieren und die Integrität der Daten sicherstellen, ohne auch eine Verschlüsselung zu verwenden, dann ist das eine Aufgabe für AH. Es gibt durchaus Anwendungen, die nicht verschlüsselt werden dürfen (zum Beispiel aufgrund gesetzlicher Vorschriften in einigen Staaten).
  AH kann man als eine digitalen Unterschrift der Datenpakete betrachten.
- *Encapsulating Security Payload (ESP)*: Schwerpunkt bei ESP ist die Vertraulichkeit, also die Verschlüsselung der Daten. Darüber hinaus verfügt ESP ebenfalls über Mechanismen zur Authentifikation. Kommt ESP zum Einsatz, kann man deswegen in den meisten Fällen auf AH verzichten.

Beide Protokolle, sowohl AH als auch ESP verfügen über Sicherheitsmechanismen, die sogenannte Replay-Attacken verhindern. Bei einer Replay-Attacke wird

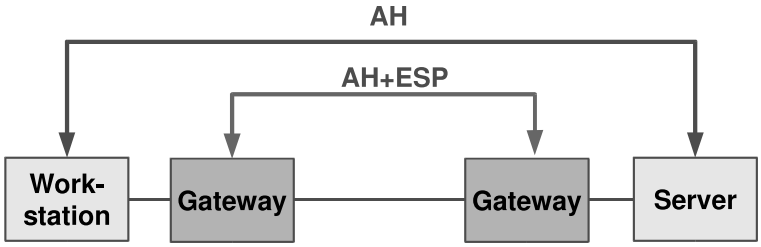

Abbildung 11.1: IPsec Authentication Header und Encapsulating Security Payload

ein Stück des Datenstromes mitgeschnitten und dann dem Zielsystem wieder vorgespielt.

Beide Protokolle können zusammen eingesetzt werden. Da jedes Protokoll für sich eine eigene Security Association erstellt, können auch die Endpunkte unterschiedlich sein. Abbbildung 11.1 zeigt ein Szenario, in dem zwei Rechner eine Ende-zu-Ende-Verbindung über AH abbilden, und der Bereich zwischen den beiden Gateways zusätzlich verschlüsselt läuft.

Die Sicherheitsprotokolle werden von IP wie ein Layer-4-Protokoll (Beispiel TCP oder UDP) behandelt. Das Protokollfeld des IP-Headers (siehe Abschnitt B.1) enthält dabei die dazugehörige Protokollnummer, 50 für ESP, 51 für AH. Werden ESP und AH parallel eingesetzt, kommt der AH-Header immer vor dem ESP-Header (siehe auch Abbildung 11.2). In diesem Falle schützt AH zusätzlich auch den ESP-Header.

11.1.2 Transport- und Tunnelmode

IPsec kennt zwei verschiedene Verbindungsmodi: den „Transport Mode“ und den „Tunnel Mode“. In Abbildung 11.2 sind die beiden Modi dargestellt.

Der „Transport Mode“ stellt eine Ende-zu-Ende-Verbindung zwischen den beteiligten Endgeräten – etwa Workstations – her. Da das Endgerät erst die Daten verschlüsselt (ESP) und/oder signiert (AH), und erst danach mit einem „IP-Umschlag“ versieht und adressiert, enthält der IP-Header den direkten Adressaten, und die Daten können auf dem normalen Wege zugestellt werden.

Der „Tunnel Mode“ stellt eine geschützte Verbindung zwischen zwei Gateways her. Im jeweiligen lokalen Netzwerk wird mit einfachen IP-Paketen gearbeitet.

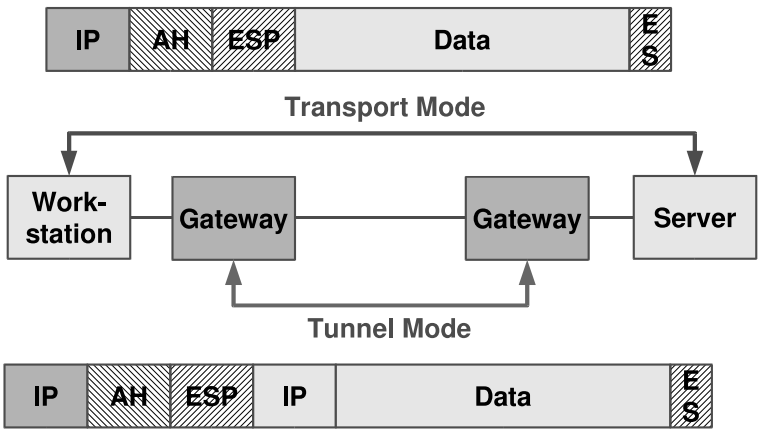

Abbildung 11.2: IPsec Tunnel und Transport Mode bei IPv4

Sobald ein Paket den Weg über das Gateway nimmt, wird das IP-Paket eingekapselt, verschlüsselt (ESP) und/oder signiert (AH) und anschließend an das zweite Gateway weitergeleitet. Der ursprüngliche IP-Header bleibt unangetastet in der Einkapselung erhalten, das Paket erhält insgesamt einen neuen Header, der aber nur als Zielbestimmung das zweite Gateway adressiert. Auf dem zweiten Gateway werden die Daten wieder ausgepackt und im ursprünglichen Zustand in das lokale Netzwerk hinter dem zweiten Gateway weitergeleitet.

Ideal wäre natürlich eine Ende-zu-Ende-Authentifikation mit Verschlüsselung. Allerdings ist der Aufwand erheblich. Jede einzelne Workstation, jeder Server, ja jedes Endgerät, das über IP kommuniziert, müßte mit Authentifikationsdaten versehen werden und ebenfalls über Informationen zu den Authentifikationsdaten aller anderen Netzwerkgeräte verfügen. Hier gibt es noch keine standardisierte Lösung, daher ist in der Regel ein umfassender Einsatz des Transport Mode nicht möglich. Ansätze hierzu bietet zumindest FreeS/WAN unter Linux mit der „Opportunistic Encryption“, wir gehen im Abschnitt 11.6 kurz darauf ein.

Der Tunnel Mode bietet hier einen guten Kompromiß zwischen Aufwand und Sicherheit. Ausgehend davon, daß die jeweiligen lokalen Netzwerke sicher sind, werden die unsicheren Strecken verschlüsselt und/oder signiert. Der Verwaltungsaufwand für die Authentifikationsdaten hält sich in Grenzen, da ja nur die Schlüssel für die beiden Gateways ausgetauscht werden müssen. Wie groß die Netzwerke hinter den Gateways sind, spielt dabei überhaupt keine Rolle.

Die unsichere Strecke ist in vielen Fällen das Internet. Das muß aber nicht unbedingt so sein. Anwendungen sind auch firmenintern denkbar, wenn etwa zwei Bereiche mit hohen Sicherheitsanforderungen über ein gemeinsames Backbone des Unternehmens kommunizieren, der Datenverkehr aber geschützt werden muß.

11.1.3 Authentifikation, automatischer Schlüsselaustausch

Bei IPsec werden zwei verschiedene Klassen von Schlüsseln eingesetzt: einmal Schlüssel für die Authentifikation, zum anderen Schlüssel für den sicheren Austausch von Daten (Signatur und/oder Verschlüsselung).

Für die Authentifikation kommen mehrere Mechanismen in Betracht:

- *Shared Secrets*: hier wird ein gemeinsamer, geheimer Schlüssel vereinbart, den beide Seiten über einen sicheren Kanal austauschen müssen. Gerät der geheime Schlüssel in falsche Hände, kann eine Verbindung vom Verbindungsaufbau an abgehört werden.
- *Public Key*: Public Key-Mechanismen haben die Eigenschaft, das jeweils nur ein öffentlicher Schlüssel ausgetauscht werden muß, während der private Schlüssel nie den Ursprungsort verläßt. Neben der Einsparung eines sicheren Kanals hat dieses Verfahren auch einen weiteren Vorteil: so ein Schlüs-

selpaar wird nur einmal je Endseite benötigt (N-mal). Bei Shared Secrets ist für jede Verbindung ein geheimer Schlüssel nötig ($\frac{N\cdot(N-1)}{2}$).
Beispiel: für 100 Knoten benötigt man bei Public-Key-Verfahren 100 Public-Private Key-Paare, bei Shared Secrets sind das aber schon $(\frac{100\cdot 99}{2}) = 499,5$

- *Zertifikate*: Zertifikate sind im Prinzip Public Keys, die von einer zentralen Autorität, einer sogenannten „Certification Authority CA“ signiert werden. Jetzt muß nicht mehr geprüft werden, ob der Public Key der Gegenseite aus halbwegs vertrauenswürdigen Quellen stammt, sondern es wird nur noch überprüft, ob die Unterschrift der CA korrekt ist. Die Prüfung der Glaubwürdigkeit des Public Keys wird somit auf die „Certification Authority“ verlagert.

Nach der Authentifikation werden neue Schlüssel für die eigentliche Verschlüsselung beziehungsweise Signatur vereinbart. Diese können manuell festgelegt und vorab ausgetauscht werden, besser aber ist es, diese in regelmäßigen Abständen automatisch neu generieren zu lassen. Eine regelmäßige Erneuerung hilft, Einbruchsversuche und Replay-Attacken nach Möglichkeit ganz auszuschalten.

Die Schlüssel für den Datenaustausch sind in der Regel symmetrisch, da bei symmetrischer Verschlüsselung wesentlich weniger Rechenaufwand erforderlich ist als bei asymmetrischen Schlüsseln (Public Key Paare, etwa für die Authentifikation).

Für den Vorgang der Authentifikation und den regelmäßigen Austausch der Schlüssel wird das „Internet Key Exchange“-Protokoll eingesetzt. Der gesamte Vorgang ist recht komplex. Zunächst wird ein sicherer Kanal (eine Security Association) für den Austausch von IPsec-Parametern erstellt. Dazu wird das „The Internet Security Association and Key Management Protocol“, kurz ISAKMP eingesetzt. ISAKMP ist als schlüsselunabhängiges Framework zu verstehen, das nur beschreibt, wie ein sicherer Schlüsselaustausch vor sich gehen soll.

IKE baut in einer ersten Phase die ISAKMP SA auf (SA: Security Assossiation). Über diese ISAKMP SA werden dann in einer zweiten Phase die Parameter für die IPsec SA ausgetauscht. Zu beachten ist, das die ISAKMP SA zwar bidirektional, eine IPsec SA nur unidirektional arbeitet. IPsec SA werden deshalb in der zweiten Phase immer paarweise erstellt.

Sobald die zweite Phase abgeschlossen ist, sind die IPsec SAs erstellt und Daten können über ESP und/oder AH ausgetauscht werden. Während ESP und AH ein eigenes Protokoll der OSI-Schicht 4 darstellen, verwendet IKE das wohlbekannte UDP (Port 500).

11.2 IPsec unter Linux

Für die Implementierung von IPsec unter Linux muß man zwischen Kernelversion 2.4 und 2.6 unterschieden. Bei Kernelversion 2.4 gibt es standardmäßig im Kernel kein IPsec. Um dem Kernel 2.4 IPsec beizubringen, muß der Kernelcode gepatcht werden. Die gängigste Implementierung ist hier FreeS/WAN beziehungsweise deren Nachfolger strongSwan und Openswan. Wie FreeS/WAN bestehen strongSwan und Openswan aus zwei Teilen: einmal bestehen sie aus einem Satz von Kernelpatches für Kernel 2.4, um diesem IPsec beizubringen. Außerdem enthalten sie eine Sammlung von Programmen (auch „User Space Tools“ genannt), über die die IPsec-Fähigkeiten des Kernel dann genutzt werden können und die darüber hinaus auch eine Reihe von Verwaltungsoperationen übernehmen (IPsec- Verbindungsauf- / --abbau, Schlüsselmanagement, etc.).

Im Kernel 2.6 ist IPsec bereits integriert. Hier stammt die IPsec-Fähigkeit nicht von FreeS/WAN ab, das es nie in den Standard-Kernel geschafft hat, sondern von einem konkurrierenden Projekt namens KAME. Letztenendes gab wohl die bessere IPv6-Fähigkeit und die sauberere Codebasis KAME den Vorzug vor den FreeS/WAN-Patches.

Nichtsdestotrotz lassen sich die Anwendungsprogramme (User Space Tools) von FreeS/WAN, Openswan und strongSwan auch unter Kernel 2.6 nutzen. Das lästige Kernelpatchen entfällt, die Anwendungsprogramme setzen direkt auf die eingebauten IPsec-Funktionen des 2.6er Kernel auf.

Zum KAME-Projekt gehören die User Space Tools `setkey` und `racoon`, die mit den FreeS/WAN-Anwendungsprogrammen konkurrieren. Auf diese Programme wird nicht weiter eingegangen, deren Anwendung ist ausschließlich auf Kernel 2.6 beschränkt. Die FreeS/WAN-Programme dagegen lassen sich unter Kernel 2.4 und 2.6 identisch konfigurieren.

11.2.1 Kernel 2.4 oder 2.6?

Das eigentliche Manko bei Kernel 2.4 ist, daß man ihn patchen muß, um IPsec einsetzen zu können. Kernel 2.6 bietet hier die IPsec-Fähigkeit bereits integriert an. Allerdings muß man auch dagegen halten, daß IPsec mit FreeS/WAN und Konsorten bereits eine lange Tradition unter 2.4 hat, während die Implementierung in 2.6 relativ neu und noch in Entwicklung befindlich ist. Einen bewährten Patch an einem bewährten Kernel auszuführen ist daher kein Nachteil gegenüber einem möglicherweise noch nicht ausgereiften 2.6er Kernel.

Umgekehrt hat Kernel 2.6 für den Einsatz von IPsec einige Beschränkungen:

- *Kein `ipsec0` Device*: der Patch für Kernel 2.4 stellt ein eigenes Device für den IPsec-Tunnel zur Verfügung. Damit läßt sich bei einer Paketfilterung leicht unterscheiden, ob ein Paket aus dem Tunnel kommt, oder nicht. Bei nati-

Kapitel 12

VPN-Gateway für Windows-Clients

Hat man ein VPN-Gateway unter Linux erst einmal am Laufen, dauert es oft nicht lange, bis sich der erste Windows-Benutzer meldet und ebenfalls VPN-Zugang zum internen Netzwerk haben möchte. Von der technischen Seite her gesehen ist das nichts anderes als der Zugang für einen Linux-VPN-Client. Im Detail gibt es aber mehrere Varianten, die mal auf der Seite des Linux-Gateways, mal auf der Seite des Windows-Clients einen höheren Konfigurationsaufwand nach sich ziehen.

Auf Protokollseite bietet Windows zwei Möglichkeiten:

- natives IPsec, oder
- L2TP („Layer 2 Tunneling Protokoll“) über IPsec.

Bei letzterer Variante wird zwar IPsec zur Verschlüsselung genutzt, darauf läuft jedoch das L2TP-Protokoll, das die zusätzlichen Möglichkeiten eines PPP-Protokolls bietet wie eine echte Userauthentifikation, die Zuweisung von IP-Adressen, DNS- und WINS-Servern.

Aus Sicht des Linux-Gateways ist natives IPsec die einfachere Variante. Am Linux-Gateway richtet man ganz normal den Zugang für Roadwarrior in Kombination mit Zertifikaten ein. Auf der Windowsseite ist ebenfalls eine vergleichbare Konfiguration erforderlich, die Dank eines Tools von Marcus Müller einem die aufwendige Konfiguration von Sicherheitsrichtlinien abnimmt, man erstellt einfach eine Konfigurationsdatei, die sich sehr stark an der strongSwan-Syntax orientiert.

Die IPsec/L2TP-Variante ist die von Microsoft favorisierte Version. Wenn man unter Windows mit dem Netzwerkassistent eine „Verbindung mit einem privaten Netzwerk über das Internet“ herstellt, ist damit immer IPsec/L2TP gemeint. Am Windows-Client ist die Konfiguration deutlich einfacher, weil keine zusätzliche Software installiert werden muß, die Einrichtung findet sozusagen mit Windows-

Bordmitteln statt. Am Gateway dagegen ist zusätzlich zur IPsec-Konfiguration ein L2TP-Daemon und der PPP-Daemon erforderlich.

Die nachstehende Übersicht von Vor- und Nachteilen bezieht sich auf das IPsec-Tool von Markus Müller bei nativem IPsec, beziehungsweise auf den Einsatz des l2tpd-Daemons (`www.l2tpd.org`) bei IPsec/L2TP und stellt nur eine Zusammenfassung dar. Ein sehr ausführlicher Vergleich findet sich unter:

`http://www.jacco2.dds.nl/networking/freeswan-l2tp.html`

Vorteile bei nativem IPsec

- strongSwan-like Konfiguration am Windows-Client.
- Standard-Konfiguration am Gateway (nur strongSwan).
- Kein Protokoll-Overhead.

Nachteile bei nativem IPsec

- Umständliche Einwahl: erst RAS-Verbindung aufbauen, dann das IPsec-Tool starten, dann einen Ping auf die Gegenstelle, damit der Tunnel aufgebaut wird.
- Keine virtuelle IP-Adresse, Client erhält die dynamische IP des Providers (schwierigere Paketfilterung).
- Nicht NAT-T fähig.

Vorteile von IPsec/L2TP

- Eingebauter Client bei Win2k/XP, für andere Microsoft-Betriebssysteme gibt es einen kostenlosen L2TP-Client von Microsoft.
- Userfriendly Dialin: Verbindungsauf-/abbau über einen Vorgang (RAS-Verbindung wird automatisch mit auf- und abgebaut).
- Einfache Konfiguration über Wizard (Internet-Verbindung über privates Netzwerk).
- Virtuelle IP-Adresse zuweisbar (sogar userabhängig!)
- Unterstützt NAT-T

Nachteile von IPsec/L2TP

- Protokoll-Overhead: `IP->IPsec->L2TP->PPP`, dadurch langsamer.
- erhöhter Konfigurationsaufwand am VPN-Gateway (neben strongSwan auch PPP- und L2TP-Konfiguration).

Für IPsec/L2TP spricht der „eingebaute Windowsclient“ und die Möglichkeit, virtuelle IP-Adressen zuzuweisen. Nachfolgend werden beide Varianten beschrieben: sowohl natives IPsec unter Zuhilfenahme des Tools von Marcus Müller, als auch der Einsatz von IPsec/L2TP. Bei beiden kommen X.509-Zertifikate zur Authentifikation zum Einsatz.

Unter Windows kann man außer mit X.509-Zertifikaten auch meistens mit einem Preshared Key arbeiten. Bei einer größeren Anzahl ist der Aufwand für die Verwaltung von Preshared Keys jedoch recht hoch, und X.509-Zertifikate sind auf Windowsseite auch nicht aufwendiger zu konfigurieren, wenn man sich an die Reihenfolge in der nachfolgenden Beschreibung hält. Wir gehen daher ausschließlich auf die Verwendung von X.509-Zertifikaten ein.

12.1 Mit Windows ans Internet: Safety first!

Bei den ersten Gehversuchen hat der Autor selbst unangenehme Erfahrungen mit der Einwahl von Windows 2000 ins Internet gemacht. Der Rechner, eine nackte Win2k-Installation mit Service-Pack 2, dient ausschließlich zu Testzwecken. Die Einwahl erfolgte via RAS/ISDN über eine Call-by-Call Nummer ins Internet. Keine 30 sec. nach der Einwahl erfolgte bereits der Absturz von `svchost.exe`, ein Programm, das sich mit um die RAS-Verbindung kümmert. Die RAS-Verbindung ist dann weder zu trennen (nur physikalisches „Stöpsel raus“ hilft hier noch), noch zu konfigurieren. Neuboot ist angesagt, und spätestens nach 30 sec. beginnt das Spiel von vorne.

Die Ursache: einer der Würmer, die den RPC-Bug unter einigen Betriebssystemen ausnutzen, versucht eine Infektion über Port 135. An manchen Tagen kann man an einem ganz normalen DSL-Anschluß (natürlich mit einer Linux-Firewall gesichert) unmittelbar nach der Einwahl mehr als 30 Hits pro Minute (!) feststellen, alles Internet-Nutzer, die sich nicht um die Absicherung ihres Systems gekümmert haben.

Daher sollte man beim Windows-Client einige Spielregeln beachten, bevor dieser die Verbindung zum Internet aufnimmt:

- *Personal Firewall verwenden.*
  Eine restriktive Handhabung der erlaubten Verbindungen über eine Personal Firewall ist unbedingt notwendig, um direkte Netzwerkangriffe zu vermeiden. Es tauchen immer neue Exploits zu bisher unbekannten Sicherheitslücken auf, denen aber allen gemeinsam ist, daß sie versuchen, eine Netzwerkverbindung zum lokalen Client aufzubauen, bevor die Infiltration erfolgt. Unterbindet man zunächst alle Verbindungen, insbesondere diejenigen, die von Außen zum Client aufgebaut werden sollen, hat man eine rea-

le Chance, auch unbekannte, neue Würmer (die netzwerkbasierte Angriffe starten) fernzuhalten.
Allerdings ist eine Personal Firewall manchmal unbequem. Gerade bei der Installation neuer Programme oder dem Einsatz bisher noch nicht verwendeter Dienste kann die ständige Fragerei, ob man eine Verbindung auch wirklich erlauben will, nervig sein. Trotzdem: lieber sinnvolle Fragen beantworten und sich etwas Zeit nehmen ist immer noch besser, als sich bequem, aber ungefragt irgendwelche Viecher einzufangen.

- *Virenscanner installieren, regelmäßig aktualisieren und benutzen (!)*
  Nicht alle Infektionen eines Rechners geschehen über eine direkte Netzwerkverbindung von außen. Attachements von E-Mails oder normal über das Web gezogene Dateien könnten verseucht sein. Hier hilft nur ein zusätzlicher Virenscanner, denn eine Personal Firewall ist nicht in der Lage, den Inhalt von Dateien auf schädliche Funktionen zu prüfen. Kombinationsangriffe in der Vergangenheit haben gezeigt, daß manchmal auch ein per E-Mail oder Download eingefangener Virus eine Personal Firewall deaktivieren kann, was dann wiederum für direkte Angriffe aus dem Internet Tür und Tor öffnet.

- *Alle aktuellen Patches einspielen.*
  Unabhängig von den anderen Sicherungsmaßnahmen sollte man immer alle aktuellen, verfügbaren Patches am Client einspielen. Man ist nie davor gefeit, den Virenscanner falsch zu konfigurieren, mal versehentlich abzustellen oder es treten Probleme beim Aktualisieren der Virensignatur auf. Eine Personal Firewall hilft meist sehr viel, manchmal aber auch zuviel, so daß man beim Testen auch schnell mal in die Versuchung kommt, sie kurz abzustellen um auszuprobieren, ob man ein aufgetretenes Problem an der Firewall liegt oder nicht. Manchmal reichen wenige Sekunden ungeschützt, um das System zu infizieren.
  Nicht immer durchgehend, aber sehr oft verwenden neue Würmer Exploits, die auf einer älteren, schon länger bekannten Sicherheitslücke beruhen. Nichts ist peinlicher für einen Adminstrator als eine Infektion eines Systems, die man mit einem bereits 6 Monate alten Patch hätte vermeiden können.

Last but not least sollte man sich darüber im Klaren sein, das trotz aller Sicherheitsmaßnahmen am Client immer noch von dort Gefahren drohen können, weil die Sicherheitsmaßnahmen dort trotz aller Sorgfalt auch einmal versagen könnten. Das ist leider nie ganz auszuschließen.

Für das VPN-Gateway bedeutet dies: auch durch den relativ sicheren Tunnel nur Verbindungen nach drinnen erlauben, die unbedingt nötig sind. Man sollte generell alles verbieten, und nur die absolut notwendigen Dinge erlauben.

Eine gute Lösung wäre etwa, innerhalb eines Firmennetzwerkes nur auf einen Terminalserver Zugriff zu gewähren, und jeglichen Datenaustausch zwischen Server und Client zu verhindern.

Dort, wo etwa über Netzwerklaufwerk Dateien ausgetauscht werden müssen, kann man einen gesicherten Bereich bereitsstellen, zum Beispiel ein gesonderter Server in einem Grenznetz, der von innerhalb des Netzwerkes und via VPN-Tunnel erreichbar ist. Darüber lassen sich dann Daten mit dem VPN-Client austauschen, den direkten Zugriff auf interne Server könnte man ganz verbieten. Der spezielle Server muß gezielt überwacht und regelmäßig auf Viren, Trojaner und Würmer gescannt werden.

Entsprechende Protokollierung von unerlaubten Verbindungsversuchen am VPN-Gateway helfen, etwaige infizierte Clients schnell zu identifizieren, da viele Würmer, die zur Verbreitung auch das Netzwerk verwenden (also sich nicht ausschließlich auf die Infektion von Dateien verlassen), versuchen, permanent wohlbekannte Ports zum Verbindungsaufbau anzusprechen.

Die Darstellung hier ist keineswegs abschließend oder erschöpfend, sondern nur als Appetizer gedacht, um sich der Problematik bewußt zu werden und einmal ausführlich darüber nachzudenken.

In jedem Falle ist es ratsam, einschlägige Security-Informationen einzuholen und – wenn möglich – täglich zu beobachten. Neben den Webseiten der Virenhersteller ist die Security-Seite des Heise-Verlages[1] eine gute Adresse, da dort aus der aktuellen Nachrichtenlage alle Security relevanten Nachrichten übersichtlich zusammengestellt sind.

Eine wichtige, aber englische Webseite ist `www.securityfocus.com`, mit einem fast unerschöpflichen Fundus an aktuellen Nachrichten und sicherheitsrelevanten Artikeln.

12.2 Import der Zertifikate unter Windows

Der Import der Zertifikate ist für natives IPsec wie für IPsec/L2TP identisch. Für den Import müssen diese im PKCS 12-Format vorliegen. Die Erstellung von Zertifikaten einschließlich Export des PKCS 12-Formates wird in Abschnitt 11.8.1 beschrieben.

Der Import wird über die Management Konsole durchgeführt. Wer das Tool von Marcus Müller einsetzt (siehe Abschnitt 12.3), kann die Management Konsole mit einem Mausklick auf die Datei `ipsec.msc` öffnen und hat dann bereits die relevanten Plugins vorkonfiguriert. Für den manuellen Start der Management Kon-

[1] `http://www.heise.de/security/`

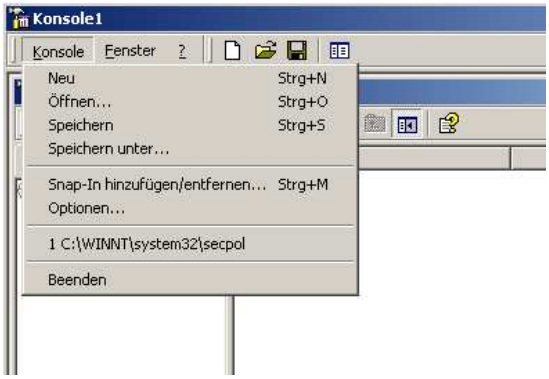

Abbildung 12.1: Managementkonsole `mmc` – „Konsole/Snap-In hinzufügen“

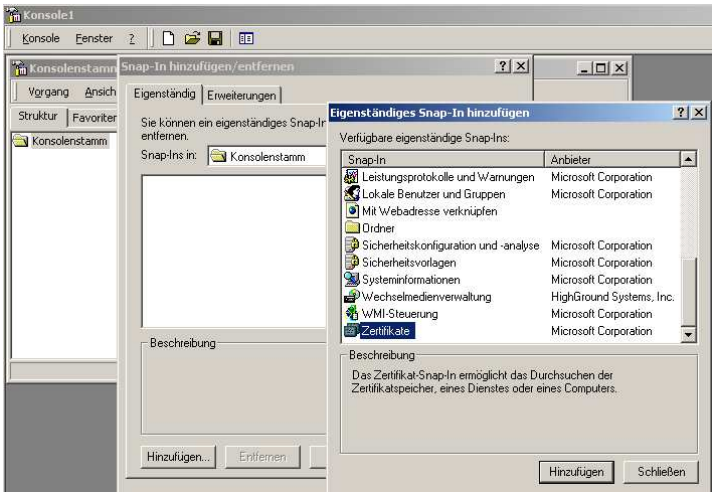

Abbildung 12.2: Managementkonsole `mmc` – „Zertifikate“

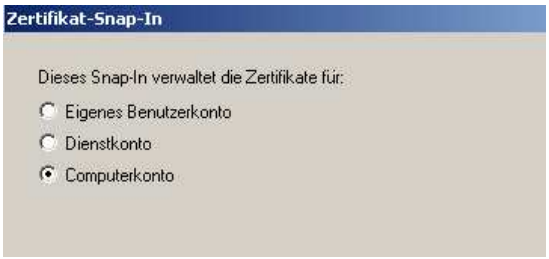

Abbildung 12.3: Managementkonsole `mmc` – „Snap-In verwaltet die Zertifikate für das Computerkonto“

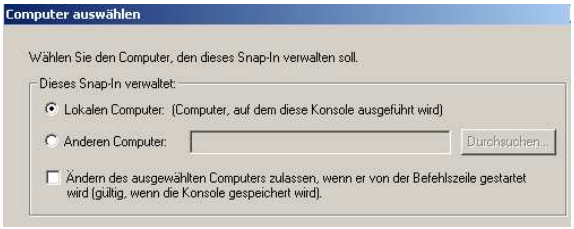

Abbildung 12.4: Managementkonsole `mmc` – „Dieses Snap-In verwaltet den lokalen Computer“

sole startet man unter „Start/Ausführen“ das Programm `mmc` und erhält eine Management Konsole ohne jegliches Plugin.

Unter „Konsole/Snap-In hinzufügen“ (Abbildung 12.1) wird dann das Snap-In „Zertifikate“ hinzugefügt (Abbildung 12.2).

Die beiden nächsten Fragen werden mit „Snap-In verwaltet die Zertifikate für das Computerkonto“ (Abbildung 12.3) und „Dieses Snap-In verwaltet den lokalen Computer“ (Abbildung 12.4) beantwortet. Im Konsolenstamm erscheint nach dem Beenden der Hinzufügen-Prozedur unter Konsolenstamm der Eintrag „Zertifikate“ (Abbildung 12.5). Mit einem rechten Mausklick auf „Eigene Zertifikate“

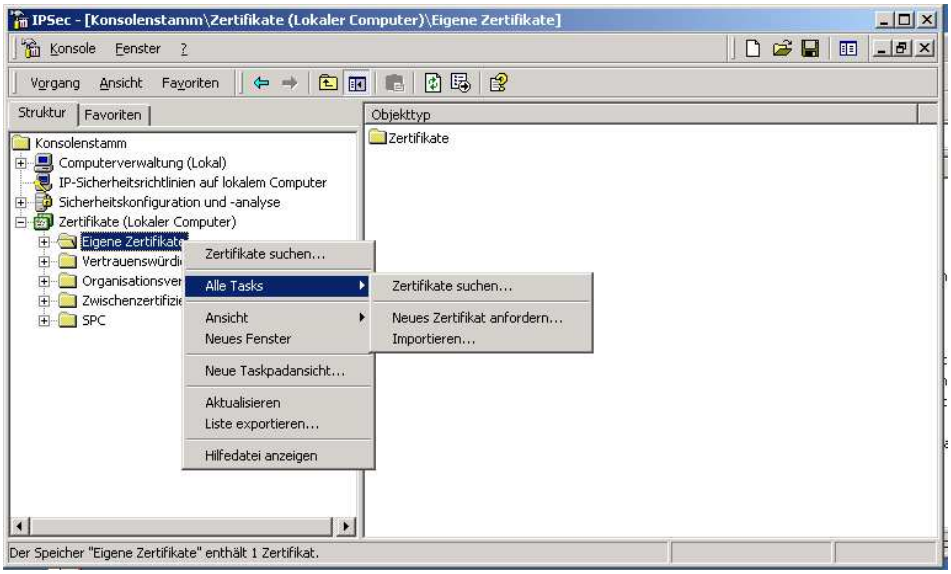

Abbildung 12.5: Start des Importvorganges: rechter Mausklick auf „Eigene Zertifikate“

Index

Symbole

.forward 180
.rhosts 200, 253
.shosts 618
/etc/iproute2/rt_tables . 525
/etc/ipsec.conf 505, 507–508
/etc/ipsec.d 543
/etc/ipsec.secrets ... 505
/etc/logfiles 440
`/etc/resolv.conf` 594
/etc/squid.conf 110

A

Abhören 29
Access Control List . 109, 111, 112
Account 197
ACL 109, 111, 112
Administratorrechte
  Angriff auf 197
AH . *siehe* Authentication Header
Aliases 180
Angriffe 9
  autonome Einheiten ... 18–21
  externe, aktive .. 15–17
  interne, aktive ... 17–18
anonymous 211
Anwendungsprotokoll .. 196
Apache 141
  AddDefaultCharset 148
  Proxy Mode .. 140, 141
  Zeichensatzprobleme .. 148–149
Apache2
  Filter 140, 144
  Konfiguration 146
Apache::ProxyScan 140–144
Application Layer 37
Application Level Gateway 53
Applikationsclient ... 196
Ausspähen von Daten 17
Authentication Header (AH) 492–494
Authentifikation
  starke 202
authorized_keys 617

B

Backdoor 16
Backup 405
Bastion Host . 54, 106, 345
  Dual Homed ... 56, 106
  Forwarding 106
  Single Homed . 56, 106
Benutzerverwaltung . 252
BIND 189
Blacklist 126
Brandschutzmauer 7
Broadcast Sturm 16
Browser 210
Brute Force 412
BSI 26, 33
Buffer overflow 204
Bundesamt für Sicherheit in der Informationstechnik . *siehe* BSI

C

CA ..... *siehe* Certificate Authority
CA-Zertifikat 537
Cache 107
Caching 105
Caching Protokoll ... 108
  HTCP 108
  ICP 108
Certificate Authority 227
Certificate Request .. 537
Certificate Revocation List 534, 537, 541–542, 546
Certification Authority .. 496, 534, 535, 537–539
Chain 69
Challenge 620
`chap-secrets` 581
Circuit Level Gateway 53
Clamav 140, 144
`clamd` 140
Common Name 536
config 618
Configure.help 65

Connection Tracking .. 78
Connection Tracking .. 66
Content Filter ... 110, 125
Content-Filter 177
Content-Filtering 361
CPAN 141, 142
CRL ..... *siehe* Certificate Revocation List

D

Dante ..... *siehe* SOCKS, Dante
Data Link Layer 36
Datensicherung .. 27, 405
`ddclient` 593–595
DDoS ..... *siehe* Denial of Service, Distributed
Debian
  file-rc 408
Default Policy
  Filterregeln 87, 266
Demilitarized Zone . *siehe* Entmilitarisierte Zone
Denial of Service 15
  Distributed 16
Diebstahl von Daten .. 17
Diebstahl von Daten .. 15
Digitale Unterschrift . 493
Distinguished Name 535, 544
  Wildcards 546
DMZ ..... *siehe* Entmilitarisierte Zone
DN .. *siehe* Distinguished Name
DNS ... 188–196, 317, 526
  BIND 189
  Caching only Nameserver 318, 340
  Filterregeln 270
  forwarder 195
  Iteration 190
  Lookup 189
  MX-Record 189
  nsswitch.conf 189
  Rekursion 189
  resolv.conf 189
  Resolver 189
  Spoofing 190
  TXT-Record ... 526, 528
  via SOCKS 317
  Zone-Transfer . 191, 195
DoD 39
Domain Name Service ... *siehe* DNS
DoS ..... *siehe* Denial of Service
DSL ... 314–315, 522, 582
  Zwangstrennung .. 572
Dual Homed Bastion Host 372
Dual Homed Hosts ... 56
Dynamic DNS .. 504, 520, 572, 593–595
dynamische IP . 520, 522, 571, 593–595
DynDNS 593–595

E

E-Mail
  sichere Konfiguration . 257
Einlaßkontrolle 8
Encapsulating Security Payload (ESP) 492–494
Entmilitarisierte Zone ... 53–54
environment 618
ESP . *siehe* Encapsulating Security Payload
Exploit 212

F

Fälschung von Information 15
File Integrity 423–435
File Transfer Protocol .... *siehe* FTP
file-rc 408
Filterregeln
  E-Mailabholung ... 325
  Ablaufdiagramm ... 70
  alle löschen 281
  automatischer Start 280, 406
  Destination-NAT .. 364
  DNS 270
  dynamische IP-Adressen 297
  FTP 273
  Grundoperationen .. 71
  HTTP 271
  ICMP 269
  ident 272
  Masquerading 305
  Matching 71
  Adressen 72
  Erweiterungen 74
  Fragmentierung 73
  ICMP-Erweiterung .. 77
  Interface 72
  MAC-Adressen 79
  Multiport-Erweiterung . 76
  Negation 73
  Owner-Prüfung 80
  Protokolle 72
  Rate 79
  TCP-Erweiterung 75
  UDP-Erweiterung ... 76
  unclean 81
  Zustandskontrolle ... 77
  POP3 271
  protokollieren 282
  Schlußregeln 275
  SMTP 271
  SOCKS 322
  Source-NAT 345
  Squid 361
  SSH 273
  SSL 272
  Targets 81
  testen 92
  xntp 274
Filterung
  Inhalt 106
Finger 240
Fingerprint 500

Fingerprinting ..... *siehe* OS-Fingerprinting
Firewall 7
äußerer Paketfilter . 354
collapsed 315
Design 35
DSL 314–315
einfacher Firewall 315–345
Grundlagen 35–61
innerer Paketfilter . 349
Konzepte 53–57
mit Grenznetz 345–372
Paketfilter 301–314
SOCKS-Client 336
SOCKS-Konfiguration . 336
SOCKS-Server 338
Standalone-Rechner ... 259–300
Firewall Chain ..... *siehe* Chain
Firewallscript
Plazierung 406
Forwarding 607
FQDN 124
Frühwarnsystem 410
Fragment Offset 41
Fragmentierung ... 41, 73
Frame Relay 492
FreeS/WAN 492, 497–499
`freshclam` 144
FTP 207–217
aktiv 208, 210
Masquerading 208
anonymous 211
Bounce Attacke ... 212, 474
Buffer Overflow ... 212
Charakteristik 207
control connection . 207
data connection
aktiv 208
data connection ... 207
Dateizugriffsrechte 211
DMZ 214
Filterregeln 273
Kommando
PASV 209
PORT 208
passiv 208, 210
PORT 212
PORT-Kommando . 211
Proxy 210, 215
Server 211, 214
extern 215
Server in Firewallumgebungen 214
Sicherheitsproblematik 210
SITE_EXEC Bug ... 212
WAREZ 213
Webbrowser 210
ftp-proxy 149–160
inbound 150–158
inetd.conf 158
Konfiguration 368
outbound 159
transparent ... 159–160
FWTK 174–176

G

Gauntlet Internet Firewall 174
Gefährdungspotentiale .. 14–21
Grenznetz ..... *siehe* Entmilitarisierte Zone

H

Hijacking 197
hosts.equiv 199, 253
HTCP 108
HTML 217–226
HTTP 217–226
1.1 122
Filterregeln 271
Header 123
HTTP-Code 134
301 134
302 134
httpf 177
Hyper Text Caching Protocol . *siehe* HTCP
Hypertext Markup Language ..... *siehe* HTML

I

ICAP 139
ICMP 47–48, 229, 600, 608
Destination Unreachable 231
Echo Reply 229
Echo Reply 48
Echo Request .. 48, 229
Filterregeln 269
Fragmentation needed 314
Parameter Problem 233
Redirect 233
Source Quench 232
Time Exceeded 233
ICMP-Scan 475
ICP 108
id_dsa 617
id_dsa.pub 617
ident
Filterregeln 272
Ident 120
IDS
seeIntrusion Detection . 412
IKE ... *siehe* Internet Key Exchange
IMAP 184
Implizite Verbindungen . 511, 529–533
abschalten 533
inetd 245, 254
inetd.conf 245, 254
Infrastruktur 29
init 407
insserv 244, 408
Installation
Paketauswahl 399
Vorbereitung 394
Integrität 10, 12, 423–424, 492

Interface 72
Internet
  Historische Entwicklung 38
Internet Cache Protocol .. *siehe* ICP
Internet Content Adaption Protocol ... 139
Internet Control Message Protocol (ICMP) *siehe* ICMP
Internet Key Exchange (IKE) 493, 496
Internet-Protokoll *siehe* IP
Internetdienste 179
  Charakteristik 179
  gebräuchliche 180
Intrusion Detection 411–423
  Informationsquellen ... 423
  netzwerkbasiert ... 412
IP 36, 40–42, 597–598
IP Tables Support (Kernel) 65
IP-Adressen
  dynamische 95, 289
IP-Datagram 41
IP-Options 42
IP-Sicherheitsrichtlinien (Windows) 572
ipchains 64
ipchains-restore 92
ipchains-save 92
iproute2 516
`iproute2` 524–525
IPsec 491–556
  Authentication Header (AH) 493–494
  Einkapselung 495
  Encapsulating Security Payload (ESP) 493–494
  Grundlagen ... 492–496
  IP-Protokollnummern . 494
  KAME 497
  Kernel 2.4 497–498
  Kernel 2.6 497–498
  Paketfilterung 550–556
  Schlüsselaustausch 495–496
  Transportmode 494–495
  Tunnelmode .. 494–495
  unter Linux ... 497–556
ipsec (Kommando) .. 509
IPsec passthrough ... 549
IPSEC.EXE 572–577
`IPsec.msc` 573
IPsec/L2TP 557–559, 577–591
`ipsec0` 497
iptables 64, 69–97
  logging 268
  Unterschiede zu ipchains
 Masquerading 626
  Unterschiede zu ipchains 625, 629
 Masquerading 628
 Paket-Routing 626
 stateful inspection .. 626
ISAKMP 496, 512
IT-Sicherheit 14
  Grundwerte 10
IT-Sicherheitspolitik .. 13

K

KAME 497
Katastrophenplan 28
Kernel
  kompilieren 64, 101
  make menuconfig .. 64
  Patch 97
  patchen 98
  Runtime-Konfiguration 66, 607
  Runtime-Parameter ... 265
Kernel 2.4 64–66
Kernel 2.6 66–68
Kernelkonfiguration 64–69
Kernelversion 63
Klartext-Übertragung 197
klogd 435
known_hosts 617
Konqueror 116

L

L2F 492
L2TP .. 492, 557, 577–591
  Destination Port ... 580
  Firewalling 584
L2TP/IPsec ..... *siehe* IPsec/L2TP
`l2tpd` 578, 580–582
LAN-to-LAN-Kopplung . 503–504, 508–512
Layer 2 Tunneling Protocol .. *siehe* L2TP
LDAP 109
`libclamav1` ... 140, 144
Linux Standard Base . 407
logcheck 442–447
Logfiles 435–462
logger 437
Login 252
logsurfer 447–462
  action 450
  continue 450
  delete 450
  dynamische Regeln 457
  exec 450
  Format 449
  ignore 450
  Konfigurationsdatei ... 448, 449
  Kontext 451–457
  match_regex 449
  non_match_regex . 449
  non_stop_regex ... 449
  open 450
  pipe 450
  report 450
  rule 451
  stop_regex 449

timeout 449
LSB 407

M

Mail 180–188
  Filterregeln
  interner Mailserver . 325
  Proxy 186
  Relay 181
Mail Delivery Agent . 180
Mail Transport Agent 180
Mail User Agent 180
make menuconfig 64
mangle ..... *siehe* Table, mangle
Markieren von Paketen .. 90
Masquerading . 60–61, 95
  Filterregeln 305
Matching
  Erweiterungen 74
Maximum Segment Size . *siehe* TCP
Maximum Segment Size (MSS) 43
Maximum Transfer Unit (MTU) 40
MDA *siehe* Mail Delivery Agent
Microsoft Management Console 573
Microsoft Management Konsole 563
`mmc` 563
`mod_clamav` 140, 144–148
`mod_vscan` 141
Mozilla 116
MS-CHAP 589
MSS ..... *siehe* Maximum Segment Size, 315
MTA *siehe* Mail Transport Agent
MTU .... *siehe* Maximum Transfer Unit, 314, 582
MUA .... *siehe* Mail User Agent
Muffin 177
MX-Record 189

N

Name Service Caching Daemon 109
Nameserver 189
  Caching only . 318, 340
  extern 194
  intern 194
  primärer 194
  sekundärer 194
NAT 60–61, 63, 92–97, 104
  Ablaufdiagramm ... 94
  Destination- . 93, 96–97, 123
  Filterregeln 364
  Kernel 2.6 und IPsec ... 498
  OUTPUT 94
  POSTROUTING 94
  PREROUTING 94
  Source- ... 60, 93, 95–96
  Filterregeln 345
NAT Traversal .. 549–550
NAT-T 549–550, 584
ncsa_auth 121
Nessus 479–485
Netfilter
  Erweiterungen . 74, 84, 97
Network Address Translation 90
Network Address Translation ..... *siehe* NAT, *siehe* NAT, 547–550
Network Associates . 174
Network Layer 36
Network Packet Filtering (Kernel) 65
Network Packet Filtering (Kernel) 64
Netzwerkanalyse 462–487
  Tools zur 464–487
News ..... *siehe* NNTP
NFS 240–242
ngrep 469–473
NIDS
  seeIntrusion Detection, netzwerkbasiert . 412
NIS 240–242
nmap 473–479
NNTP 234–238
Notfallpläne 28
NSCD 109
NTP 238–239

O

OE .. *siehe* Opportunistic Encryption
  Reverse Eintrag ... 528
Öffentliche Netzwerke ... 491
Öffentliche Infrastrukturen .. 491
OpenAntiVirus 141
  Projekt 139
  Scanner 139
openantivirus.org ... 110
OpenPGP 534, 535
OpenSSH .. 207, 611–623
OpenSSL .. 535–537, 539, 542
Openswan 497–499
Opportunistic Encryption 495, 508, 511, 525–529
OS-Fingerprinting ... 476
OSI-Referenzmodell 36–38

P

Pakete
  manipulieren ..... *siehe* Table, mangle
Paketfilter 53
  dynamisch 104, 211
  statisch 104
Paketfilterung . 48–51, 63, 69–92
  auf Kernelebene 63

dynamisch 50
Serverdienste
FTP 365
HTTP 363
statisch 49
Paketzähler 89
PAM 109, 253
PAP 589
Parent Cache 142, 145
Partitionierung 396
Passphrase 612
Passwörter
Einmal- 197
patch-o-matic 97–101
Perfect Forward Secrecy . 575
Personal Firewall (Windows) 559
Pforte 7
PGP 534
Physical Layer 36
ping 48
Ping 229
Ping-Sweep 475
PKCS12 ..... 540, 561
Policy Groups .. 508, 527, 529–533
mit X.509-Zertifikaten . 546–547
Policy Routing 525
POP3 183
Filterregeln 271
PortSentry 486–487
Postfix 180
PPP 557, 582
PPPD .. 571, 572, 581, 589
ip-down 523, 572
ip-up 520, 523, 572, 595
Optionen 582, 583
PPPo 582
PPPoE 315
Presentation Layer 37
Preshared Key 529
Pretty Good Privacy . 534
Private Host Key 612
Private Key 505
Private Networks 57
Private Netzwerkbereiche *siehe* Private Networks
Private Netzwerke ... 491
Private User Key 617
Protokollierung 9
Proxies 103–177
Proxy 51–53, 59
applikationsspezifisch . 53, 104, 105
Benutzerauthentifikation 104
Browserkonfiguration . 116
Cache 108
Caching 105
Filterung 106
generisch 53, 104
herkömmlich 59
HTTP 176, 360
httpf 177
Muffin 177
Squid 177, 360–362
httpd 107–125
implizit 52, 103
Parent Cache . 142, 145
Protokollierung ... 105
Reverse 104
transparent 61, 103, 109, 371
veränderte Clients .. 52
veränderte Verfahren .. 52
veränderter Client . 103
verändertes Verfahren . 103
Virenscan 136–149
Grundprinzip 137
Web *siehe* Proxy, HTTP
Public Host Key . 613, 617
Public Key . 201, 227, 495, 499, 505
Public User Key 617

Q

Qmail 180

R

r-Kommandos
Client-Ports 200
passwortlose Autorisierung ... 198
Server-Ports 199
`racoon` 497
RAS-Verbindung (Windows) . 574, 575
rc 618
rcp 198–201
rdist 199
rdump 198
Redirector 109, 110, 124–125
Regelketten
userdefiniert ... 82, 268
Verwendung 83
Rekursion .... *siehe* DNS, Rekursion
Relay-Mißbrauch 186
Replay-Attacke . 493, 496
Resolver ..... *siehe* DNS, Resolver
rexec 199
rlogin 198–201, 253
Road Warrior ... 504–505, 518–520
X.509-Zertifikate .. 545
Root-CA 534
Rootkits 16
Routing Decision 70
Routingtabelle 525
`rp-l2tpd` . 578, 582–583
rp_filter 267, 608
RPC-Scan 476
RPM
Checksummen 424
Paket-Update 404
rpm -V 424–426
rrestore 198
RSA 202
RSA Authentifikation 499, 505
rsh 198–201

S
SA ..... *siehe* Security Association
Scan-Detektoren 485–487
scanlogd 485–486
Schichtmodell ..... *siehe* OSI-Referenzmodell
Schlüsselaustausch 495–496
Schlüsselbund 500
Schlüsselmanagement ... 493
scp 203
Screened Subnet .... *siehe* Entmilitarisierte Zone
Secure Shell 201–207
SecureDNS 526
Security Association 493, 494, 496, 512
Security Policy 13–33
Security Scanner 479
Sendmail 180
  Sicherheitslücken .. 181
Session Layer 37
`setkey` 497
Shadow 253
Shared Secrets .. 495, 499
Shell 196
shosts.equiv 614
Sicherheit 13
  physikalisch 394
Sicherheitsaudits 30
Sicherheitsbedürfnis .. 13
Sicherheitsbestimmungen
  Durchsetzen von 8
Sicherheitskonzept ... 13, 26–33
Sicherheitspolitik . 10, 13, 21–26
Sicherheitsprotokolle 492–494
Sicherheitsrichtlinien (Windows) 557
Single Homed Host ... 56
SMTP 180–188
  Filterregeln 271
SMTP-Gateway 185
snort 414–422
sockd.conf 163
Sockets 43
SOCKS 160–174
  Client 336
  Client socksfähig machen 170
  Dante 162
  Server 163
  DNS 317
  Filterregeln 322
  Library 160
  NEC 161, 162
  Protokoll 160
  Server 338
  V4 161
  V5 161
socks.conf 171
socksify 170
Source Routing 42
Spam 181, 183, 186
SPAM 16, 136
SPAM-Filter 136
Sperrliste 125–127
Spoof Protection 267, 608
Spoofing 197
  DNS 190
Squid . 107–125, 177, 226, 360, 361
  Acceleration 123
  Acceleration Mode 109
  ACL 118–120
  Aktionsanweisung 112
  Authentifikation 120–122
  Basiskonfiguration 110–113
  Benutzerauthentifikation 109
  Browserkonfiguration . 116
  Cachegröße 114
  Default Policy 112
  Diskspace 113
  DNS-Caching 109
  Filterregeln 361
  Hauptspeicher 113–115
  ICAP-Client 139
  Ident 120
  ncsa_auth 121
  Objekt 111–115
  Redirector 109, 110, 124–125
  transparenter Proxy ... 122–124
  Virenscan 136–149
  Zugriffskontrolle .. 118
squid-vscan 110
`squid-vscan` 139
SquidGuard 125–136, 177
  Blacklist 126
  Datenbank 126, 136
  Diff-File 135
  Konfiguration 127–134
  Sperrliste 125–127
  Pflege 135–136
  Squid-Anpassung 134–135
Squirm 140
SSH ... 201–207, 611–623
  Agent 621
  Authentifikation
  DSA 620
  Hostbased 620
  Passwort 621
  Pubkey 620
  Rhosts 619
  RSA 620
  RSAHosts 620
  Buffer overflow ... 204
  Exploits 204
  Filterregeln 273
  Hostkey 612
  known_hosts 202
  Literaturhinweise . 623
  OpenSSH 207
  Passphrase 612
  Port 204
  Schlüsselerzeugung ... 615

Serverkonfiguration ... 612
ssh-agent 621
Start des Daemon . 613
systemweite Dateien .. 613
Userfiles 617
Verbindungen 204
ssh-keygen 612
ssh_config 614
ssh_host_dsa_key ... 612
ssh_host_dsa_key.pub ... 613
ssh_host_key 612
ssh_host_key.pub 613
ssh_host_rsa_key 612
ssh_known_hosts 613
sshd_config 613
sshrc 615
SSL 226–229
Filterregeln 272
Stateful Inspection ... 211
strongSwan 492, 497–499
strongSwan 498–550
dynamische IP 519–525
Implizite Verbindungen 529–533
Installation ... 499–503
Kernelpatch 501
Konfiguration 505–525
NAT mit 547–550
nexthop 517–518
Opportunistic Encryption . 525–529
Policy Groups 527, 529–533
Road Warrior . 518–520
STATE_MAIN 512
STATE_QUICK 512
Szenarien 503–505
Verbindungsaufbau ... 512
Wildcards 546
X.509-Zertifikate 533–547
SuSE Proxy Suite ... *siehe* ftp-proxy
SYN Flooding 15
syslog 435, 437–442
syslog.conf 438
syslogd 437

T

Table
filter 90
mangle 90
nat 90
Targets *siehe* Filterregeln, Targets
ACCEPT 81
Builtin 81
DNAT 96
DROP 81
Erweiterungen .. 81, 84
LOG 85
MARK 90
MASQUERADE 95
QUEUE 82
REJECT 86
RETURN 82
SNAT 95
TOS 91
userdefiniert 81, 82
TCP 42–46, 598–599
Flags 44
Maximum Segment Size ..... *siehe* TCP
MSS 315
Segmente 43
SYN-Cookies 608
Syncookies 64, 65
TCP-Scan
ACK 475
Connect 473
FIN 474
FTP-Proxy 474
Half-Open ... 467, *siehe* TCP-Scan, SYN
Ident
reverse 476
mit Fragmenten ... 474
NULL 474
SYN 467, 473
TCP Ping 475
Xmas 474
TCP/IP 36, 39–48
tcpdump 464–469
TCPMSS 315
Telnet 196–198
Port 196, 197
Session 197
Sicherheitsproblematik 197
Verbindungen 197
Terminal-Session 196
The Internet Security Association and Key Management Protocol ..... *siehe* ISAKMP
Three Way Handshake (TCP) 45
Time to Live (TTL) 42
TIS 174–176
TLS 226–229
TOS *siehe* Type of Service, 91
Transport Layer 36
Transportmode . 494–495
tripwire 426–434
Trojaner 19
TTL ... *siehe* Time to Live
Tunnelmode 494–495
TXT-Record 526, 528
Type of Service (TOS) . 41

U

UCE .... *siehe* Unsolicited Commercial E-Mail
UDP 46–47, 600
Datagramme 46
UDP-Scan 475
Uniform Resource Locator ... *siehe* URL
Unsolicited Commercial E-Mail 181
URL 124, 218
Userspace 82
UTF-8 149

V
Verfügbarkeit 10, 11
Vernetzung
  Manipulation 29
Verschlüsselungsalgorithmen 493
Vertrauenskette . 534–535
Vertraulichkeit .. 10, 492, 493
Viralator 139
Viren 19
Virenscan 136–149
  Apache::ProxyScan 140–144
  ICAP 139
  mod_clamav 140, 144–148
  mod_vscan 141
  squid-vscan 139
  Viralator 139
Virenscanner (Windows) 560
Virtual Private Network . *siehe* VPN
VPN 491–556
VPN passthrough ... 549
VPN-Gateway
  dynamische IP 520–524, 571–572
  DynDNS 520–524, 571–572
  für Windows-Clients .. 557–591
Vulnerability Scanner 479

W
Würmer 19
  E-Mail 20
  Internet-Wurm 19
  Netzwerk 20
WAREZ 213
Web Cache 108
Webbrowser 210
Webproxy 107–125
Webserver
  sicherer 364
Werbemails .. *siehe* SPAM
`wget` 140
Windows 557–591
  Import von X.509-Zertifikaten ... 561–568
  IPSEC.EXE ... 572–577
  IPsec/L2TP ... 584–591
  L2TP 584–591
  natives IPsec .. 572–577
  Safety first 559–561
`windowsupdates.com` . 568, 584
World Wide Web ... *siehe* WWW
WWW 217–226

X
X Window System 242–243
X.500 534
X.509 ..... 499, 534
X.509-Zertifikate 533–547
  Import nach Windows . 561–568
X11
  forwarding ... 202, 206
xntp
  Filterregeln 274
XNTP ..... *siehe* NTP
XNTPD ..... *siehe* NTP

Y
YP ..... *siehe* NIS

Z
Zertifikat 227
Zertifikate .. 496, 499, 534
  Gültigkeitsdauer .. 541
  Host- 535, 539–540
  User- 535, 539–540
  Widerruf . 534, 541–542
  Windows 540–541
Zertifkate
  PKCS12 540–541
Zugangsberechtigungen . 27
Zugangskennung 197
Zwangstrennung ... 522, 572, 593