Stefan Schumacher

Das NetBSD-Sicherheitshandbuch

Der Autor



ist selbständiger Unternehmensberater im Bereich IT-Sicherheit, seine Webseite finden Sie unter www.kaishakunin.com.

Geboren 1980, beschäftigt er sich seit 1992 mit Computern (Robotron KC85/3), seit 1994 mit PCs (486/SX25 mit MS-DOS 6.2), seit 1998 mit Unix (SuSE Linux 4.x) und seit 2001 mit NetBSD (1.5.1 auf einer DEC Alpha) im speziellen. Inzwischen benutzt er NetBSD auf PCs, HP Jornada, DEC Alpha und Vax, Sun Sparc und Sparc 64, Apple PowerMac, HP PA-Risc und Apollo. Als Datenbankadministrator und -Entwickler befasst er sich ausgiebig mit PostgreSQL und Perl. Beruflich betreibt er mehrere Web-, Samba- und Datenbankserver unter NetBSD. Er unterstützt das NetBSD-Projekt mit Informations- und Werbematerial sowie Dokumentationen.
Außerdem ist er Mitglied im Chaos Computer Club (CCC), der German Unix User Group (GUUG), sowie der Deutschsprachigen Anwendervereinigung TEX (Dante).

Er hält regelmäßig Fachvorträge zu den Themen NetBSD, Sicherheit, Kryptographie, Hacking und Unix im allgemeinen, z. B. auf den Chemnitzer Linux-Tagen, dem LinuxTag, dem Frühjahrsfachgespräch der GUUG oder dem Chaos Communication Congress und veröffentlicht Artikel in der UpTimes und der Datenschleuder. Zusammen mit Mario Heide ist er Herausgeber des zweiwöchentlich erscheinenden Podcasts für alternative Computersysteme (Pofacs.de).

In seiner Freizeit befasst er sich mit japanischen Kampfkünsten (Graduierungen im Shotokan-Karate, Jiu-Jitsu sowie Chi Ryu Aiki Jitsu) und militärischem Nahkampf, japanischer Kultur und Geschichte, Astronomie und Büchern von Stephen King.

Seine persönlichen Webseiten sind erreichbar unter www.net-tex.de und www.cryptomancer.de. Seine Unternehmensberatung unter www.kaishakunin.com

Für Anfragen zum Buch können Sie sicherheitshandbuch@net-tex.de verwenden. Der passende PGP-Schlüssel ist 0xB3FBAE33.

Stefan Schumacher

Das NetBSD-Sicherheitshandbuch

Ein Leitfaden zur Absicherung

eines NetBSD-Rechners

www.net-tex.de
www.cryptomancer.de
Magdeburg, den 22.11.2007



Made
in
Germany

© 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007 Stefan Schumacher
Alle Rechte verbleiben beim Urheber.
»NetBSD« and »pkgsrc« sind eingetragene Warenzeichen der NetBSD Foundation »Solaris« und »OpenSolaris« sind eingetragene Warenzeichen von SUN Microsystems. »HP-UX« ist ein eingetragenes Warenzeichen von Hewlett-Packard. »BSDi« ist ein eingetragenes Warenzeichen von Berkeley Software Design, Inc. »UNIX« ist ein eingetragenes Warenzeichen von The Open Group / X Open. »CERT« is a registered trademark and service mark of Carnegie Mellon University. »Microsoft Windows« is a registered trademark and service mark of Microsoft. »S/Key« ist ein eingetragenes Warenzeichen von Bell Communications Research, Inc. »Kensington« ist ein eingetragenes Warenzeichen von Kensington Computer Products Group. »registered trademark« ist ein eingetragenes Warenzeichen von Registered Trademarking. »Identigram« und »Holographic Shadow Picture« sind Warenzeichen der Bundesdruckerei. Alle Rechte verbleiben beim jeweiligen Inhaber, alle Warenzeichen bei ihren Besitzern. Sämtliche Urheberrechte werden in vollem Umfang anerkannt.

Dies ist Version 1.17 mit 301 Seiten, kompiliert am 22.11.2007 mit pdfLaTeX 3.141592-1.21a-2.2 (Web2C 7.5.4). Gesetzt aus der Palatino.

Erlernt man Kriegsstrategien, ohne sich auf den rechten Weg zu stützen, wird man schließlich schlaue Worte und oberflächliches Wissen für den Weg des Kriegers halten.

Issai Chozan, »Tengu Geijutsuron«
(Zen und Schwert in der Kunst des Kampfes)

INHALTSVERZEICHNIS

Quellcode- und Befehlsverzeichnis 13
Tabellenverzeichnis 15
Abbildungsverzeichnis 17

1 Vorwort 3
1.1 Über dieses Dokument 3
1.1.1 Dank 3
1.2 Warnung 4
1.3 Urheberrechtshinweis 4
1.4 Formatierungsregeln 4
1.5 Struktur des Dokuments 6

I THEORETISCHER HINTERGRUND ZUR SICHERHEIT 7
2 Was ist Sicherheit? 9
2.1 Was ist Sicherheit? 9
2.2 Warum werden Rechner angegriffen? 11
2.3 Wer greift wie an? 11
2.4 Angriffsvektoren 12
2.4.1 Rechteseigerung 12
2.4.2 Schadprogramme 12
3 Verteidigung 15
3.1 Strategie, Taktik, operative Mittel 15
3.1.1 Erste organisatorische Schritte 16
3.2 Verteidigung in der Tiefe 20
3.3 Risiko, Risikobewertung und Risikofolgeabschätzung 22
3.4 Die Sicherheitsrichtlinie 28
3.5 Inventur der vorhandenen Systeme 30
3.6 Behandlung eines Sicherheitsvorfalls 31
3.6.1 Digitale Forensik 31
4 Social Engineering 35
4.1 Psychologische Grundlagen der Manipulation 38
4.1.1 Reziprozität 39
4.1.2 Wer A sagt . . . – von Commitment und Konsistenz 41
4.1.3 Soziale Bewährtheit 47
4.1.4 Sympathie 49
4.1.5 Autorität 51
4.1.6 Knappheit 51
4.2 Mitarbeiter authentifizieren 51

4.2.1 Mitarbeiterausweise 52
4.2.2 Der deutsche Personalausweis 54
4.2.3 Entfernte Authentifizierung von Mitarbeitern 56
5 Grundlagen der Verschlüsselungstechnik 61
5.1 Warum sollte man Kryptographie einsetzen? 61
5.2 Prüfsummen 62
5.2.1 Das MD5-Prüfsummenverfahren 62
5.2.2 Verified Executables 63
5.3 symmetrische Verfahren 64
5.3.1 mcrypt(1) 64
5.4 asymmetrische Verfahren 64
5.4.1 Ein asymmetrisches Verfahren genauer betrachtet: RSA 65
5.5 kryptographische Signaturen 66
5.5.1 Schlüsselsignatur 67
II DAS BASISSYSTEM HÄRTEN 69
6 Das Basissystem härten 71
6.1 Hardware testen 71
6.2 Das System herunterfahren, neu starten oder in den Single-User-Modus bringen 72
6.3 Erste Schritte 74
6.3.1 Schlankes System 74
6.3.2 Das System aktuell halten 75
6.3.3 Niemals als Root arbeiten 75
6.3.4 Speicherauszüge vermeiden 77
6.3.5 Debugger Deaktivieren 77
6.3.6 Regular Expressions considered harmful™ 78
6.3.7 Monitor und Tastatur schützen 78
6.3.8 Das System herunterfahren 79
7 Zen, oder: Die Kunst ein Passwort zu finden 81
7.1 Gefahren für Passwörter 81
7.1.1 Klartext-Passwörter 81
7.1.2 Social-Engineering 81
7.1.3 Tastaturrekorder 83
7.1.4 Kompromittierende Strahlung 83
7.2 Was ist ein starkes Passwort? 84
7.2.1 Verwenden Sie kein Passwort das erraten werden kann! 85
7.2.2 Verwenden Sie kein Passwort das in einem Wörterbuch steht! 85
7.2.3 Verwenden Sie ein langes Passwort mit Groß- und Kleinschreibung, Zahlen und Sonderzeichen! 86
7.2.4 Das Passwort muss geheim bleiben! 87
7.2.5 Verwenden Sie nicht überall das selbe Passwort! 88
7.2.6 Passwortwechsel 88
7.3 Passwörter von Hand generieren 89

7.4 Passwörter automatisch generieren 90
7.5 Passwort-Hashes generieren 92
7.6 Passwörter in NetBSD 93
7.7 Passwörter cracken 96
7.7.1 Mit Crack 96
7.7.2 Mit John the Ripper 96
7.8 Einwegpasswörter mit S/Key 97
7.9 OpenSSH 98
7.9.1 SSH-Logins verbieten 99
7.9.2 Verschlüsselte Tunnel 99
8 Systemaufrufe mit Systrace reglementieren 101
8.1 Systemaufrufe mit Systrace steuern 101
8.1.1 Die Grammatik der Richtlinie 103
8.1.2 Erzeugung einer Richtlinie 105
8.2 Implementierung 105
8.3 Sicherheit des Systems 106
8.4 Apache überwachen 108
8.5 Systrace-überwachte Shell 113
8.6 Weiterer Nutzen 116
8.7 Fazit 116
9 Daten sicher löschen 117
9.1 Wie werden Daten gespeichert und gelöscht? 117
9.1.1 Zerstörungsfreie Datenvernichtung 121
9.2 Programme 124
9.2.1 Die NetBSD/Schrubber Live-CD 129
9.2.2 Physikalische Datenvernichtung 130
9.3 Vorbeugende Maßnahmen 134
9.4 Datensicherung und Datenvernichtung 135
9.5 Fazit 135
10 Schadsoftware bekämpfen 137
10.1 Dateisystemintegrität mit Mtree prüfen 137
10.1.1 Konfiguration und Einsatz 137
10.2 Dateisystemintegrität mit Aide prüfen 140
10.3 Dateisystemintegrität mit tripwire(1) prüfen 143
10.4 Prüfsummen einzelner Dateien erstellen 143
10.5 Den Rechner mit Chkrootkit auf Rootkits prüfen 144
10.6 Das Netzwerk auf Zombies untersuchen 145
11 Netzwerke 147
11.0.1 TCP-Wrapper 147
11.1 Netzwerkverkehr mit OpenSSH verschlüsseln 147
11.1.1 Konfiguration des Dæmons 147
11.1.2 Konfiguration des Clients 152
11.1.3 OpenSSH benutzen 156
11.1.4 OpenSSH härten 157

11.2 Verschlüsselte IP-Tunnel mit Stunnel 160
11.3 Portscans mit Portsentry verhindern 160
11.4 Rechnerpräsenz in Netzwerken mit Arpwatch überwachen 161
11.5 Netzwerkverkehr mit Tcpdump überwachen und analysieren 163
11.5.1 Ausgabe 163
11.5.2 Datagrammtypen 164
11.5.3 Filter 166
11.5.4 Tcpdump-Daten mit Tcptrace aufbereiten 169
11.6 Emails über SSL getunnelt mit Fetchmail abholen 171
12 Angewandte Kryptographie 173
12.1 Das kryptographische Pseudogerät CGD 173
12.1.1 Was ist CGD? 173
12.1.2 Installation und Einrichtung 173
12.1.3 Eine Datenpartition verschlüsseln 174
12.1.4 CGD für Swap und /tmp 175
12.1.5 CGD für die /tmp-Partition 177
12.1.6 CGD für Container und/oder CDs/DVDs 178
12.2 Das kryptographische Dateisystem `cfs` 179
12.2.1 Was ist CFS? 179
12.2.2 Installation und Konfiguration 180
12.2.3 CFS auf Wechselmedien 181
12.2.4 CFS und Datensicherung 182
12.2.5 CGD oder CFS? 182
12.3 Symmetrische Verschlüsselung mit mcrypt(1) 183
12.4 Verschlüsselung mit OpenSSL 185

iii ANWENDUNGEN 189
13 Anwendungen 191
13.1 Installierte Pakete auf Sicherheitslücken überprüfen 191

iv DATENSICHERUNG 195
14 Strategien zur Datensicherung 197
14.1 Definitionen 197
14.2 Vorbereitung der Strategie 197
14.3 Inventur 198
14.4 Bedrohungsszenarien analysieren 198
14.5 Wichtige Daten finden und organisieren 200
14.6 Sicherung der Sicherungssysteme 201
14.7 Organisation der Sicherung 201
14.7.1 Einweisung der Benutzer 202
14.7.2 Cronjob vs. Batchjob 202
14.7.3 Homogene Systeme 203
14.7.4 Dokumentation der Adminstration 203
14.7.5 Shellskripte statt Handarbeit 203
14.7.6 Logdateien erstellen und auswerten 204

14.7.7 Verifikation der Daten 204
14.7.8 Den GAU erwarten 204
14.7.9 Alles sichern 205
14.7.10 Testsysteme und -läufe 205
14.7.11 Verschiedene Sicherungskreise 206
14.7.12 Maximale Archivierungsdauer 206
14.7.13 Wochenende und Feiertage beachten 206
14.8 Dokumentation 207
14.9 Testen, Testen, Testen 207
14.10 Verminderung von Ausfällen und Ausfallzeiten 208
14.11 Datenrettung 209
15 Sicherungsvarianten 211
15.1 Komplettbackup 211
15.2 Differentielles Backup 211
15.3 Inkrementelles Backup 211
15.4 Dump-Level 211
15.5 Beispielstrategie 212
15.6 Komprimierung und Verschlüsselung 214
15.7 Medien 216
15.7.1 Organisation und Lagerung der Medien 217
15.8 Prüfliste zur Strategie 218
16 Programme zur Datensicherung 221
16.1 dump(8), dump_lfs(8) und restore(8) 221
16.1.1 restore 222
16.1.2 Dump im Detail 224
16.2 Tar, Star, Cpio, Afio und Pax 226
16.3 dd(1) 228
16.4 Ghost for Unix (g4u) 228
16.5 Amanda 229
16.5.1 Einrichtung des Servers 230
16.5.2 Einrichtung der Clients 233
16.5.3 Programme im Amanda-Paket 233
16.5.4 Praktischer Einsatz 234

V EINBRUCHSERKENNUNG IN NETZWERKE 237
17 Einbruchserkennung in Netzwerke 239
17.1 Intrusion Detection Systeme 239
17.1.1 Wozu Einbruchserkennung? 239
17.1.2 Was ist/kann IDS nicht? 239
17.2 Prinzipien der Einbruchserkennung 240
17.2.1 Datensammlung 240
17.2.2 Angriffserkennung 241
17.3 Angriffe gegen IDS 243
17.3.1 Integritätsprüfung 243

17.3.2 Signaturerkennung 243
17.3.3 Programme 245
17.4 Multilayer IDS (mIDS) 246
17.5 Honeypot 247
17.5.1 Analyse 248
17.5.2 Implementierungsprobleme bzw. Angriffsmöglichkeiten 249
17.5.3 Programme 250
17.6 Fazit 251

VI DEN FEIND KENNEN 253
18 Eigene Systeme aufklären und Angreifen 255
18.1 Schwachstellen-Scanner 255
18.1.1 Nessus 255
19 Feindaufklärung: Informationen über Sicherheitslücken und Einbrecher 259
19.1 NetBSD und Sicherheit 259

VII ANHÄNGE UND VERZEICHNISSE 261
A ICMP-Pakettypen und Codes 265
B US-Tastaturlayout 267
C The Hacker Manifesto 269
D Weitere Informationsquellen 273
E Verwendete Pakete 275
F Literaturverzeichnis 279

QUELLCODE- UND BEFEHLSVERZEICHNIS

1.1 Dies ist ein Formatierungsbeispiel . . . 5
1.2 eine Manpage anzeigen lassen . . . 6
3.1 Daten durch ein Netzwerk übertragen . . . 33
3.2 Systeminformationen sichern . . . 34
5.1 Beispiel einer PGP-Signatur . . . 67
6.1 lasterzeuger.sh, ein Skript um CPU und RAM zu testen . . . 72
6.2 Verschiedene Shutdown-Beispiele . . . 74
6.3 /etc/ttys mit »sicheren« Konsolen . . . 76
6.4 /etc/ttys mit »unsicheren« Konsolen . . . 77
6.5 Speicherauszüge verhindern . . . 78
6.6 Das Bein mit einem Leerzeichen wegschießen . . . 78
10.1 von mtree erzeugte Eigenschaftenliste . . . 138
10.2 von mtree erzeugte Eigenschaftenliste (2) . . . 139
10.3 Prüflauf mit mtree . . . 140
10.4 Einsatz von Aide . . . 142
10.5 Optionen in aide.conf . . . 142
10.6 Beispiel aus einer Aide-Datenbank . . . 143
10.7 Dateien mit MD5 und SHA1 vergleichen . . . 144
10.8 Beispiel eines Ddos-Scan-Laufs . . . 146
11.1 SSH mit Schlüsselauthentifikation . . . 156
11.2 PostgreSQL-Ports mit SSH tunneln . . . 157
11.3 Entfernte Befehle mit SSH ausführen . . . 157
11.4 Auszug aus einem Arpwatch-Protokoll . . . 162
11.5 Tcpdump-Beispiel . . . 164
11.6 Tcpdump-Beispiel einer ARP-Anfrage . . . 165
11.7 Tcpdump-Beispiel von AFS-Verkehr . . . 165
11.8 Tcpdump-Beispiel von ICMP-Paketen . . . 166
11.9 Tcpdump-Filterbeispiele . . . 169
11.10 Beispiel einer .fetchmailrc . . . 172
12.1 CGD im Kernel aktivieren . . . 174
12.2 CGD-Konfiguration erstellen . . . 175
12.3 CGD für die Swappartition konfigurieren . . . 176
12.4 Disklabel für die Swappartition sichern . . . 176
12.5 Verschlüsselte Swappartition beim Start einbinden . . . 177
12.6 CGD für /tmp konfigurieren . . . 177
12.7 verschlüsselte /tmp-Partition einbinden . . . 178
12.8 CGD-Container . . . 178
12.9 Verschlüsseltes und entschlüsseltes CFS-Verzeichnis . . . 179
12.10 CFS-Verzeichnisse einrichten und Dienste starten . . . 180

12.11 verfügbare CFS-Algorithmen . . . 181
12.12 CFS Verzeichnis anlegen, einbinden und deaktivieren . . . 181
12.13 Eine Datei mit mcrypt(1) ver- und entschlüsseln . . . 184
12.14 OpenSSL-Version anzeigen . . . 185
12.15 Prüfsummen mit OpenSSL erstellen . . . 185
12.16 Dateien mit OpenSSL verschlüsseln . . . 186
13.1 Aufbau der Paketdatenbank in /var/db/pkg . . . 192
13.2 Installierte Pakete auf Sicherheitslücken prüfen . . . 192
14.1 Daten verifizieren . . . 204
15.1 Archive komprimieren und verschlüsseln . . . 215
16.1 NODUMP setzen . . . 222
16.2 Sicherung mit dump . . . 222
16.3 dump-Befehl für mehrere Volumes . . . 223
16.4 Dump im Netzwerk . . . 223
16.5 Zwei Dumps mit restore zurückspielen . . . 223
16.6 einzelne Dateien mit restore zurückspielen . . . 224
16.7 Protokoll einer Dump-Sicherung . . . 225
16.8 Dateien mit tar archivieren und entpacken . . . 226
16.9 Dateien mit cpio sichern . . . 227
16.10 Sicherung und Rücksicherung mit pax . . . 227
16.11 Daten mit dd schreiben . . . 228
16.12 Festplatten-Images mit g4u erstellen . . . 229
16.13 Spoolplatten für Amanda definieren . . . 232
16.14 dumptypes in amanda.conf definieren . . . 232
16.15 Zu sichernden Pfade in der disklist festlegen . . . 233
16.16 Amanda-Server-Dienste starten . . . 233
16.17 Amanda-Client-Dienste starten . . . 233
16.18 Zugriffsrechte auf dem Client konfigurieren . . . 234
16.19 Amanda Header vom Band restaurieren . . . 235
17.1 Konfigurationsbeispiel für honeyd . . . 251
18.1 Einen Nessus-Benutzer anlegen . . . 256

Tabelle 3.1 10 Schritte auf dem Weg zu mehr Sicherheit 19
Tabelle 3.2 Prinzipien der Verteidigung in der Tiefe 21
Tabelle 3.3 Risikotypen im Überblick 26
Tabelle 3.4 Kategorisierung der Risiken 26
Tabelle 3.5 Risiko-Gegenmaßnahmen 27

Tabelle 6.1 Tastenkombinationen um ddb(4) aufzurufen 79

Tabelle 10.1 eingebaute Prüfsummen 144

Tabelle 11.1 ein Tcpdump-Beispiel näher betrachtet 164
Tabelle 11.4 Pakettypen für Tcpdump spezifizieren 168
Tabelle 11.6 Tcptrace-Optionen 169

Tabelle 12.1 in OpenSSL verfügbare Prüfsummenverfahren 186
Tabelle 12.2 in OpenSSL verfügbare Kryptoalgorithmen 187

Tabelle 15.1 Wochensicherung mit Leveln 212
Tabelle 15.2 Datenmenge einer Komplettsicherung 212
Tabelle 15.3 Datenmenge einer Inkrementellsicherung 213
Tabelle 15.4 Datenmenge einer Differentiellsicherung 213
Tabelle 15.5 Türme-von-Hanoi-Algorithmus 214
Tabelle 15.6 Datenverteilung im Hanoi-Algorithmus 214
Tabelle 15.7 Vergleich von Sicherungsmedien 217
Tabelle 15.8 Prüfliste zur Strategie 219

Abbildung 3.1 Aus Filtergründen segmentiertes Netzwerk 22

Abbildung 4.1 Muster eines deutschen Personalausweises 55
Abbildung 4.2 Erzeugung von ITANs 58

Abbildung 7.1 Beispiel für einen Benutzereintrag aus NetBSDs `/etc/master.passwd` 86
Abbildung 7.2 aussprechbare Passwörter mit apg erzeugen 91
Abbildung 7.3 nicht-aussprechbare Passwörter mit Apg erzeugen 91
Abbildung 7.4 Passwort-Hash mit OpenSSL oder pwhash(1) erzeugen 93
Abbildung 7.5 Aufbau der /etc/master.passwd und /etc/passwd 94
Abbildung 7.6 Zeitbedarf zur Generierung von Blowfish-Hashes 95
Abbildung 7.7 Konfigurationsmöglichkeiten in der /etc/passwd.conf 95
Abbildung 7.8 Passwörter cracken mit John 97
Abbildung 7.9 S/Key-Schlüssel erstellen 98
Abbildung 7.10 SSH mit Schlüsselauthentifikation 99
Abbildung 7.11 PostgreSQL-Ports mit SSH tunneln 100

Abbildung 8.1 Xsystrace bittet den Benutzer um eine Entscheidung 108
Abbildung 8.2 Automatisch eine Richtline für Apache erstellen 109
Abbildung 8.3 Automatisch erzeugte Richtlinie für Apache (Auszug) 110
Abbildung 8.4 Automatisch erzeugte Richtlinie korrigiert für Apache (Auszug) 111
Abbildung 8.5 Automatisch generierte Richtlinie für Apache, die jede Datei einzeln aufführt. 111
Abbildung 8.6 Angepasste Apache-Richtline 111
Abbildung 8.7 Angepasste Apache-Richtline 112
Abbildung 8.8 Kapselung des Shell-Aufrufes 114
Abbildung 8.9 Kapselung des Shell-Aufrufes 115

Abbildung 9.1 Eine Partition mit dd(1) überschreiben 128
Abbildung 9.2 Eine Partition 33-mal mit wipe(1) und dd(1) überschreiben 128
Abbildung 9.3 Degaußer 2 von Ibas (Quelle: www.ibas.de) 131

Abbildung 11.1 mit Gnuplot geplotteter Tcpdump-Zeitsequenzgraphen 170

Abbildung 18.1 Nessus starten und einloggen 257

Abbildung 18.2 Nessus-Plugins und Prüfoptionen konfigurieren 257
Abbildung 18.3 Nessus-Prüfbericht 258

1 VORWORT

ouch erkante ich nie sô wisen man,
en möhte gerne künde hân,
welher stiure disiu mære gernt
un waz si guoter lêre wernt.

(Wolfram von Eschenbach, »Parzival«)

1.1 ÜBER DIESES DOKUMENT

Das Handbuch entstand als Sammlung verschiedener Artikel, die ich im Laufe der Zeit über NetBSD oder verwandte Themen geschrieben habe. Nachdem es etwas zu groß für eine einfache Sammlung wurde, habe ich passende Zwischenkapitel und etwas mehr zur Theorie hinzugefügt, so das ein halbwegs vollständiges Handbuch entstanden ist. Bei dem gewählten Thema können nicht alle Programme und Gesichtspunkte berücksichtigt werden.

Das Literaturverzeichnis wurde mit BibTEX verwaltet und mit dem dinat-Stil formatiert.

Die Versionsverwaltung erfolgte mit CVS, CVS-Informationen sind in der Titelei angegeben.

1.1.1 *Dank*

Wesentlicher Dank für ungezählte inhaltliche Hilfe und Korrekturen gebührt Dr. Hubert Feyrer[1].

[1] http://www.feyrer.de

1.2 WARNUNG

Ich habe dieses Buch sorgfältig recherchiert und weiß im Allgemeinen wovon ich rede. Trotzdem gewähre ich diese Informationen nur auf eigene Gefahr, da das Kapitel der Computersicherheit sehr komplex ist und ich weder alle Fälle und Besonderheiten kennen noch berücksichtigen kann.
Wenn ein professionelles Sicherungsverfahren notwendig ist, Sie aber nicht wissen wie es fehlerfrei umgesetzt werden kann, besorgen Sie sich professionelle Hilfe. Das ist in der Regel billiger als ein Sicherheitsvorfall.

Fragen werde ich in der Regel beantworten, es bietet sich aber auch an die NetBSD-Mailinglisten zu benutzen.

Ich gebe in diesem Buch einige Beispiele an, wie generierte Passwörter oder Konfigurationsoptionen. Übernehmen Sie diese auf gar keinen Fall Eins zu Eins in ihr System! kann und wird auch ein Einbrecher dieses Buch lesen und ausnutzen, denn wie alle Instrumente ist es nicht wertfrei oder per se gut, sondern erbt die Werte von der Gesellschaft, in der es eingebettet ist[2].

Benutzen Sie weitere Literatur oder Informationsmöglichkeiten und nicht nur dieses Werk allein.

1.3 URHEBERRECHTSHINWEIS

Sämtliche Rechte am Werk verbleiben beim Urheber. Weitergabe des unveränderten Artikels ist gestattet und erwünscht, Wiedergabe in kommerziellen Medien bedarf der Genehmigung des Autors. Zitate gemäß den üblichen Regeln sind erwünscht.

1.4 FORMATIERUNGSREGELN

Befehle, die in der Konsole eingegeben werden müssen, werden als Fließabbildung gesetzt, die im Text über die Nummer »(siehe Listing 1.1)« referenziert werden. Die Befehle werden dabei in dicktengleicher Schrift gesetzt. Handelt es sich um Kommandozeilenbefehle, werden die Zeilen mit dem Prompt eines normalen Benutzers, dem Dollar (»$«) oder dem des Systemadministrators root, dem Doppelkreuz (»#«) eingeleitet. Da es bei einigen Ausgaben zu überlangen Zeilen kommt, die in dieser Druckversion nicht gesetzt werden können, verwende ich Auslassungszeichen (»[...]«)

[2] Frei nach Joseph Weizenbaum »*Computer sind wie alle Instrumente nicht wertfrei, sondern erben ihre Werte von der Gesellschaft, in der sie eingebettet sind.*«

um Auslassungen zu kennzeichnen oder Zeilenumbrüche, die mit einem Backslash (»\«) gekenzeichnet werden. Die umgebrochene Zeile wird dann eingerückt. Zeichen die normalerweise nicht angezeigt werden, bspw. Passwort-Eingaben, werden mit Asterisken »*« dargestellt.

In der selben Umgebung werden auch Auszüge aus Konfigurationsdateien gesetzt, dies wird dann in der Umgebungsunterschrift angegeben. Alle derartige Umgebungen werden im Befehlsverzeichnis (siehe Seite 13) verzeichnet.

[!h]

```
$ uname -sr | figlet -f small
 _  _     _   ___ ___ ___    _ _   __   ___ ___ _____ _
| \| |___| |_| _ ) __|   \  | | | /  \ | _ ) __|_   _/_\
| .` / -_)  _| _ \__ \ |) | |_  _| () || _ \ _|  | |/ _ \
|_|\_\___|\__|___/___/___/    |_(_)__/_|___/___| |_/_/ \_\
                                    |___|
$ whoami
stefan
$ su
Password:
# whoami
root
# exit
$
```

Listing 1.1: Dies ist ein Formatierungsbeispiel

Am Ende eines Kapitels befindet sich der Abschnitt »Siehe auch:«, dort werden weitere Informationsquellen zum behandelten Thema genannt. Dies sind zum Einen die Verweise auf entsprechende Quellen die im Literaturverzeichnis stehen, zum Anderen aber auch insbesondere Manpages zu betroffenen Programmen. Dies erfolgt dann in der Form: »`man 1 intro`«, wobei die Zahl vor dem Befehl die Sektion der Manpage anzeigt. Um die Manpage zu lesen, geben Sie in der Konsole »man Befehl« ein, wie in Listing 1.2 gezeigt. Alternativ sind die Manpages in der PDF-Datei mit der NetBSD-Webseite für Manpages verlinkt, so das mit einem Klick auf »`man 1 intro`« im Webbrowser die passende Manpage angezeigt wird. Dies funktioniert nur wenn Ihr PDF-Betrachter diese Option unterstützt.

Im Fließtext habe ich die Programm- / Befehlsnamen der deutschen Rechtschreibung weitestgehend angepasst, so wird aus `audit-packages` in einem gewöhnlichen Satz »Audit-Packages«. Um den genauen Befehl zu bezeichnen verwende ich die altbekannte Notation mit der Sektion, also »audit-packages(8)«, hierbei bedeutet »(8)«, das sich die Man-Page zum Programm in der achten Sektion befindet, Sie sie also mit »`man 8 audit-packages`« anzeigen lassen können. Bei der Aufzählung von Optionen oder kurzen Befehlen im Text verwende ich eine dicktengleiche Schriftart:

Nach der Installation des Paketes Audit-Packages stehen die beiden Be-

[!h]

```
$ man man
MAN(1)NetBSD General Commands ManualMAN(1)

NAME
man -- display the on-line manual pages (aka 'man pages')

SYNOPSIS
man [-acw|-h] [-C file] [-M path] [-m path] [-S srch] [[-s] section] name
         ...
man [-k] [-C file] [-M path] [-m path] keyword ...

DESCRIPTION
The man utility displays the manual pages named on the command line.  Its
options are as follows:
[...]
```

Listing 1.2: eine Manpage anzeigen lassen

fehle download-vulnerability-list(8) und audit-packages(8) zur Verfügung. Mit der Option `-d` lädt audit-packages(8) die Verwundbarkeitsliste selbständig herunter: `audit-packages -d`.

Da in diesem Handbuch naturgemäß viele Pakete aus Package-Source besprochen werden, habe ich ein eigenes Verzeichnis der Pakete erstellt, es ist unter Kap. E, »Verwendete Pakete« auf Seite 275 ff. angehängt und enthält die Kurzbeschreibung der Pakete, deren Verzeichnis in Pkgsrc und Verweise auf die passende Webseite. Im Text referenziere ich Pakete in der Form »[PP25]«, wobei PP für Pkgsrc-Paket steht und 25 die entsprechende laufende Nummer im Verzeichnis ist. In der PDF-Datei verbirgt sich hinter dem Text ein Verweis auf das Verzeichnis.

1.5 Struktur des Dokuments

Im ersten Teil werden theoretische Grundlagen der Sicherheit erläutert. Es wird

Im siebten Teil werden Informationsquellen über Angreifer und Sicherheitslücken vorgestellt, die besonders nützlich sind um weitere Informationen über Entwicklungen in der Szene zu bekommen.

Im achten und letzten Teil befinden sich die Anhänge, wie Skripte, Index und Bibliographie.

Teil 1

THEORETISCHER HINTERGRUND ZUR SICHERHEIT

2 WAS IST SICHERHEIT?

Wir müssen wissen – wir werden wissen.

(David Hilbert)

2.1 WAS IST SICHERHEIT?

DIN61508 definiert »Sicherheit« als Freiheit von unvertretbaren Risiken. Übertragen auf ein Betriebssystem bedeutet dies, daß es sich zu *jedem Zeitpunkt* in einem *wohldefinierten Zustand* befindet. Dieser Zustand wird vom Systemverwalter oder Systembenutzer vorgegeben und nicht durch einen unrechtmäßigen Zugriff beeinflusst.

Sicherheit lässt sich in gewissen Maßen messen und berechnen, und zwar als

$$\mathtt{Sicherheit} := 1 - \mathtt{Risiko}$$

wobei das Risiko die Wahrscheinlichkeit des Eintretens eines negativen Ereignisses beschreibt. Im wirtschaftlichen Zusammenhang kann man einem Risiko eine finazielle Bewertung beimessen und in diesem Zusammenhang die Schadenshöhe eines Einzelrisikos als Produkt der Schadenshöhe mal Eintrittswahrscheinlichkeit definieren. Betrachtet man *Sicherheit* und *Risiko* eines Systems aus Sicht eines Ingenieurs, ist hierbei immer von einer Menge von Risiken auszugehen, welche additiv verknüpft werden können. Außerdem spielt in technischen Systemen die Zeit eine wichtige Rolle, da insbesondere die Laufzeit eines Systemes das Risiko erhöht. Dies gilt in besonderem Maße für Rechnersysteme, da alte Software in der Regel mehr bekannte Sicherheitslücken aufweist. Mann kann die Definition um den positiven Laufzeitfaktor erweitern und die Einzelrisiken zum Gesamtrisiko addieren:

$$\mathtt{Sicherheit} := 1 - (\mathtt{Gesamtrisiko} + \mathtt{Laufzeitfaktor})$$

Weiterhin hängen Einzelrisiken eines Teilsystem voneinander ab, Sicherheit lässt sich nur für ein komplettes Szenario betrachten. Man kann ein Teilsystem nicht als »sicher« oder »unsicher« klassifizieren, sondern es nur im Gesamtszenario betrachten.

Ein Beispiel für diese These ist die Emailverschlüsselung: es ist sinnvoll Emails zu signieren und zu verschlüsseln, da diese so auf dem Übertragungswege nicht mehr manipuliert werden können. Man kann aber nicht per se davon ausgehen das ein System mit Emailverschlüsselung sicher ist, wenn nicht ebenfalls Maßnahmen getroffen wurden um Schadsoftware, Spionageprogramme, Tastaturrekorder und ähnliches fernzuhalten.

Dem *Risiko* auf Seiten des Systemverwalters steht der *Aufwand* des Angreifers entgegen. Dies umfasst alle Ressourcen die dieser aufbringen muss um in das System einzudringen. Gelingt es dem Verteidiger eines Systems, den Aufwand für einen Einbruch soweit zu erhöhen, das er den Nutzen des Einbruchs weit übersteigt, sinkt die Angriffswahrscheinlichkeit. Daraus folgt im Umkehrschluss das wertvolle Systeme lohnendere Angriffsziele stellen und damit aufwändigere Angriffe provozieren.

Ebenso wie ein Angreifer benötigt auch ein Verteidiger Strategien, Taktiken und operative Mittel um sein Ziel zu erreichen. Daher ist einer der ersten und wichtigsten Schritte, die zu erreichenden Ziele zu identifizieren.

Diese sind in der Computersicherheit:

- Datenschutz

  PRIVATSPHÄRE Private Daten sind zu schützen.

  VERTRAULICHKEIT Geheime Daten sind zu schützen.

  ÜBERTRAGUNGSSICHERHEIT Übertragene Daten dürfen nicht manipuliert werden.

- Datensicherheit

  INTEGRITÄT DES SYSTEMS Das System muss sich allzeit bereit im gewünschten, integeren Zustand befinden.

  FUNKTIONSBEREITSCHAFT Alle benötigten Funktionen des Systems müssen zur Nutzung bereitstehen.

  VERBINDLICHKEIT IM SINNE DER URHEBERSCHAFT Der Urheber eines Datums darf die Urheberschaft nicht abstreiten können.

  AUTHENTIZITÄT DER DATEN Daten dürfen nicht geändert worden sein.

- ebenso:

ZUGRIFFSKONTROLLE Nur authentifizierte und autorisierte Personen dürfen Zugriff auf Systeme und Daten haben.

Für jedes dieser Ziele lassen sich geeignete Taktiken und operative Mittel identifizieren. Vertraulichkeit und Übertragungssicherheit lässt sich beispielsweise mit kryptographischen Methoden sicherstellen. Damit diese aber nicht kompromittiert werden können, muss das verwendete System integer sein. Somit wird deutlich, das hier mehrere Ziele einander bedingen und somit eine Gesamtbetrachtung des Szenarios erfordern.

Sicherheit ist ein komplexer Prozess.

Es ist sinnlos an einem nicht-vertrauenswürdigen Rechner in einem Internet-Cafe eine verschlüsselte Email abzuschicken, da nicht sichergestellt werden kann, das die Email nicht von einem Trojaner oder Tastaturrekorder mitgeschnitten wird.

2.2 WARUM WERDEN RECHNER ANGEGRIFFEN?

Die Gründe für Angriffe auf Rechnersysteme können stark variieren. Von Script-Kiddies, die einfach nur randalieren wollen, in dem sie ihre neuesten Skripte abfeuern bis hin zu professionellen Einbrechern, die im Auftrag einer feindlichen Organisation z. B. Industriespionage oder Sabotage durchführen.

Für den Verteidiger eines Systems ist es lebenswichtig zu wissen wie Angreifer arbeiten und Angriffe funktionieren. Es gibt keinen besseren Verteidiger als einen versierten Hacker.

Heutzutage werden die meisten Systeme über unsichere Netzwerkdienste bzw. -Protokolle kompromittiert. Dabei sind die angegriffenen Dienste entweder von Haus aus unsicher oder wurden vom Administrator falsch konfiguriert.

NetBSD verfügt standardmäßig über eine große Anzahl von Sicherheitsfunktionen und zusätzlichen Programmen, die die Sicherheit und Stabilität eines Systems erhöhen.

2.3 WER GREIFT WIE AN?

Für gewöhnlich hat ein Angreifer irgendein Motiv um in ein System einzudringen. Die Motive variieren stark und reichen von Vandalismus aus Spass hin zu Erpressungsversuchen oder von Geheimdiensten unterstützen Industriespionage.

Ebenso weit gefächert wie die Motive ist die Herkunft und der Hintergrund der

Angreifer.

2.4 ANGRIFFSVEKTOREN

Es gibt verschiedene Wege ein System anzugreifen. Unter Angriff verstehe ich alle Maßnahmen, die von einem Angreifer ausgeführt werden um Root-Rechte und damit die Kontrolle über das System zu erlangen.

2.4.1 *Rechtesteigerung*

Ein Benutzer kann unter gewissen Umständen Programme ausführen, die mehr Rechte haben als er selbst. Dies sind meist sog. SETUID-Programme, die mit Root-Rechten laufen. Ein Angreifer kann derartige Programme manipulieren um eigenen Prozessen ebenfalls Root-Rechte zuzugestehen.

Programme mit SETUID sind potenziell sehr gefährlich und sollten nach Möglichkeit vermieden und streng überwacht werden. Programme die SETUID-gesetzt sind, können mit `find / -perm -4000` gefunden werden.

2.4.2 *Schadprogramme*

Allgemein betrachtet ist ein Schadprogramm ein Programm das das Ziel hat anderen Programmen, Systemen oder Nutzern zu schaden.

Es gibt verschiedene Klassen von Schadprogrammen, die sich jeweils mit geeigneten Maßnahmen bekämpfen lassen.

2.4.2.1 *Viren*

Ein Virus ist ein Programm das sich selbst reproduzieren und damit andere Rechner infizieren kann. Viren konzentrieren sich zwar hauptsächlich auf kommerzielle Betriebssysteme, die traditionell extreme Sicherheitsprobleme haben. Trotzdem kann jedes Betriebssystem angegriffen werden.

Normalerweise verwendet man Virenscanner um Viren anhand ihrer Signatur aufzuspüren. Dazu muß der Virus aber schon bekannt, analysiert und im Lexikon des

Scanners hinzugefügt worden sein.

Da ein Virus auch nur ein gewöhnliches Programm ist, kann es mit Maßnahmen wie Verified Executables (siehe Kap. **??**, »**??**« auf Seite **??** ff.) bekämpft werden.

Früher verbreiteten sich Viren oft über Wirtsprogramme oder den Master-Boot-Sektor einer Diskette. Heute eher über manipulierte Webseiten oder Email, was wiederum von kaputten Emailprogrammen begünstigt wird.

2.4.2.2 *Würmer*

Ein Wurm gleicht in gewissen Grenzen einem Virus. Er repliziert sich aber nicht über ein Programm, sondern über das Netzwerk. Für gewöhnlich benutzen Würmer verwundbare Netzwerkdienste wie BIND oder Sendmail um sich zu verbreiten. Sind sie auf einem Rechner seßhaft geworden, suchen sie die nächsten Opfer im Netz.

Würmer tragen meist zerstörerische oder gefährliche Programme als Nutzlast mit sich. Aber selbst wenn dies nicht der Fall ist, können Würmer allein durch ihre Verbreitung das Netzwerk verstopfen.

Gegen Würmer helfen die selben Maßnahmen wie gegen Viren, außerdem müssen Netzwerkdienste sicher konfiguriert und mit einem Paketfilter geschützt werden.

2.4.2.3 *Rootkits, Hintertürchen und Trojaner*

Ein Rootkit ist ein Programm, welches für gewöhnlich nach einem erfolgreichen Einbruch installiert wird, um den Einbruch zu verschleiern und dem Einbrecher das weitere Vorgehen zu erleichtern. Es tauscht bspw. Systembefehle wie ps(1) oder ls(1) gegen manipulierte Versionen aus, die die Dateien und Prozeße des Einbrechers verstecken. Zusätzlich installiert ein Rootkit meist eine Hintertür (engl. Backdoor), über die sich der Einbrecher jederzeit Zugang zum System verschaffen kann.

Es gibt aber auch Hintertüren die standardmäßig in Programmen oder Geräten vom Hersteller versteckt wurden. Meist handelt es sich um irgendeine Art von Wartungszugang oder Generalschloss um ausgesperrte Benutzer wieder zu entsperren. Leider ermöglicht dies auch Einbrechern diese Hintertürchen zu mißbrauchen. Sofern möglich, sollten sie Ihre Hardware auf Standardpasswörter oder Hintertüren untersuchen (zum Beispiel erstmal mit einer Suchmaschine danach suchen). Bekannt sind zum Beispiel Zugänge zum BIOS, wie »589589« für einige Award-BIOS oder »PASSWORD« für alte AMIBIOS-Modelle.

Trojaner sind Programme, die vorgeben etwas anderes zu sein, ähnlich dem Trojanischen Pferd des Epeios. So wird dem Anwender vorgetäuscht ein nützliches Programm oder interessante Daten (z. B. Tittenbilder) zu sein, während bei der Installation Spionageprogramme oder Hintertüren installiert werden.

SIEHE AUCH:
Kap. 10.5, »Den Rechner mit Chkrootkit auf Rootkits prüfen« auf Seite 144 ff.
Kap. 10.1, »Dateisystemintegrität mit Mtree prüfen« auf Seite 137 ff.
Kap. 10.2, »Dateisystemintegrität mit Aide prüfen« auf Seite 140 ff.
Kap. **??**, »**??**« auf Seite **??** ff.

3 VERTEIDIGUNG

Wer über gewisse Dinge den Verstand nicht verliert, der hat keinen zu verlieren.

(Gotthold Ephraim Lessing)

3.1 STRATEGIE, TAKTIK, OPERATIVE MITTEL

Um Sicherheit in einer größeren Organisation herzustellen, sind komplexe organisatorische Maßnahmen notwendig. Derartige Maßnahmen wurden und werden schon seit Menschengedenken untersucht, erforscht und angewandt. Bekanntestes und ältestes Beispiel ist die hohe Kunst des Krieges. Seit der Elektrifizierung und der damit verbundenen technischen Revolution ist Sicherheit auch in vielen anderen ingenieurwissenschaftlichen[1] Bereichen wichtig geworden. Insbesondere die Elektrotechnik, Luft- und Raumfahrttechnik, Automobilbau und im besonderen auch die Umweltforschung befassen sich mit Sicherheit und deren Folgen. Daher existieren schon seit einigen Jahrtausenden Überlegungen und Methoden um Sicherheit zu gewährleisten.

Die ältesteten und ausgefeiltesten Methoden bietet das Militär, da es jahrtausendelang wichtigstes politisches Instrument war und entsprechend gefördert wurde bzw. auch genügend zum praktischen Einsatz kam. Daher lassen sich Organisationsmethoden und -strukturen des Militärs erfolgreich und umfassend auf nicht-militärische Anwendungsgebiete übertragen.

In (Clausewitz 1832) definiert von Clausewitz: »Die Taktik ist die Lehre vom Gebrauch der Streitkräfte im Gefecht, die Strategie die Lehre vom Gebrauch der Gefechte zum Zweck des Krieges«. Damit lässt sich festhalten das die Strategie geplantes und langfristig ausgerichtetes Anstreben eines Ziels oder einer vorteilhaften Lage ist. Die Taktik setzt die längerfristigen Ziele der Strategie kurzfristig und an die Lage bzw. die verfügbaren Mittel angepasst um. Dazu beachtet sie die operativen Mittel, also

[1] Der »Ingenieur« geht auf den »ingeniarius« zurück, und das ist nichts anderes als ein Festungsbaumeister.

jene Mittel die zur Umsetzung der Strategie taktisch zur Verfügung stehen.

Das Verständnis dieser Vorgehensweise ist wichtig, denn nur im Zusammenspiel von strategischen, taktischen und operativen Maßnahmen über eine längerfristige und kurzfristige Planung ist es möglich in größeren Netzen Sicherheitsmaßnahmen zu implementieren.

3.1.1 *Erste organisatorische Schritte*

In der Tabelle 3.1 werden die von Sun Microsystems aufgestellten 10 Regeln für mehr Sicherheit aufgezählt. Diese Tabelle ist ein einfaches Mittel um eine einfache Strategie bzw. Taktik zu etablieren.

Wichtig hierbei sind zuerst Punkt 2 und 1, sorgen Sie in Ihrer Organisation dafür, das Zuständigkeiten und Verantwortlichkeiten zweifelsfrei geklärt sind. Sorgen Sie ggf. dafür das eine neue Abteilung gegründet wird, die sich um die IT-Sicherheit kümmern soll. Wenn Ihre Organisation hinreichend groß bzw. entsprechend mit Personal ausgestattet ist, beauftragen Sie mit dieser Aufgabe nicht nur Ingenieure oder Informatiker, sondern ebenfalls Juristen, Psychologen, Datenschutzbeauftragte und/oder Arbeitnehmervertreter (Betriebsrat etc.). Regeln Sie unbedingt Weisungsbefugnisse der Sicherheitsbeauftragten, denn oftmals gehen diese im Zuständigkeitswirrwarr und Kompetenzgerangel diverser »Würdenträger« unter. Achten Sie auch auf Interessenkonflikte und Kenntnisse der eingesetzten Mitarbeiter. Ein Systemadministrator der überall Root-Rechte hat und gleichzeitig Datenschutzbeauftragter ist wird früher oder später in Interessenskonflikte seiner beiden Positionen kommen. Ebenso ist ein völlig fachfremder Datenschutzbeauftragter, bspw. ein ausgeloster Buchhalter, oft nicht mit den technischen Möglichkeiten vertraut und somit leicht umgehbar.

Bei der Aufstellung von IT-Sicherheitsbeauftragten sollte unbedingt auf Redundanz geachtet werden. Ein einzelner Mitarbeiter wird unter Umständen mit den Aufgaben überlastet werden und kann auch ausfallen. Neben den üblichen Urlaubszeiten kann ein Mitarbeiter auch krank werden oder sogar sterben. Auf diesen Fall muß die Struktur der Abteilung vorbereitet sein, so das der Ausfall eines Führers nicht den Ausfall der Abteilung nach sich zieht.

Wenn sich die Sicherheitsbeauftragten mit dem eigentlichen Thema der Sicherheit befassen, müssen Sie dies aus einem globalen Blickwinkel tun, wobei sich hier »global« auf die Organisation bezieht und nicht auf alles was auf unserem Globus kreucht und fleucht. Es ist vital für die Sicherheit einer IT-Landschaft das sie unternehmensweit und interdisziplinär betrachtet wird, ggf. müssen angeschlossene Organisationen (z. B. Zulieferer oder externe Dienstleister) mit einbezogen werden. Es hat keinen Sinn eine Abteilung oder Filiale mit 23 verschiedenen Sicherheitsmechanismen abzudichten

und in der Nachbarfiliale nichts dergleichen zu tun.

Es wurde bereits dargelegt das Sicherheit ein Prozeß ist und niemals mit einhundertprozentiger Sicherheit gerechnet werden kann. Zudem ist in einem wirtschaftlichen Unternehmen (auch in einem privaten Haushalt) das Ziel Aufwand und Kosten zu minimieren gegeben. Daraus ergeben sich kostenbedingt Einschränkungen bei der Wahl operativer Mittel. Hierbei kommt es zu einem Optimierungsproblem, bei dem die besten Verteidigungsmittel ausgewählt werden müssen. Dies kann nur effizient und effektiv geschehen, wenn die Risiken mitsamt ihrer Wahrscheinlichkeit und den resultierenden Kosten abgeschätzt werden. Dazu priorisieren Sie die angebotenen Dienste und bewerten deren Wichtigkeit resp. den möglichen Schaden bei Ausfall. Anhand dieser Prioritätenliste können Sie die verfügbaren Mittel zuweisen.

Ihren gesamten Sicherheitsmaßnahmen muss eine wohldurchdachte Sicherheitsrichtlinie zugrundeliegen. Es hat absoult gar keinen Sinn in blindem Aktionismus Programme zu installieren oder irgendwelche Schlösser auszutauschen. Der größte Teil beim Etablieren von Sicherheitsmaßnahmen umfasst den Entwurf und die Planung der Maßnahmen. Halten Sie sich beim Implementieren der Maßnahmen an die Richtlinien – allerdings nicht sklavisch, sondern passen Sie ggf. die theoretischen Richtlinien der praktischen Realität an[2].

Passend dazu müssen alle Maßnahmen hinreichend dokumentiert werden. Allerdings sollte sich dies nicht nur auf die expliziten Sicherheitsmaßnahmen beschränken sondern auch normale administrative Aufgaben und besondere Vorkommnisse umfassen. Insbesondere wenn nicht nur eine Person als Administrator arbeitet ist eine korrekte und umfassende Dokumentation vital. Hierzu bieten sich datenbankbasierte Dokumentationssysteme an.

Arbeiten Sie proaktiv. Es hat keinen Sinn neue Systeme zu installieren und einzurichten und hinterher in halsbrecherischen Aktionen Sicherheitslücken zu stopfen. Überlegen Sie *vor* der Anschaffung und Einrichtung wie sich das neue System absichern und in die bestehende Sicherheitsrichtlinie integrieren lässt. Wählen Sie gegebenenfalls Architektur, Betriebssystem und/oder Anwendungsprogramme in Abstimmung mit der Sicherheitsrichtlinie aus.

Neben den technischen Maßnahmen umfasst die Sicherheitsrichtlinie auch soziale und juristische Richtlinien. Arbeiten Sie diese mit Fachleuten wie Juristen oder Psychologen aus und wenden Sie sie an. Dazu gehört auch das Ihre Mitarbeiter geschult und unterwiesen werden. Menschen sind in der Sicherheitskette meist das schwächste Glied. Insbesondere dann, wenn die Mitarbeiter Zweck und Sinn einer technischen Maßnahme nicht verstanden haben, wird es schwer diese durchzusetzen. Ein Beispiel ist die Wahl starker Passwörter. Ein starkes Passwort lässt sich selten leicht merken, daher neigen Menschen dazu es zu notieren. Das dies die

[2] Getreu dem Motto: Kein Schlachtplan überlebt den ersten Kontakt mit dem Feinde.

Sicherheitsmaßnahme »starkes Passwort« unterminiert ist meist nicht klar. Erläutern Sie daher den Sinn und Zweck der Maßnahme und legen Sie Konsequenzen offen. Neben juristischen und arbeitsrechtlichen Maßnahmen ist es meist wirksamer auf die entstandenen Gefahren hinzuweisen. Kein loyaler Mitarbeiter möchte gerne der Konkurrenz die Industriespionage ermöglichen indem er ein simples Passwort verwendet. Wenn er diese Zusammenhänge verstanden hat, akzeptiert er meist auch die Sicherheitsmaßnahmen.

Wie schon erläutert ist die Sicherheitsrichtlinie nicht für die Ewigkeit in Stein gemeißelt sondern muss regelmäßig überprüft und angepasst werden. Dies wird vereinfacht, wenn von vornherein die Mitarbeiter in die Entscheidungsfindung einbezogen und deren Wünsche und Probleme berücksichtigt werden. Man muss sich vergegenwärtigen das sich Sicherheitsmaßnahmen zuallererst gegen den berechtigten Benutzer richten und ihm das Leben erschweren. Werden hier nun zu restriktive Maßnahmen oktroyiert umgehen die Mitarbeiter sie. Dies ist nicht wünschenswert und nur bedingt durch Strafmaßnahmen zu korrigieren, außerdem möchte man ja nicht gegen die eigenen Mitarbeiter[3] arbeiten, sondern mit ihnen. Weiterhin verfügen die Mitarbeiter über praktisches Wissen und praktische Erfahrung, die in diesem Prozeß von großer Bedeutung sein können. Auf sie zu verzichten ist schlichtweg Verschwendung, erhöht ihre Beachtung doch Praxistauglichkeit und Effizienz der Maßnahmen. Außerdem hat dies auch Auswirkungen auf die Moral, das Arbeitsklima verbessert sich und die Bereitschaft mitentschiedene Vorgaben zu befolgen steigt.

Überwachen Sie alle Systeme auf besondere Vorkommnisse im laufenden Betrieb sowie weiterführende Informationsquellen zu Sicherheitslücken und neuen Angriffen. Klären Sie hierzu Verantwortlichkeiten rechtzeitig ab und sorgen Sie allen Eventualitäten vor. Wenn Ihre Systeme rund um die Uhr an 365[4] Tagen im Jahr verfügbar sein müssen, können die verantwortlichen Administratoren keinen normalen Acht-Stunden-Dienst schieben, sondern müssen über Bereitschaften oder Schichten zu jeder Tages- und Nachtzeit verfügbar sein.

Bilden Sie alle Mitarbeiter (also nicht nur die direkten Systemadministratoren und Sicherheitsfachleute) in Sachen Sicherheit weiter. Ständig werden neue Sicherheitslücken und Probleme in Programmen entdeckt. Es erschließen sich aber auch komplett neue Angriffswege und -strategien, so daß eine ständige Aktualisierung der Sicherheitsrichtlinien und Programme notwendig ist. Selbstverständlich gehört es auch dazu den beauftragten Mitarbeitern entsprechende Zeit- und Mittelkontingente zur Verfügung zu stellen. Es ist sinnbefreit eine Sicherheitsrichtlinie zu entwerfen und umzusetzen, wenn sie nicht weitergepflegt wird. Dazu gehört eben auch die Weiterbildung der Administratoren, also das diese bspw. ein Buch oder eine Zeitschrift lesen und nicht unmittelbar an der Tastatur sitzen und »arbeiten[5]«.

[3] Deswegen heißen sie ja auch *Mit*arbeiter und nicht *Gegen*arbeiter.

[4] 366 Tage in Schaltjahren.

[5] Leider haben EDV-ferne Entscheider oftmals nicht den wünschenswerten Überblick was Arbeit und Arbeitszeiten in der EDV angeht. Hier wäre manchmal sicherlich ein »Roter Knopf« des BofH

Beachten und Planen Sie das Ausscheiden, den Wechsel oder Rechteverlust von Mitarbeitern, insbesondere wenn diese hohe Positionen inne haben und über entsprechende Informationen verfügen. Etablieren sie entsprechende Prozeduren in Verbindung mit der Personalabteilung die frühzeitig auf derartige Ereignisse reagiert. Setzen Sie ggf. Berechtigungen mit zeitlicher Begrenzung ein, die regelmäßig erneuert werden müssen.

1. Gehen Sie das Thema Sicherheit von einer umfassenden, unternehmensweiten Perspektive aus an.
2. Schaffen eine eigene Abteilung, die für die IT-Sicherheit im gesamten Unternehmen zuständig ist.
3. Bewerten Sie Risiken durch die Bestimmung möglicher Bedrohungen.
4. Legen Sie Ihren Investitionen den reellen Wert wichtiger IT-Assets im Unternehmen zugrunde.
5. Definieren Sie Sicherheitsrichtlinien, um Risiken in sinnvoller Weise einzugrenzen und Ihre Geschäftsziele zu schützen.
6. Konzipieren, implementieren und dokumentieren Sie eine geeignete Sicherheitsarchitektur.
7. Überprüfen Sie die Architektur vorbeugend auf mögliche Sicherheitslücken und nehmen Sie gegebenenfalls nötige Veränderungen vor.
8. Beobachten Sie die Infrastruktur fortlaufend auf Abweichungen vom erwarteten Verhalten und reagieren Sie nötigenfalls umgehend.
9. Führen Sie Schulungen und Fortbildungen zu Fragen der Sicherheit für Ihre Mitarbeiter durch.
10. Halten Sie Ihre Sicherheitspraktiken stets auf dem neuesten Stand, indem Sie sie regelmäßig überprüfen und fortlaufend verbessern und weiterentwickeln.

Tabelle 3.1.: 10 Schritte auf dem Weg zu mehr Sicherheit (aus (Sun Microsystems 2005))

wünschenswert ;-)

3.2 VERTEIDIGUNG IN DER TIEFE

Spätestens seit dem Dreißigjährigen Krieg hat sich im Festungsbau das Konzept der Verteidigung in der Tiefe etabliert. Nachdem die Artillerie soweit fortgschritten war, das man mit ihr unter massiertem Einsatz Festungs-Werke sturmreif schießen konnte, mussten neue Wege in der Verteidigung gefunden werden. Schnell erkannten Ingenieure das die weitere Verstärkung einer einzigen Verteidigungslinie sinnlos war, da auch die Waffentechnik beständig verbessert wurde und Reichweite sowie Durchschlagskraft stiegen. Stattdessen ging man dazu über mehrere Verteidigungsringe aufzubauen die zusammen und unter gegenseitigem Schutz verteidigend gegen die Angreifer vorgingen. Insbesondere im Gefecht der verbundenen Waffen zeigte sich diese Taktik sehr wirkungsvoll.

Prinzipiell lässt sich das Prinzip in den Punkten aus Tab. 3.2 zusammenfassen. Dabei gilt als Grundsatz das jede einzelne Verteidigungslinie als unsicher betrachtet werden und durch eine andere Maßnahme gedeckt werden muss. Dies erfordert einen angemessenen Entwurf der Verteidigungsmaßnahmen mit ausgewogener Redundanz und Diversifikation der Maßnahmen, so daß Konsequenzen durch den Fall einer Linie abgemildert werden können.

Eine Verteidigungslinie lässt sich dabei einer der drei Kategorieren »menschlich«, »prozedural« oder »technologisch« zuordnen und arbeitet dabei »statisch« oder »dynamisch«. Ein Sicherheitsschloß an einer Tür ist eine technologisch-statische Barriere, ein Wachmann mit Schutzhund, der Ausweiskontrollen am Tor durchführt und ggf. Platzverweise aufgrund des Haus- und Schlüsselrechtes ausspricht ist eine dynamische, menschliche und prozedurale Barriere. Es zeigt sich also das eine Barriere durchaus gemischt sein kann. Wenn bei der Ausweiskontrolle technische Maßnahmen wie Iris-Abtastung und RFID-gesicherter Ausweis zum tragen kommen, hat man eine dynamische Barriere, die menschlich, prozedural und technologisch arbeitet. Barrieren richten sich in ihrer Wirkweise gegen den Angreifer (z. B. seine IP-Adresse wird blockiert) oder auf das zu schützende Gut (z. B. manipulierter Code wird entdeckt).

Neben der erstmaligen Implementierung von Barrieren müssen die existierenden Schutzmechanismen regelmäßg gewartet und aktualisiert werden. Jede einzelne Barriere kann mit den Mitteln der Risikoanalyse (z. B. Risikobaum) untersucht und bewertet werden. Ebenso sind erfolgte oder erfolgende Angriffe zu analysieren und aus der Analyse Rückschlüsse zu ziehen und ggf. Modifikationen an den Sicherheitsmaßnahmen vorzunehmen.

Mehrere Verteidigungslinien kommen gegen verschiedene Angriffe unterschiedlich zum Tragen. Ein Wurm der sich via Netzwerk verbreitet muss zuerst den Paketfilter überwinden und sich anschließend vor dem Virenscanner verstecken. Ein Virus der per Mail verteilt wird braucht sich vor dem Paketfilter nicht zu fürchten, da dieser

Emails nicht filtert.

1. Mehrere Verteidigungslinien sichern die zu schützenden Werte.
2. Jede einzelne Linie dient der globalen Verteidigung.
3. Jede einzelne Linie ist autonom und für sich gesehen unabhängig von allen anderen Linien dienstfähig. Der Fall einer Verteidigungslinie schwächt zwar die globale Verteidigung, eine einzelne Linie verfügt aber über alle Mittel um Angriffe anzuwehren.
4. Eine Linie wird mit allen Mitteln verteidigt.
5. Verteidigungslinien müssen koordiniert und geordnet betrieben werden.
6. Jede gefallene Linie muss den Angriff schwächen. Zum Beispiel in dem weitere Informationen über den Angreifer gewonnen werden und so die verbliebenen Verteidigungswerke verstärkt werden können.
7. Die gesamte Verteidigung erfolgt dynamisch und nicht statisch. Das heißt das Verteidigungsmaßnahmen geändert und der Lage angepasst werden müssen. Nichts ist so beständig wie die Lageänderung.
8. Defensive heißt nicht Verzicht auf Offensive.

Tabelle 3.2.: Prinzipien der Verteidigung in der Tiefe

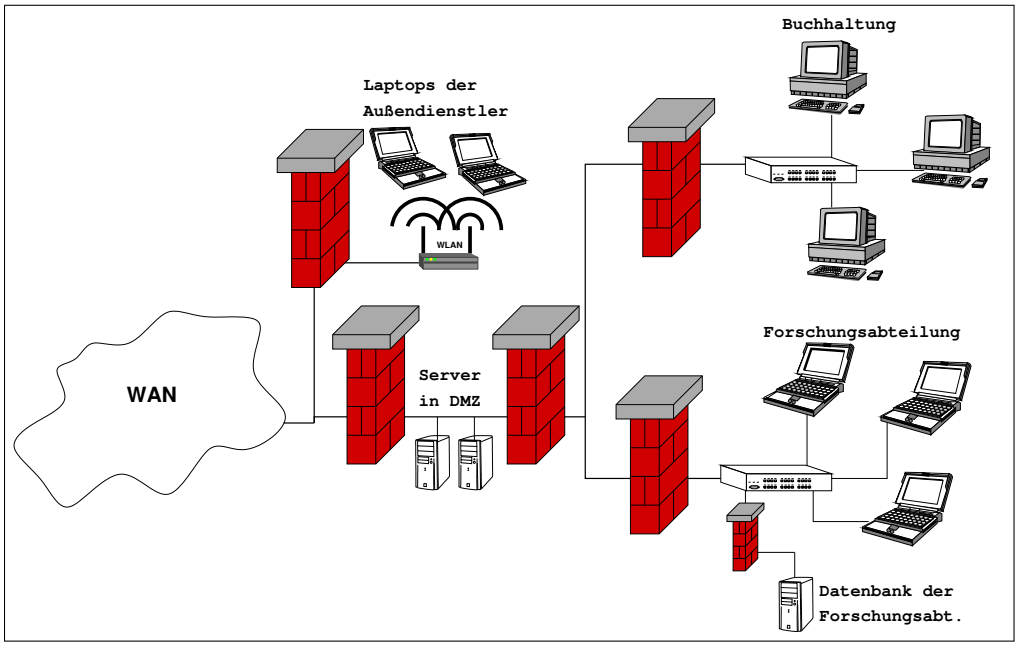


Abbildung 3.1.: Aus Filtergründen segmentiertes Netzwerk

3.3 Risiko, Risikobewertung und Risikofolgeabschätzung

Wie schon bei der Definition des Begriffes »Sicherheit« gezeigt wurde, bedingen »Risiko« und »Sicherheit« einander. Daher verwundert es auch nicht, das es keine klare Definition für ein Risiko gibt und meist auch nicht geben kann. Spielen zwei Mannschaften Fußball gegeneinander ist das Verliererrisiko der einen Mannschaft die Siegeschance der anderen. Es kommt also durchaus bei der Bewertung eines Risikos auf den persönlichen Standpunkt des Betrachters an. Untersucht man das Risikoverhalten der Menschen, kommt man unweigerlich zu dem Schluß das sich diese nicht immer rational verhalten. So ist das Risiko im Straßenverkehr zu Schaden zu kommen wesentlich höher als die Bedrohung durch Kernkraftwerke. Trotzdem werden KKW als gefährlicher eingestuft. Ein weiteres Beispiel für nicht-funktionierende Risikofolgenabschätzung ist der Verzicht auf Rauchmelder in Deutschland. Die Zwangseinführung von Rauch- bzw. Brandmeldern in den USA hat gezeigt das die Zahl der Feuertoten um mehr als 90% sinkt. Trotz der geringen Kosten von nicht einmal 5€ pro Melder setzen nur sehr wenige Haushalte Brandmelder ein.

Nichtsdestotrotz muß man als Ingenieur bei der Bewertung und Analyse von Risiken professionell und deterministisch vorgehen. Daher ist es angebracht mögliche Risikien

anhand ihrer Eintrittswahrscheinlichkeit und Schadenshöhe zu klassifizieren.

Dabei erkennt man relativ leicht das sich Risiken grob in drei Klassen einteilen lassen. Im Normalbereich liegen Risiken, die nur geringe Ungewißheiten in Bezug auf die Wahrscheinlichkeitsverteilung von Schäden und minimale Schwankungsbreiten von zu erwartendem Schaden und Eintrittswahrscheinlichkeiten. Weiterhin sind die Schäden weitgehend reversible und zeitliche sowie örtliche Ausdehnung beschränkt. Derartige Risiken lassen sich, wie in den Ingenieurwissenschaften oder der Wirtschaftswissenschaft üblich mit relativ einfachen mathematischen Methoden, der Multiplikation von Schadenshöhe und Eintrittswahrscheinlichkeit bemessen und bewerten. Ist dieser Faktor relativ gering, liegt das Risiko im Normalbereich. In der IT sind dies bspw. normale Hardwarefehler wie ein Plattencrash oder ein in Flammen aufgehendes Netzteil.

Darüberhinausgehende Risiken liegen vorerst im Grenzbereich. Hier liegen Eintrittswahrscheinlichkeit sowie die Schwankungsbreiten von potentiellem Schaden und der Eintrittswahrscheinlichkeit sehr hoch. Weiterhin sind zeitliche und örtliche Ausbreitung groß und bleibende Schäden hoch. Risiken im Grenzbereich sind schwer zu handhaben, da sie meist neben dem hohen Risiko und potentiellem Schaden auch eine hohe Ertragschance bieten.

Bietet ein Risiko im Grenzbereich zusätzlich zum hohen Risiko auch noch einen geringen Nutzen steigt das Risiko in den Verbotsbereich auf. Dieser Bereich ist leicht zu handhaben, wie der Name schon sagt, werden die hier verorteten Risiken schlichtweg verboten. Ein Beispiel wäre ein anonymer FTP-Server, auf dem jeder beliebig Daten ablegen kann. Diese Vorgehensweise wird verboten und gegen einen SFTP-Server ausgetauscht, auf dem sich Benutzer mit einem Passwort authentifizieren und authorisieren müssen. Der SFTP-Server läuft nun maximal im Grenzbereich.

Problematisch sind ebenso Risiken die eigentlich eindeutig im Verbotsbereich angesiedelt sind, deren Schaden aber erst stark zeitlich verzögert nach dem auslösenden Ereignis eintritt. Dies beeinflußt die Wahrnehmung der Risiken erheblich und beeinflusst deren Klassifizierung. Die hohe komplexität von Risiken, gerade in der IT, vereinfacht die Analyse und Entscheidungsprozeße ebenfalls nicht.

In (Wissenschaftlicher Beirat der Bundesregierung 1998) werden weiterhin auch die international gängigen Risikotypen eingeführt. Diese sind nach griechischen Sagengestalten benannt und werden Anhand der Eintrittswahrscheinlichkeit W und dem Schadensausmaß A klassifiziert. Die Risikotypen wurden zwar für Umweltrisiken wie Kernkraftwerke, Chemieanlagen oder Naturkatastrophen erstellt, lassen sich aber auch mit kleineren Einschränkungen und Anpassungen auf die IT-Sicherheit übertragen, daher werden sie in Tabelle 3.3 aufgeführt. In der Tab. 3.4 werden einzelne Schritte bei der Risikobewertung vorgestellt.

Damokles, bekannt durch die Sage vom Damokles-Schwert, bezeichnet ein Risiko, bei

dem mit geringer Wahrscheinlichkeit ein sehr hoher Schaden einhergeht. Beispiele sind Naturkatastrophen wie Meteoriteneinschläge oder Kernkraftwerke. In der eigentlichen IT sind derartige Risiken wenig verbreitet.

Ein *Zyklop* ist ein aus der Odyssee bekannter einäugiger Riese, der damit nur eine Seite der Wirklichkeit erfassen konnte. Der Risikotyp Zyklop zeichnet sich durch sehr hohen Schaden aus, der auch bestimmbar ist. Die Eintrittswahrscheinlichkeit des Schadens ist allerdings unbekannt. Beispiele sind Erdbeben oder Überschwemmungen.

Pythia, eine Seherin des Orakels von Delphi, war zwar die Bekannteste, allerdings gab sie ihre Antworten sehr unklar[6]. Dies führte dazu das der Ratsuchende falsche Schlüsse zog, wie bspw. Krösus, der sein gesamtes Reich in den Untergang führte. Pythia-Risiken zeichnen sich daher auch durch sehr hohe Ungewissheiten der Eintrittswahrscheinlichkeiten und Schadenshöhe aus. Beispiele sind plötzliche Klimaänderungen und bestimmte Genmanipulationen.

Pandora, die ihre berühmte Büchse öffnete und die Übel und die Welt entließ, ist Namensgeberin für eine weitere Klasse. In der Pandora-Klasse sind Auswirkungen eines Risikos unbekannt oder bestenfalls der Schädlichkeit verdächtig. Klar ist aber das diese Risiken irreversibel sind und sich sehr stark zeitlich und örtlich ausdehnen.

Kassandra-Risiken sind Risiken, die Eintrittswahrscheinlichkeit und das Schadenspotential sind recht hoch und bekannt, die Schäden treten aber erst in ferner Zukunft auf. Daher werden diese Risiken meist subjektiv nicht als gefährlich eingestuft, sind objektiv aber im Verbotsbereich.

Medusa-Risiken sind Risiken, die zwar objektiv recht gering sind, da sie meist noch im Grenzbereich liegen, aber subjektiv als äußerst gefährlich angesehen werden. Beispiele sind Elektrosmog.

Damokles und Zyklop sind spontan auftretende Risiken, Medusa, Kassandra und Pandora eher schleichend. Pythia hingegen umfasst hauptsächlich Unfälle.

Hauptproblem der Damokles-Klasse ist das hohe Schadenspotential. Als Gegenmaßnahmen kann man das Katastrophenpotential reduzieren, indem das Schadenspotential gesenkt und die Ausbreitung des Schadens begrenzt wird. Das technische System muss gegen Fehler geschützt werden, in dem die Fehleranfälligkeit gesenkt wird. Schlußendlich muss neben den risikoreduzierenden Maßnahmen das Risikomanagement durch Ausbildung und Bildung der Risikomanager ausgebaut werden.

Um die Zyklop-Risiken zu handhaben, ist es notwendig mehr Informationen über die

[6] In Wirklichkeit war sie durch natürliche Äthylen-Vorkommen zugedröhnt und gab von Spionen vorbereitete Antworten wieder.

Eintrittswahrscheinlichkeiten zu sammeln. Weiterhin gilt es Übwerachungsmaßnahmen einzusetzen und wie bei Damokles-Risiken Risikomanagement zu betreiben.

Pythia-Risiken erfordern neben der Erforschung der Eintrittswahrscheinlichkeiten auch umfangreiche Vorsorgemaßnahmen. Dies kann den Einsatz von Mitteln gemäß dem »Stand der Wissenschaft und Technik« beinhalten aber auf jeden Fall auch eine Eingrenzung des Risikobereichs.

Da Pandora-Risiken einen ungewöhnlich hohen Schaden aber nur unklare Eintrittswahrscheinlichkeiten haben, ist die oberste Maxime hier Ersetzung der riskanten Programme bzw. Stoffe. Weiterhin sind Risikopotentiale dadurch zu minimieren, das Risikoquellen örtlich und zeitlich eingegrenzt werden.

Kassandra erfordert eine besondere Strategie, da durch die hohe zeitliche Verzögerung des Schadens oft Verantwortlichkeiten ignoriert werden. Weiterhin gelten hier auch wieder Risikominimierungsmaßnahmen und Maßnahmen des Risikomanagements als Abwehrmaßnahmen.

Medusa erfordert eher soziale als technische Maßnahmen. Das Mobilisierungspotential ist relativ hoch und die subjektiv empfundene Bedrohung muss durch Aufklärung offensichtliche Abwehrmaßnahmen bekämpft werden.

Name	Art
Damokles	W gegen 0, $S(W)$ hoch A gegen ∞ $S(A)$ hoch
Zyklop	W ungewiß, $S(W)$ ungewiß A hoch $S(A)$ mittel bis hoch
Pythia	W ungewiß, $S(W)$ ungewiß A ungewiß, potentiell hoch $S(A)$ ungewiß
Pandora	W ungewiß, $S(W)$ ungewiß A ungewiß, nur Vermutung $S(A)$ ungewiß hohe Schadensdauer
Kassandra	W eher hoch, $S(W)$ eher gering A eher hoch $S(A)$ eher hoch hohe Verzögerungswirkung
Medusa	W eher gering, $S(W)$ eher gering A eher gering $S(A)$ hoch

W: Eintrittswahrscheinlichkeit, A: Schadensausmaß,
$S(W)$: Abschätzungssicherheit der Eintrittswahrscheinlichkeit
$S(A)$: Abschätzungssicherheit des Schadensausmaßes

Tabelle 3.3.: Die Risikotypen im Überblick: Charakterisierung und Beispiele. Nach (Wissenschaftlicher Beirat der Bundesregierung 1998)

- Hierarchisierung der Risiken
  - Schweregrad nach örtlicher und zeitlicher Begrenzung
  - Schweregrad nach Reversibilität
  - Eintrittswahrscheinlichkeit des Risiko unter Beachtung der etablierten Barrieren
- Einteilung in Risikobereich
  - Normalbereich
  - Grenzbereich
  - Verbotsbereich
- qualitive Analyse der Risiken
  - induktive Analyse von den auslösenden Ereignissen zum Schaden
  - deduktive Analyse vom Schaden zum auslösenden Ereignis (Fehlerbaum)
  - postmortale Analyse (Unfallgutachten o.ä.)

Tabelle 3.4.: Kategorisierung der Risiken

DAMOKLES

- Katastrophenpotential reduzieren
- Fehlertauglichkeit verbessern
- Katastrophenmanagement aufbauen

ZYKLOP

- Eintrittswahrscheinlichkeit erforschen und ermitteln
- Gegen Überraschungen vorbeugen
- Katastrophenmanagement aufbauen

PYTHIA

- Vorsorge verbessern
- potentiellen Schaden eingrenzen
- Wissen über das Risiko verbessern
- Katastrophenmanagement aufbauen

PANDORA

- Ersetzungsmöglichkeiten suchen
- Einsatz des Riskanten Stoffes eingrenzen oder ggf. verbieten
- Katastrophenmanagement aufbauen

KASSANDRA

- Verantwortlichkeiten festlegen
- Ersetzungsmöglichkeiten suchen und einsetzen
- Katastrophenmanagement aufbauen

MEDUSA

- Vertrauen bilden
- Wissen verbessern
- Kommunikation verbessern

Tabelle 3.5.: Risiko-Gegenmaßnahmen nach (Wissenschaftlicher Beirat der Bundesregierung 1998)

3.4 DIE SICHERHEITSRICHTLINIE

Das wichtigste Instrument im Kampf um die Sicherheit eines Systems ist die Sicherheitsrichtlinie. In ihr werden alle notwendigen Maßnahmen (Strategie, Taktik, operative Mittel) beschrieben und festgelegt. Sie stellt den Idealzustand der Sicherheit dar und beschreibt alle Informationen und Maßnahmen die notwendig sind, diesen Idealzustand zu erreichen. Die Sicherheitsrichtlinie dient dazu Vertraulickeit, Integrität, Verfügbarkeit und Authentizität *(VIVA)* aller Informationen sicherzustellen.

Die Richtlinie wird von der Leitung der Organisation verabschiedet und vorgelebt. Alle Mitglieder der Organisation müssen die Richtlinie zur Kenntnis nehmen und beachten. Nicht-Beachtung der Richtlinie muß strafbewehrt sein und entsprechend sanktioniert werden.

KLASSSIFIZIERUNG DER DATEN Zu allererst benötigen Sie eine Klassifizierung aller anfallenden Daten in Ihrer Organisation. Ordnen Sie den Daten einen finanziellen Wert und eine Geheimhaltungsstufe zu. Bei der finanziellen Bewertung geht es nicht darum, den tatsächlichen Wert auf Heller und Pfennig bestimmen zu können, sondern grob den Wert zu erfassen. Die Spezifikation eines neuen Programmes, an derm 20 Diplom-Informatiker 2 Jahre lang gearbeitet haben ist eben wesentlich mehr wert als das Telefonverzeichnis der Mitarbeiter. Lassen Sie die Bewertung gegebenenfalls von den Fachabteilungen vornehmen. Falls es keine direkte Kosten- und Leistungsrechnung gibt, überlegen Sie, wie lange wieviel Mitarbeiter arbeiten müssten, um die Daten neu zu erstellen. Multiplizieren Sie dann die Mannstunden mit den Stundensätzen der Mitarbeiter.

Für die Geheimklassifizierung benötigen Sie verschiedene Schutzstufen. Anstatt eigene zu entwickeln, können Sie die Stufen der Geheimschutzordnung des Deutschen Bundestages verwenden. Diese sind öffentlich dokumentiert und werden im öffentlichen Dienst, der Polizei und der Bundeswehr eingesetzt.

Die 4 Stufen sind:

1. **STRENG GEHEIM (str. geh.)** die Kenntnisnahme durch Unbefugte kann den Bestand oder lebenswichtige Interessen der Organisation gefährden.

2. **GEHEIM (geh.)** die Kenntnisnahme durch Unbefugte kann die Sicherheit der Organisation gefährden oder ihren Interessen schweren Schaden zufügen.

3. **VERSCHLUSSACHE – VERTRAULICH (VS-Vertr.)** die Kenntnisnahme durch Unbefugte kann für die Interessen der Organisation schädlich sein.

4. **VERSCHLUSSACHE – NUR FÜR DEN DIENSTGEBRAUCH (VS-NfD)** die

Kenntnisnahme durch Unbefugte kann für die Interessen der Organisation nachteilig sein.

Lassen Sie die Klassizifierung gegebenenfalls von den Fachabteilungen durchführen. Beachten Sie dabei auch gesetzliche Regelungen, wie das Datenschutzgesetz. Allein die Zuordnung eines Geburtsdatums zu einem Namen unterliegt der Datenschutzstufe 1 und ist damit VS–NfD!

Markieren Sie *alle* Medien und Akten mit der höchsten Geheimhaltungsstufe, die in ihnen vorkommt. Nutzen Sie dazu ein standardisiertes Verfahren, wie Deckblätter, farbige Etiketten oder ähnliches. Markieren Sie in Ausdrucken jede einzelne Seite, beispielsweise mit Wasserzeichen oder einer Kopf- bzw. Fußzeile. Verwenden Sie farbiges Papier für die Ausdrucke und Kopien. Markieren Sie alle Medien und Ausdrucke von vornherein mit einem Verfallsdatum.

KLASSIFIZIERUNG DER MITARBEITER Legen Sie fest, welcher Mitarbeiter welcher Sicherheitsstufe unterliegt. Überprüfen Sie, welche Daten der Mitarbeiter zur Erfüllung seiner Aufgaben benötigt. Dokumentieren Sie diesen Bedarf. Grenzen Sie die Sicherheitsstufen auf den Arbeitskreis des Mitarbeiters ein. Es ist nicht notwendig das ein Entwicklungsingenieur auf Personaldaten zugreifen kann. Begrenzen Sie gegebenenfalls die Dauer einer Freigabe. Wenn ein Entwicklungsingenieur für ein bestimmtes Projekt Daten braucht, darf sich die Freigabe nur für die Laufzeit des Projektes auf die Projektdaten beziehen.

PHYSIKALISCHE ABSICHERUNG Die physikalische Sicherheit der Rechner und Daten ist ebenso wichtig wie die Netzwerksicherheit. Die meisten Rechenzentren verfügen über eine adäquate physikalische Sicherung durch Lambertz-Zellen und vielschichtigen Zugriffsschutz, allerdings werden häufig die normalen Büroräume weniger gut abgesichert. Dabei bietet oft schon der Zugang zu einem normalen Büro oder Besprechungsraum den Zugang zum internen Netzwerk oder anderen wichtigen Informationen.

Sorgen Sie dafür, das sensible Daten, Rechner und Akten in entsprechend geschützten Räumen gelagert werden. Halten Sie Publikumsverkehr aus diesen Bereichen fern. Setzen Sie nötigenfalls mehrschichtige Authentifikationsmechanismen, wie Magnetkarten und Passwörter oder Wachleute ein. Sichern Sie insbesondere Telefon- und Netzwerkverteiler und die Poststelle/-körbe ab.

Schützen Sie auch Papierkörbe in denen sensibles Material lagert. Verwenden Sie abschließbare Modelle, die ähnlich einem Briefkasten funktionieren.

Protokollieren Sie den Zutritt zu besonders sensiblen Bereichen und setzen Sie Videokameras zur Überwachung ein.

3.5 Inventur der vorhandenen Systeme

Grundlage der Planung ist ein umfassendes Inventarverzeichnis der eingesetzten Systeme. Neben der eingesetzten Hardware und er Netztopologie ist hierbei insbesondere auch der der Einsatz der Systeme, also laufende Anwendungsprogramme und deren Einsatzzweck.

Zuerst gilt es die Netztopologie zu erfassen, dies kann graphisch oder tabellarisch Erfolgen, wobei aus einer Tabelle auch ein Graph generiert werden kann. Dabei sind folgende Systeme und Eigenschaften von Interesse:

- Systeme wie Server, Workstations, Router, Switches, Bridges, Access Points, NAS, Netzwerkdrucker und andere Komponenten
- Netzwerkverbindungen der Komponenten, LAN-Verbindungen (Ethernet, WLAN), Backbone-Techniken, evtl. Tunnel wie VPN
- Ausgänge nach außen, Modem, ISDN, DSL, Standleitungen, Tunnel durch das Internet zu anderen Orten, Funkstrecken

Für jedes System benötigt man diverse Eigenschaften:

- Identifikationsmerkmal (Inventarnummer, Hostname, IP o.ä.)
- Funktion (Webserver, Mailserver, Gateway, Backupserver, Workstation ...)
- Architektur und Betriebssystem, ggf. Änderungen am Betriebssystem erfassen (eigener Kernel o.ä.)
- Standort (Gebäude, Raumnummer, Schrank o.ä.)
- Benutzer und verantwortliche Systemadministratoren
- Netzadresse
- Kommunikationsschnittstelle
- Medientyp
- Netzprotokolle

- ggf. Details zum externen Netz (ISP, Kontaktdaten)
- Schutzbedarf des Systems

SIEHE AUCH:
(Secrétariat général de la défense nationale 2004)
(Sun Microsystems 2005)
(Systems and Network Attack Center 2001)
(Systems and Network Attack Center 2006)
(Albanese u. a. 2004)
(Mackey 2002)
(OECD 2003)
(Wissenschaftlicher Beirat der Bundesregierung 1998)
(Wiedemann u. a. 2002)
(Bundesamt für Sicherheit in der Informationstechnik 2006b,c,d)
(Bundesamt für Sicherheit in der Informationstechnik 2006a)

3.6 BEHANDLUNG EINES SICHERHEITSVORFALLS

Es ist wichtig, das sich die IT-Sicherheitsverantwortlichen auch auf den Eintritt eines Sicherheitsvorfalls und dessen Nachbereitung vorbereiten. Schließlich zeigt ein erfolgreicher Einbruchsversuch das die bisherige Verteidigungsstrategie nicht funktioniert hat.

3.6.1 *Digitale Forensik*

Die digitale Forensik befasst sich mit der Untersuchung und Beweissicherung auf IT-Systemen nach Einbrüchen oder anderen Sicherheitsvorfällen. Im kriminalistischen Umfeld wird die digitale Forensik meist eingesetzt um auf IT-Systemen Beweise für nicht-Computer-Verbrechen zu finden, da diese oftmals mithilfe der IT geplant und durchgeführt werden.

Forensische Maßnahmen sind angebracht bzw. zwingend erforderlich wenn es zu einem Sicherheitsvorfall gekommen ist. Nur durch die möglichst lückenlose Aufklärung des Tathergangs ist es möglich die verwandten Sicherheitslücken zu identifizieren und zu stopfen. Die Beweissicherung kann ebenfalls in einem Gerichtsverfahren von Bedeutung sein. Normalerweise unterliegen Logdateien oder ähnliche Spuren der freien Beweiswürdigung, d.h. sie werden vor Gericht wie eine Zeugenaussage behandelt. Möchten Sie forensische Maßnahmen in einem Gerichtsprozeß nutzen

bzw. durchführen setzen Sie sich rechtzeitig, also vor Untersuchungsbeginn, mit einem Rechtsanwalt und/oder Notar in Verbindung. Dieser kann Ihnen helfen die sichergestellten Beweise gerichtsverwertbar zu halten.

Wichtigste Grundlage der Forensik ist die Spurensicherung. Dazu notwendig sind vor allem Log-Dateien des Betriebssystems und der Anwendungsprogramme, daher müssen Sie bereits von Anfang an möglichst umfangreich vom System bzw. den laufenden Diensten Protokolldateien erstellen lassen und diese durch Rotation und andere Maßnahmen (Sicherung, Verschlüsselung) vor Manipulation schützen. Siehe dazu auch Kap. **??**, »**??**« auf Seite **??** ff..

Das wichtigste bei einer forensischen Untersuchung ist überlegtes Handeln und ruhiges Vorgehen. Also: keine Panik und das Handtuch nicht vergessen. In der Untersuchungsmannschaft sollten nur erfahrene Administratoren eingesetzt werden, die mit den Gegebenheiten des inkriminierten Systems vertraut und in der Lage es zu analysieren sind. Auch hier gilt das alte Sprichwort, das zu viele Köche den Brei verderben, setzen Sie daher nur so viele Forensiker wie nötig ein. Es bietet sich aber an mindestens zwei Mitarbeiter einzusetzen, da so das Vier-Augen-Prinzip greift.

Bevor das System untersucht wird, führen Sie eine Hardware-Inventur durch. Protokollieren Sie diese in einem Logbuch. Dieses enthält numerierte Seiten und jeder Eintrag muss ausführlich und präzise sein. Dazu wird er mit einem Datum versehen. Vermerken Sie ebenfalls die Differenz von Realzeit und Systemzeit des betroffenen Systems. Führen Sie das Logbuch sorgfältig und sofort – nicht erst im Nachhinein.

Vermeiden Sie bei der Untersuchung unbedingt jegliche Änderung am System. Reißen Sie auf keinen Fall den Netzstecker oder das Netzwerkkabel heraus und spielen sie auch keine Sicherung ein. Dies vernichtet wertvolle Informationen und verhindert so die Sicherheitslücke aufzufinden und zu beheben.

Sammeln Sie zuallererst Daten vom System, übertragen Sie diese auf ein Untersuchungssystem und arbeiten Sie nur mit diesen Kopien. Verwenden Sie zur Datensammlung eine CD/DVD mit einem NetBSD-System, welches alle notwendigen Binaries statisch gelinkt enthält. Es könnte sein, das der Einbrecher ein Root-Kit installiert hat, welches die System-Binaries gegen manipulierte austauscht.

Sichern Sie die Daten vom System in der Reihenfolge ihrer Flüchtigkeit:

1. Register-/Cacheinhalte
2. Arbeitsspeicher
3. Netzwerzkustand (ifconfig(1), netstat(1))

4. laufende Prozeße (ps(1), fstat(1), sockstat(1), pstat(1), modstat(1))
5. Festplatten (dd(1), dd_rescue)
6. überschreibbare Medien (Diskette, Magnetband, USB-Stick . . .)
7. nicht-beschreibbare Medien (CD, DVD,Ausdrucke)

Führen Sie das Logbuch parallel zu Ihrer Arbeit. Loggen Sie Ausgaben am Terminal oder unter X mit. Dazu können Sie script(1) verwenden, das alle Terminalausgaben in die Datei »./typescript« schreibt, oder xwd(1), das einen Screenshot vom X-System macht.

Um Daten auf ein anderes System zu loggen, verwenden Sie netcat/cryptcat oder SSH. Bsp. 3.1 zeigt verschiedene Anwendungsbeispiele. Zuerst müssen Sie, wie im ersten Bsp. gezeigt, Netcat auf dem Zielrechner starten und an einen Port binden. Anschließend können Sie auf dem Quellrechner beliebige Datenströme an den Zielrechner schicken. Dies erfolgt allerdings unverschlüsselt, so das im zweiten Bsp. bdes(1) verwandt wird, um die Daten vor der Übertragung zu verschlüsseln bzw. nach der Übertragung zu entschlüsseln. Als verschlüsselnde Alternative steht das funktionsgleiche cryptcat (Bsp. drei) oder SSH zur Verfügung.

[!h]

```
stefan@zielrechner$  nc -l -p 50000
quellrechner$ echo "Hallo Welt" | nc 127.0.0.1 50000
Hallo Welt
stefan@zielrechner$

zielrechner$ nc -l -p 50000 | bdes -a -d -k passwort > /tmp/log
quellrechner$ echo "Hallo Welt" |  bdes -a -k passwort | nc 127.0.0.1 50000
Hallo Welt
stefan@zielrechner$

quellrechner$ echo "Hallo Welt"  | cryptcat 127.0.0.1 50000
zielrechner$ cryptcat -l -p 50000
Hallo Welt
stefan@zielrechner$

quellrechner$ echo "Hallo Welt" | ssh stefan@127.0.0.1 "cat > /tmp/test"
```

Listing 3.1: Daten durch ein Netzwerk übertragen

In Bsp. 3.2 werden einige Möglichkeiten gezeigt, Systemdaten zu sichern. Verwenden Sie netcat/cryptcat um die Log-Dateien auf ein anderes System zu schreiben.

Nachdem der betroffene Rechner in seiner Umgebung untersucht wurde, ist es ratsam eine bitweise Kopie der Festplatte zu erstellen. Dies geschieht am besten mit dd(1)

[!h]

```
# ps -axu >  /tmp/log/ps
# ls -lR /proc/ >  /tmp/log/prcofs-ls
# modstat >  /tmp/log/kernel-mod
# ifconfig -a >  /tmp/log/netzwerk
# netstat -ai » /tmp/log/netzwerk
# arp -a » /tmp/log/netzwerk
# route show » /tmp/log/netzwerk-route
# netstat -nr » /tmp/log/netzwerk-route
# fstat >  /tmp/log/dateien
# last>  /tmp/log/benutzer
# who» /tmp/log/benutzer
# w» /tmp/log/benutzer
# fdisk>  /tmp/log/platten
# pkginfo |sort >  /tmp/log/pakete
```

Listing 3.2: Systeminformationen sichern

oder dem Paket dd_rescue, welches auch beschädigte Dateisysteme kopieren kann. Die bitweise Kopie der Festplatte kann anschließend untersucht werden, außerdem werden in ihr die Zugrifsszeiten der Dateien nicht geändert, und freigewordene Datenblöcke sowie nichtbelegte Block-Reste gesichert. Die Kopie kann anschließend auf einem Analyse-System oder einem Duplikat des betroffenene Systems untersucht werden. Entweder wird sie dazu auf eine leere Festplatte übertragen, in einem Emulator gestartet oder via vnd(4) eingebunden.

Nachdem die Daten des Systems gesichert wurden, gilt es andere Datenquellen auszuwerten bzw. sicherzustellen. Hierzu zählen Logserver, auf die per Syslog oder ähnlichen Programmen Logdateien geschrieben werden, sowie Logs von Routern, Switches, Bridges und Firewalls. Anschließend müssen IDS wie Snort an ihren Konsolen ausgewertet werden. Hostbasierte IDS wie Dateisystemintegritätsprüfer (Mtree, Aide, Tripwire) sollten auf der Kopie des Systems ausgeführt werden, da sie die Zugriffszeiten der Dateien ändern. Dies setzt voraus, das sie nach der Installation des Systemes Mtree eingerichtet haben und die Datenbank geschützt wurde.

Zusätzlich zu den gesicherten Systeminformationen sollten Sie das befallene System auch von außen mit nmap(1) scannen und mit den Ausgaben von netstat(1) vergleichen. Dies zeigt oft ob ein System trojanisiert wurde.

Nachdem Sie alle Daten vom Zielsystem sichergestellt haben, können Sie sie analysieren. Dazu untersuchen Sie zuerst die gesicherten Konfigurationsdateien (/etc/passwd, /etc/group, /etc/rc.d/ usw. usf.) auf Veränderungen. Anschließend verwenden Sie Mtree (oder Aide bzw. Tripwire) um Dateien auf Veränderungen zu prüfen. Untersuchen Sie Textdateien mit vi(1) oder grep(1) und Binärdateien mit strings(1). Ebenso ist die Shell-History (.history, .bash_history) eine wichtige Informationsquelle

4

SOCIAL ENGINEERING

Der Mensch ist ein Seil, geknüpft zwischen Tier und Übermensch, ein Seil über einem Abgrunde.

(Friedrich Nietzsche)

Social Engineering ist eine Angriffsstrategie, die nicht die Technik als Opfer auserkoren hat. Stattdessen wird hier viel lieber – und vor allem effizienter – das schwächste Glied in der Sicherheitskette angegriffen: der Mensch.

Ein Angreifer verwendet verschiedene Strategien und Taktiken um aus Benutzern der Systeme Informationen herauszuholen. Mithilfe dieser Informationen kann er erfolgreiche Angriffe gegen Zielsysteme fahren.

Derartige Techniken sind äußerst erfolgreich wenn größere Organisationen wie Großunternehmen, Behörden oder Universitäten angegriffen werden sollen. Oftmals gelingt es einem Angreifer mit der Aggregation öffentlich zugänglicher Informationen und entsprechender Kaltschnäuzigkeit wichtige Informationen und Zugriff auf geschützte Systeme zu erlangen.

Die Motivation für einen Angriff kann unterschiedlich sein, neben »professionellen« Gründen wie Industriespionage oder Identitätsdiebstahl kommen auch soziale Gründe wie Rache (z. B. durch Ex-Mitarbeiter) oder Spaß und Machtgefühl in Frage.

Eine weitverbreitete Masche ist das sogenannte Phishing, bei dem mit gefälschten Emails oder Webseiten Benutzer dazu verleitet werden sollen, Benutzerdaten zu ihren Bank- oder Emailkonten anzugeben. Derartige Angriffe sind sehr leicht durchzuführen, da man dazu lediglich etwas Webspace benötigt und dort die entsprechende Webseite nachahmt. Leider fallen immer wieder viele Benutzer auch auf holprige Emails mit Stilblüten wie »Zu der Beachtung!« – eine wortwörtliche Übersetzung des englischen »To the Attention!« – herein und offenbaren ihre Bankdaten.

Das »richtige« Social Engineering, Human Based Social Engineering genannt, setzt zum Großteil auf soziale Beziehungen als Angriffsvektor.

Zuerst benötigt der Angreifer möglichst viele Informationen über die anzugreifende

Organisation. Diese kann er aus freien Informationen, wie bspw. der Webseite oder Werbebroschüren, zusammensammeln und sich so ein Bild machen. Man kann auch den Müll durchwühlen und so an relevante Daten kommen. Außerdem können gewieftere Angreifer über Email oder Telefon Kontakte zu Mitarbeitern herstellen und sich als jemand anderes ausgeben. Als ein Kunde oder Vorgesetzter bspw., der dringend irgendwelche Informationen braucht. Mit diesen kann er dann andere Opfer beeindrucken oder einwickeln. Vorsicht ist daher mit der Freigabe von Informationen geboten, auch solchen, die auf den ersten Blick unverfänglich erscheinen. So mag es für die Mitarbeiter nützlich sein, das alle Mitarbeiter des Rechenzentrums inklusive Telefonnummer, Emailadresse, Aufgabengebiet und Foto auf einer Webseite aufgeführt werden. Wird diese Seite jedoch weltweit freigegeben, kann sich ein Angreifer sehr nützliche Informationen über die Struktur des Rechenzentrums beschaffen. So kann er anderen Mitarbeitern am Telefon bspw. intime Kenntnisse des Betriebs vorgauckeln oder schneller Kontakte knüpfen. Weiß der Angreifer bspw. das ein Mitarbeiter Pferdenarr ist, kann er Termindruck vortäuschen, weil sein eigenes Pferd dringendst zum Tierarzt muss. Das Opfer der Attacke wird als Pferdefreund sofort Sympathie für den Angreifer empfinden und versuchen ihm zu helfen.

Dieses Sympathie-Aufbauen lässt sich auf verschiedenen Wegen durchführen, bspw. mit der Verbrüderung von Opfern, indem man einen gemeinsamen Gegner vorgibt: »Die im Rechenzentrum haben wieder mal den Server abstürzen lassen ...« oder »Das dumme Word hat mal wieder die Datei zerstört ...« führen in der Regel zur Solidarität des Opfers – und damit möglicherweise zur Herausgabe der Daten.

Andere Maschen sind Vorspiegelung von Autorität, was insbesondere in strengen Hierarchien wie Polizei, Armee oder Feuerwehr sehr gut funktioniert. Selbst jemand der nur Grundwehrdienst geleistet hat, kann mit etwas Geschick und Witz als Offizier auftreten und eine Dienststelle auseinandernehmen. Wer das nicht glaubt, sollte einmal die Abenteuer des Gefreiten Asch in Hans Helmut Kirst 1954 nachlesen oder den Hauptmann von Köpenick anschauen. Aber auch in normalen Unternehmen gibt es Hierarchien, die sich ausnutzen lassen, so kann sich der Angreifer als wichtiger Kunde oder hoher Verantwortlicher einer anderen Filiale ausgeben. Unter Aufbau einer passenden Drohkulisse – »Wenn wir ihretwegen den Kunden verlieren, dann . . . !«, »Es kann ja wohl nicht sein, das Sie Ihren Laden nicht beisammen haben und mir die Akten nicht geben wollen!« oder »Wenn das Dr. Müller-Lüdenscheid bei unserem nächsten Golfturnier erfährt, wird er bestimmt sehr sauer sein!« – kann man hier Mitarbeiter zur Herausgabe von Daten bewegen.

Gerade bei Telefonaten muss der Angreifer über rhetorisches Geschick und Intelligenz verfügen. Man muss das Opfer in ein Gespräch verwickeln und eher nebenher nach den wichtigen Informationen fragen. Es ist außerdem auch von Vorteil, den Kontakt nicht abreißen zu lassen. So kann man als geplagter Pferdefreund aus dem obigen Beispiel ja man nach zwei Wochen wieder beim Opfer anrufen und freudestrahlend von der Genesung des Gauls berichten – dies führt zu einer Festigung der Beziehung.

Es gibt keine rein-technischen Mittel gegen Social Engineering[1], da das Angriffsziel nunmal ein Soziales ist.

Gegen soziale Attacken können soziale Maßnahmen helfen, dies beinhaltet Verträge, Richtlinien, Gebote, Verbote und andere Maßnahmen. So kann man bspw. festlegen das am Telefon niemals das Rechenzentrum nach einem Benutzerpasswort fragen wird und Benutzer daher niemals ihr Passwort herausgeben sollen.

Das wichtigeste Instrument im Kampf gegen soziale Angriffe ist Wissen – nur wenn der Mitarbeiter weiß, das es Social Engineering gibt und das Bösewichte soetwas ausnutzen können, wird er sich Gedanken zum Thema machen. Außerdem ist es wichtig zu wissen, das auch scheinbar harmlose Informationen zu Sicherheitslücken führen können. Schulen Sie alle Mitarbeiter zu diesem Thema. Je höher die Stellung und die Sicherheitsstufe des Mitarbeiters ist, desto intensiver muss die Schulung sein. Und vergessen Sie dabei die Sekretärinnen und Assistenten nicht! Diese verfügen in der Regel über umfangreiches Wissen, werden meist aber von den »Führungskräften« ignoriert.

Grenzen Sie die Anzahl der Geheimnisträger ein. Informationen die ich nicht habe, kann ich nicht weitergeben.

Vernichten Sie Daten sicher, auch offenbar harmlose Datenträger oder Akten können einem Angreifer wertvolle Informationen liefern. Mehr dazu finden Sie im Kapitel !!!

Etablieren Sie in Ihrer Organisation ein Sicherheitsbewußtsein. Machen Sie Ihren Mitarbeitern klar, das es Angriffe gegen die IT und gegen die Mitarbeiter selbst geben wird. Erläutern Sie die möglichen Angriffswege und -taktiken und bereiten Sie Ihre Mitarbeiter darauf vor. Etablieren Sie entsprechende Meldewege. Sorgen Sie auch dafür, das nicht authentifizierte Anrufer keine Daten oder Informationen am Telefon bekommen. Und sichern Sie Ihre Mitarbeiter gegen die Vorgesetzten ab. Wenn ein Mitarbeiter dafür bestraft wird, am Telefon keine Daten herausgerückt zu haben, brauchen Sie nicht einmal mehr davon träumen, Ihre Organisation abzusichern! Sorgen Sie dafür, das Sicherheitsbewußtsein ein Teil der Firmenkultur wird. Dies funktioniert nicht nur über das Rechenzentrum, sondern muss von jeder Abteilung – inklusive den Vorgesetzten – mitgetragen werden. Viele Manager verstehen die technischen Details oder Hintergründe nicht. Wenn das der Fall ist, argumentieren Sie nicht mit der Technik, sondern mit einer Sprache, die sogar Manager verstehen: €€€. Zeigen Sie an einem Beispiel was es das Unternehmen kosten kann, wenn ein Konkurrent an bestimmte Daten kommt. Es gibt einige dokumentierte Fälle von Industriespionage, diese können Sie als Beispiele anbringen.

[1] *»If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology«* - Bruce Schneier

4.1 psychologische grundlagen der manipulation

Ein Social Engineer (oder auch Trickebtrüger oder Hochstapler) verwendet als Angriffstechnik verschiedene Beeinflussungsmethoden. Robert Cialdini, Professor für Psychologie an der Arizona State University, beschreibt in seinem Buch (**cialdini**) die psychologischen Mechanismen des Überzeugens:

Der Mensch ist ohne Zweifel das am höchsten entwickelte Lebewesen auf dieser Welt. Er verfügt über eine enrome psychische Energie und Kapazität – da wir aber auch in der komplexesten und schnellebigsten Welt seit Menschengedenken leben, benötigen wir immer häufiger Faustregeln. Diese Faustregeln ersparen uns viel Zeit und Arbeit und erlauben uns so, relativ schnell und einfach Entscheidungen zu treffen. Dabei ersparen Sie uns viel mentale Arbeit und führen meist auch noch zum Erfolg.

Unsere Faustregeln und Stereotype sind feste Handlungsmuster, also *Verhaltensweisen*, die praktisch jedesmal gleich ablaufen. Sie verkürzen und vereinfachen anhand von *Urteilsheuristiken* die gedanklichen Prozesse zur Entscheidungsfindung. Hervorgerufen werden sie durch ein oder mehrere *Auslösemerkmale*. Man bezeichnet eine derartig mechanische Reaktion auf bestimmte Informationen als *automatisches Verhalten*. Reaktionen anhand einer gründlichen Analyse aller verfügbaren Informationen sind *kontrolliertes Verhalten*.

Unter anderem von **chen** wird nachgewiesen, das Menschen Entscheidungen viel bewusster treffen, wenn Sie motiviert und fähig zu einer sorgfältigen Analyse sind. Diese Erkenntniss sollte man im Hinterkopf behalten, wenn es um die Abwehr sozialer Angriffe geht. Je wichtiger eine Sache für einen Menschen ist, desto weniger wahrscheinlich ist kontrolliertes Verhalten. Allerdings muss er dazu auch in der Lage sein, die Situation einschätzen und Handlungen durchspielen zu können. Nicht umsonst suchen viele Steuerpflichtige einen Steuerberater auf und lassen ihn die Entscheidungen fällen, wie die Steuererklärung auszusehen hat.

Das automatische Reaktionen weitreichende Folgen haben können, zeigt **watson**. John Watson Jr., der spätere IBM-Chef, war während des Zweiten Weltkrieges dafür zuständig Unfälle der Luftwaffe, bei denen hohe Offiziere verletzt wurden, zu untersuchen. Im Fall von Uzal Ent kam einer automatischen Reaktion eine fatale Bedeutung zu. Der Kopilot des Generals hatte sich kurzfristig krankgemeldet und wurde durch einen anderen Piloten ersetzt. Während des Starts begann Ent ein Lied zu summen und nickte dazu rhythmisch mit dem Kopf. Der Kopilot zog daraufhin das Fahrwerk ein. Die Maschine verunglückte, ein Propeller löste sich und schoss durch die Kabine. Uzal Ent wurde dabei schwer verletzt und beidseitig gelähmt. Watson befragte den Kopiloten, warum er das Fahrwerk eingezogen hatte, obwohl er ganz genau wusste, das die Flughöhe noch nicht erreicht war. Seine Antwort war simpel: »Ich dachte, der General wollte, dass ich das tat.«.

Sicherlich ein dummes Verhalten, wenn man aber die psychologischen Reaktionen und Verhaltensweisen kennt, verständlich. Mehr noch, man kann nicht nur solche Reaktionen besser verstehen, sondern auch selbst herbeiführen. Neben allerlei zwielichtigen Gestalten – vom Social Engineer über den Trickbetrüger hin zum Hochstapler – nutzen auch andere dieses Verhalten: Verkäufer und Marketingabteilungen. Dazu ist es lediglich notwendig, die Auslösemerkmale für automatische Handlungsmuster zu kennen und geschickt einzusetzen.

Ziel der nächsten Kapitel ist es, diese Auslösemerkmale näher vorzustellen. Dies soll Ihre Anfälligkeit für derartige Manipulationsversuche vermindern. Dazu werden auch Abwehrstrategien vorgestellt. Außerdem lassen sich die vorgestellten »psychologischen Taschenspielertricks« im positiven Sinne nutzen – wenn Sie beispielsweise das Sicherheitsbewusstsein Ihrer Mitarbeiter anheben wollen oder mit dem Management um Ressourcen ringen müssen.

4.1.1 *Reziprozität*

Die Regel der Reziprozität (Wechselseitigkeit) besagt, das wir uns für erhaltene Gefälligkeiten, Geschenke und dergleichen revanchieren müssen. Die Etablierung dieser Regel hatte für die entsprechende Gesellschaft enorme Vorteile: Eine Person konnte einer anderen Person etwas geben, ohne es wirklich wegzugeben. Aufgrund der Reziprozität konnte der Schenker vom Beschenkten in Zukunft eine Gegenleistung erwarten. Ein hoch entwickeltes System der Reziprozität ermöglichte den Ausbau des Handels, der Verteidigung und der gegenseitigen Hilfeleistung.

Da diese Regel enormes Potential für eine Gesellschaft birgt, setzt sie alles daran ihre Mitglieder entsprechend zu sozialisieren. So werden Menschen, die nur nehmen anstatt zu geben sehr schnell ausgegrenzt – daher zahlt man manchmal beim Zurückgeben eher etwas drauf, als in den Verdacht zu geraten, Egoist zu sein. Erwartung der Gegenleistung gilt nur dann nicht, wenn der Beschenkte nicht in der Lage ist, sich entsprechend zu revanchieren – schließlich würde niemand ein Gegengeschenk erwarten, wenn er seiner dreijährigen Nichte einen Teddybären schenkt. Erstaunlicherweise gilt dies aber auch umgekehrt: Schenker, die dem Beschenkten erst gar keine Möglichkeit zur Revanche geben, sind genauso verpönt.

Die Durchschlagskraft dieser Regel ist beeindruckend, sie wurde in Experimenten wie von **regan** bewiesen. Zwei Experimentteilnehmer sollten einige Bilder bewerten. Einer der Teilnehmer, der in Wirklichkeit ein Assistent war, verschwand nach einer Weile und brachte 2 Cola mit zurück. Eine gab er dem anderen, echten, Teilnehmer, die zweite trank er selbst. Im Anschluss an die gestellte Bildbewertung erzählte der Assistent, das er Losverkäufer sei und noch ein paar Lose verkaufen müsse. Es verwundert nicht, das der »Verkäufer« in der Gruppe der er eine Cola spendierte

mehr Umsatz generierte als in der Kontrollgruppe ohne Cola. Weit interessanter als diese Erkenntnis ist aber, das der Losverkäufer den beschenkten nicht wesentlich sympathischer erschien. Für gewöhnlich entscheidet die Sympathie darüber, ob wir einem Unbekannten einen Gefallen tun oder nicht – bei den mit der Cola beschenkten, spielte dies keine Rolle mehr. Die Reziprozitätsregel war so stark, das sie Sympathie oder Antipathie schlichtweg übertünchte.

Vertreter oder Verkäufer setzen diese Masche oft ein, in dem sie Gratisproben verteilen. Ebenso gehen Spendensammler oder Umfrager vor. Sie schicken mit dem Spendenaufruf ein kleines Geschenk mit und können so die erhaltenene Spenden im Schnitt verdoppeln. **james** weisen sogar nach, das es effektiver ist einem Fragebogen einen 5$-Scheck beizulegen, als dem Antworter 50$ zu versprechen.

Die Reziprozitätsregel ist so mächtig, das unbeliebte Personen damit ihre Erfolgschancen wesentlich erhöhen können – auch indem sie *ungebetene* Geschenke machen. Im oben genannten Experiment mit der Cola und dem Losverkäufer hatte niemals das »Opfer« die Wahl. Der Verkäufer brachte die Cola aus freien Stücken mit und bot sie dem anderen Teilnehmer an. Diese nahmen allesamt die Cola an, den schließlich hatte sich der Verkäufer bereits in Kosten gestürzt, um die Cola zu beschaffen. Es wäre da doch sehr unhöflich, das Geschenk abzulehnen. Außerdem brachte er 2 Cola mit – eine für sich selbst und eine für den anderen Teilnehmer. Dies verschleierte seinen »Bestechungsversuch«.

Eine weitere Konsequenz der Reziprozitätsregel ist es, auch Zugeständnisse zu machen. Wenn uns gegenüber jemand in seinen Forderungen Zugeständnisse macht, reagieren wir ebenfalls mit Zugeständnissen – so kommen Kompromisse zustande. Eingesetzt wird dieses Verfahren im Marketing als sogenannte »Neuverhandeln-nach-Zurückweisen-Taktik«. Ein Verkäufer beginnt dazu mit einem überteuerten Angebot. Nach der Ablehnung zieht er sich auf ein billigeres zurück – er macht dem Käufer also ein Zugeständnis.

Neben dem Zugeständnis kommt hier aber noch das *Kontrastprinzip* zum tragen. Danach erscheinen Kontraste größer, als sie unter anderen Umständen erscheinen würden. Ein einfaches Experiment verdeutlicht dies: Man benötigt 3 Eimer mit Wasser, eines eiskalt, eines lauwarm und eines sehr heiß. Hält man nun die linke Hand in das eiskalte und die rechte Hand in das heiße Wasser und dann beide gleichzeitig in das lauwarme, erscheint das lauwarme Wasser einmal als heiß und einmal als kalt – je nachdem, in welchem Wasser die Hand vorher war. Eingesetzt wird das Prinzip in ähnlicher Weise von Verkäufern. Verlang man beispielsweise einen Anzug und ein Hemd zu kaufen, wird immer zuerst der Anzug gezeigt – denn nach einem relativ teuren Anzug, gibt man in der Regel mehr Geld für ein Hemd aus. Für den Verkäufer wäre es Kontraproduktiv, erst das Hemd an den Mann zu bringen – denn ein Anzug würde danach noch teurer erscheinen.

Nun kann man natürlich die Frage stellen, ob eine derartige Masche das »Opfer«

nicht doch verärgern könnte und es eine gegebene Zuzsage nicht einhält. **miller** untersuchte dies in einem Experiment, indem er Studenten um Blutspenden bat. Zuerst bat er sie darum, 3 Jahre lang alle 6 Wochen zu spenden – was natürlich abgelehnt wurde, danach bat er sie um eine einfache Blutspende. Es ist nun nicht mehr überraschen, das aus dieser Gruppe wesentlich mehr Studenten wirklich zur Spende erschienen, als aus der Gruppe, die nicht um eine regelmäßige Spende gebeten wurden. Interessant ist aber auch, das mehr als doppelt so viele Studenten aus der ersten Gruppe ihre Telefonnummer registrieren ließen um auch zukünftig spenden zu können. Die Neuverhandeln-nach-Zurückweisen-Taktik führte also nicht nur dazu, das mehr Studenten erschienen, es waren auch mehr bereit künftig zu spenden. Andere Untersuchungen, wie z. B. die von **schindler** belegen auch, das eine so erzielte Einigung eine höhere Zufriedenheit beim »Opfer« erzeugt.

4.1.1.1 *Abwehrstrategien*

Eine Regel von solch hoher gesellschaftlicher Durchschlagskraft wie sie die Reziprozität hat, lässt sich leicht missbrauchen – und dem Missbrauch ist nur schwer beizukommen. Grundsätzlich alle dargebrachten Gefälligkeiten und Geschenke abzulehnen ist nicht angeraten – es sei denn man möchte vollkommen im sozialen Abseits landen.

Die einzig funktionierende Abwehrmaßnahme ist es, durchaus Geschenke und Gefälligkeiten anzunehmen. Man kann sich dafür später auch entsprechend revanchieren. Man sollte aber strikt darauf achten, ob man übervorteilt wird. Entspricht das, was der Geber an Gefälligkeiten zurückfordert vom Wert her dem gegebenen? Oder versucht er das Gefühl der Verbundenheit auszunutzen und beispielsweise zu einem teuren Kauf oder der Herausgabe eines Passwortes zu verleiten?

Wenn man sich klar wird, das der Gefallen in Wirklichkeit nur ein Manöver war, um Vorteile zu erlangen, verliert die Reziprozitätsregel ihre Durchschlagskraft. Sehen Sie dann den Gefallen nicht mehr als Gefallen, sondern als Bestechungs- oder Betrugsversuch.

4.1.2 *Wer A sagt . . . – von Commitment und Konsistenz*

Den Menschen wohnt ein geradezu zwanghaftes Verhalten inne, in Konsistenz mit ihren früheren Handlungen zu erscheinen – also konsequent zu sein. Wurde eine Entscheidung getroffen, treten intrapsyschische und interpersonelle Vorgänge in Kraft, die uns dazu drängen, konsistent zu bleiben. **fazio** weisen nach, das wir uns einfach selbst davon überzeugen die richtige Entscheidung getroffen zu haben und fühlen uns dadurch wohler mit ihr. In einer Studie haben **knox** Wetter untersucht.

Nachdem die Wetter einen Wetteinsatz auf ein bestimmtes Pferd gesetzt haben, steigt ihre Zuversicht, daß das Pferd gewinnt.

Dieses Verhalten lässt sich auch in den beliebten Heiligen Kriegen beobachten. Die wenigsten Emacs-Benutzer wollen zugeben, das ihre Entscheidung für Emacs falsch war und sie zu Vim konvertieren möchten. Stattdessen verrennen sie sich in ihrer Entscheidung und finden sogar scheinbare Argumente *für* Emacs.

Die Konsistenz ist so stark, das Menschen gegen ihre eigenen Interessen verstossen, nur um nach außen hin als Konsistent zu gelten. **moriarty** führte dazu ein Experiment durch. An einem Strand breitete neben einer zufällig ausgewählten Versuchsperson ein Assistent ein Strandtuch aus und baute ein Radio auf. Nach ein paar Minuten schlenderte der Assisten von dannen und ließ das Radio zurück. Ein weiterer Assistent gab den Taschendieb, griff sich das Radio und zog von dannen. In 20 Durchläufen hat nur eine einzige der Versuchspersonen eingegriffen und versucht den Dieb zu stellen.

Modifizierte man das Experiment stieg die Erfolgsrate dramatisch. Der erste Assistent bat einfach seine Nachbarn, auf das Radio achtzugeben. Nun verfolgten 19 von 20 Versuchspersonen den »Dieb« und stellten ihn teils unter Einsatz körperlicher Gewalt zur Rede.

Auch die Konsistenz hat ihren gesellschaftlichen Nutzen – schließlich können inkonsequente Menschen schlecht zusammenarbeiten. Inkonsistente Personen gelten darüberhinaus auch schnell als verwirrt oder gar geisteskrank.

Ein weiterer, wichtiger Punkt der Konsistenz ist ihre stärkste Form – die Ignoranz. Manchmal handelt man konsequent, weil man Angst hat vor dem Ergebnis einer nüchternen, sachlichen Analyse. Daher hält man an einem einmal eingeschlagenen Weg fest. Oder wie es **cialdini** so schön ausdrückte: »[...] Hinter den dicken Festungsmauern sturer Konsistenz halten wir der Belagerung durch die Vernunft unverrückbar stand.«

Ein wunderbares Beispiel für den kommerziellen Einsatz der Konsistenz führt er auch im 3. Kapitel seines Buches auf. Und zwar leidet die Spielzeugbranche unter saisonalen Schwankungen. In der Vorweihnachtszeit explodiert der Umsatz, dafür bricht er nach Weihnachten um so dramatischer ein. Daher etablierten einige Unternehmen eine interessante Absatzstrategie. Zu Beginn der Vorweihnachtszeit wurde ein neues interessantes Spielzeug eingeführt und massiv und aggressiv beworben. Solch ein Beispiel ist Furby, der 1998 vertrieben wurde. Es handelt sich hierbei um potthässliches, nerviges[2] Plüschtier, das ein paar Sätze sagen kann. Das Produkt wurde zwar

[2] Man verzeihe meinen Hass auf das Vieh, aber ich habe 1998/99 in einem Supermarkt an der Kasse gearbeitet. Und dort standen dann Containerweise diese Dinger rum. Jeder Kunde grabschte sich eines und drückte es, um es reden zu hören. Natürlich redeten die Dinger immer weiter, so das am Ende mehrere Dutzend quakender Furbies vor den Kassen rumlagen. Nicht sehr angenehm, wenn

aggressiv beworben, aber in viel zu geringer Stückzahl auf den Markt gebracht. Das führte dazu, dass viele Eltern zwar einen Furby zu Weihnachten versprachen, aber keinen bekamen. Also kauften sie ein anderes Geschenk und zogen von dannen. Nach den Weihnachtsfeiertagen setzte die Furby-Werbung erneut ein – was die Kinder wieder an ihren alten Weihnachtswunsch erinnerte. Also marschierten sie schnurtracks zu ihren Eltern und verlangten einen Furby. Da diese es ihrem Nachwuchs meist schon vor Weihnachten versprochen hatten, waren sie in der Konsistenzfalle gefangen. Schließlich ist versprochen versprochen, und die eilig auf den Markt geworfenen Furbies gingen weg wie warme Semmeln. Wahrscheinlich um mit ihrem Gequatsche stundenlang arme Eltern zu nerven.

Was ist nun das Auslösemerkmal für eine solche Konsistenzreaktion? Die Sozialpsychologie geht davon aus, das eine Bindung an oder eine Festlegung auf etwas, ein sogenanntes *Commitment*. Jede darauf aufbauende Überzeugungsstrategie arbeitet damit, uns zu einem Commitment zu bringen. Spendensammler von Wohltätigkeitsprganisationen (oder Drückerkolonnen) beginnen ein Telefonat oft mit einer Frage nach dem Befinden. Antwortet der Angerufene, das es ihm gut geht, gibt er schon ein Commitment ab. Im weiteren Verkaufsgespräch wird darauf zurückgegriffen und versucht, die Spendenbereitschaft zu erhöhen, indem von der misslichen Lage der Spendenbegünstigtene berichtet wird.

Der Marktforscher **howard** testete dieses Verfahren in Dallas. Er rief zufällig ausgewählte Einwohner an und fragte sie, ob es in Ordnung wäre, wenn ein Vertreter vorbeikäme. Dieser Vertreter verkaufe Plätzchen, der Erlös käme den Betreibern einer Armenküche zu Gute. Begann er das Telefonat mit der Frage nach dem Befinden und wartete die Antwort ab (die 90% positiv beantworteten), verdoppelte sich die Zustimmung fast von 18% auf 32%. Von den nach ihrem Befinden befragten, kauften dann auch 89% dem Vertreter die Kekse ab – sie verhielten sich also Konsistent.

Viele Vertreter oder Verkäufer beginnen eine »Spirale der Willfährigkeit« in Gang zu setzen. Dazu drängen Sie den Kunden dazu, eine Reihe kleinerer Commitments abzugeben. Die Drückerkolonnen, die Zeitschriftenabos verticken, fangen beispielsweise immer mit der Frage »Sehen Sie sich gerne gute Filme an?«, um das Opfer einzuwickeln. Auch einige Tierschutzorganisationen (die meist mehr an Geld als an Tierschutz interessiert sind) beginnen häufig mit der Frage ob man Tiere möge. Meine Antwort »Ja, am liebsten als Chicken McNuggets!« lässt das Verkaufsgespräch sehr schnell beenden.

Die Taktik mit einer kleinen Bitte zu beginnen, um sich dann zur großen vorzuarbeiten wird in der Sozialpsychologie und im Marketing »Fuß-in-der-Tür-Taktik« genannt. **freedman** führten ein Experiment durch, bei der kalifornische Hausbesitzer gebeten wurden eine riesige Plakatwand mit der Aufschrift »Augen auf im Straßenverkehr!« im Vorgarten aufzustellen. Von der Kontrollgruppe gaben lediglich 17%

man das 8 Stunden am Tag hört.

ihre Einwilligung, von der »vorbereiteten« Gruppe erstaunliche 76%. Diese Gruppe wurde zwei Wochen vor der Frage nach der Plakatwand gebeten, ein 8x8cm großes Schild mit der Aufschrift »Fahren Sie vorsichtig!« sichtbar anzubringen. Nach diesem Ergebniss wandelten Sie den Test etwas ab und führten ihn erneut durch. Anstatt erst ein kleines Schild zu verteilen, bat man die Testgruppe um eine Unterschrift unter die Petition »Kalifornien soll noch schöner werden«. Zwei Wochen danach versuchte man wieder die riesigen Schilder an den Mann zu bringen – und hatte in über der Hälfte der Fälle Erfolg.

Das interessante am zweiten Test war die Zusammenhanglosigkeit der Bitten – die erste, kleinere Bitte hatte nichts direkt mit der zweiten zu tun. Es ging zuerst um die Verschönerung der Landschaft, danach um den Straßenverkehr. Offenbar hatte die Unterschrift unter die Petition dazu geführt, das die Leute ihr Selbstbild änderten. Sie nahmen sich bewusster als guten Staatsbürger wahr und handelten dementsprechend bei der zweiten Bitte nach ihrem neuen Selbstbild.

Dieser Mechanismus ist das gefährliche an der Fuß-in-der-Tür-Taktik. Hat man das Selbstbild einer Person ersteinmal in eine neue Rolle manipuliert, tut die Person nahezu alles um mit dem neuen Selbstbild konsistent zu bleiben.

Allerdings wirken nicht alle Commitments gleich. Es gibt einige Bedingungen die gelten müssen. Es muss aktiv, öffentlich, mit Anstrengung verbunden und freiwillig sein. Erst wenn diese Bedingungen erfüllt sind, zeigt ein Commitment Wirkung.

Ein aktives Commitment ist das Versprechen etwas zu tun. In der umgekehrten Form – *nicht* versprechen etwas *nicht* zu tun – zeigte es keinen großen Einfluss. Eine öffentliches Commitment zeigt mehr Wirkung als ein nichtöffentliches, schließlich wird dieses ja nicht bekannt und kann so das Bild einer Person ändern. Am effizientesten ist eine schriftliche Verpflichtung. Dies ist ein unumstreitbarer materieller Beweis für eine Festlegung. Deshalb empfehlen auch viele Diät- oder Anti-Rauch-Ratgeber die gefassten Vorsätze schriftlich niederzulegen.

Interessant ist bei den Meinungsbekundungen auch, das diejenigen, die sie lesen, annehmen, der wiedergegebene Standpunkt sei der des Schreibers. Selbst wenn das Publikum weiß, das der Schreiber zu der aussage gezwungen wurde, gehen sie davon aus, das der Schreiber zu der vertretenen Meinung steht (**allison**).

Wichtiger noch als die bisher besprochenen Eigenschaften eines Commitments – es muss aktiv, öffentlich und mühevoll sein – es darf kein »Belohnungs-Hintertürchen« geben. Jemand, der ein Commitment nur wegen einer hohen Belohnung abgegeben hat, wird nicht so stark an es gebunden, wie jemand der keine oder nur eine niedrige Belohnung bekommt. Es geht nicht darum, jemandem irgendein Commitment abzuringen, sondern er muss die volle Verantwortung dafür übernehmen. Ein Versuch von **freedman65** beweist diese These. Er verbot mehreren 7 – 9-jährigen Jungs mit einem besonders interessanten Spielzeug zu spielen. Und wer Jungs kennt – insbesondere

7 – 9-jährige – weiß das Verbote etwas nur wesentlich interessanter machen. Eine Gruppe wurde mit einer Drohung dazu gebracht, die andere mit einer Begründung. In beiden Gruppen, jeweils 22 Jungs stark, spielte nur ein einziger verbotenerweise mit dem Spielzeug.

Interessant wurde es aber erst im zweiten Teil. Sechs Wochen nach dem Test, wurden die selben Gruppen wieder in Kontakt mit dem Spielzeug gebracht. Diesmal wurde kein Verbot ausgesprochen, so das 17 Jungs aus der »Drohungsgruppe« sofort zum verbotenen Spielzeug griffen. Da sie nun nicht mehr mit der Bestrafung rechnen mussten, wurde das 6 Wochen vorher ausgesprochene Verbot quasi wirkungslos. In der 2. Gruppe, die mit einer Begründung vom Spielzeug abgehalten worden waren, griffen nur 7 Jungs zum verbotenen Spielzeug.

Die erste Gruppe hatte also sehr schnell erkannt, das der Testleiter nicht mehr da war, um seine Drohung wahr zu machen. Nur wenn die Jungs befürchteten, ertappt und bestraft zu werden, hielten sie sich an das Verbot. Die 2. Gruppe hingegen wurde nicht mit einer Strafandrohung vom Spielzeug ferngehalten, sondern mit einer Begründung. Das erste Testergebnis entsprach dem der 1. Gruppe – nur jeweils ein Junge spielte mit dem verbotenen Spielzeug. Das entscheidende spielte sich im Inneren der Jungs ab – in der 2. Testgruppe gelangten sie zu dem Entschluss, nicht mit dem Spielzeug spielen zu *wollen*.

In weiterführenden Studien kamen **deci**, **higgins** und **lepper** zu dem Ergebnis, das hohe materielle Belohnungen sogar eine bestehende innere Verantwortung reduziert. Somit ist später, wenn die Belohnung nicht mehr gegeben ist, die Bereitschaft eine Handlung auszuführen wesentlich geringer.

Ein im Marketing bzw. im Verkauf eingesetzter Trick des Commitments ist die sogenannte »throwing a low ball«-Taktik. Hierbei wird ein PKW unter dem üblichen Marktwert angeboten. Natürlich erhöht dieses »Schnäppchen« den Absatz – es werden also mehr Kunden angelockt. Wenn diese dann den Vertrag unter Dach und Fach bringen wollen, bemerkt der Verkäufer oder die Finanzierungsbank, das der Betrag aufgrund eines Fehlers zu niedrig ist und man den Verkaufspreis auf den marktüblichen Preis erhöhen muss. Normalerweise würde man davon ausgehen das verärgerte Kunden schwarenweise diese Autohäuser verlassen und nie wieder betreten. Das Gegenteil ist aber der Fall. Sehr viele Kunden akzeptieren diese Masche und kaufen den Wagen trotzdem – da sie eben schon einige Commitments gemacht haben, als sie den »Papierkram« ausfüllten.

In einer amerikanischen Studie von **pallak** wurde gezeigt, das die »throwing a low ball«-Taktik Hausbesitzer zum Energiesparen bringen kann. Zu Beginn des Winters wurde eine Gruppe von Hausbesitzern von einem »Berater« besucht, der ihnen Tips zum Energiesparen gab. Am Ende des Winters wurde der Energieverbrauch mit dem einer Kontrollgruppe verglichen. Es überrascht nicht wirklich zu erfahren, das es keine nennenswerten Einsparungen gab.

In einer dritten Gruppe hingegen, wurde die »throwing a low ball«-Taktik angewendet. Der »Berater« gab diesmal nicht nur die Energiespartips, sondern verkündete ebenfalls, das es eine Art Energiesparwettbewerb geben würde. Besonders engagierte Energiesparer sollten namentlich in der Lokalzeitung genannt werden. Innerhalb eines Monats senkten die so präparierten Familien ihren Verbrauch um $12m^3$ Erdgas.

Nach dem ersten Monat wurden alle Familien der 3. Gruppe angeschrieben. Ihnen wurde mitgeteilt, das die Lokalzeitung einen Rückzieher gemacht hat, und es keine öffentliche »Hitparade der Energiesparer« geben würde.

Die Frage war nun, ob diese 3. Gruppe in ihre alten Verbrauchsmuster zurückfallen würde. Das Gegenteil war sogar der Fall. Im ersten Monat – als noch die Aussicht auf namentliche Nennung bestand – sparten die Familien im Schnitt 12,2%. In den Monaten danach, als die Zeitung schon ihren Rückzieher bekanntgegeben hat, sparten sie satte 15,5%. Sie sparten also nicht nur weiter, sondern steigerten ihr Engagement noch, obwohl die Belohnung wegfiel.

Welche Erklärung gibt es für dieses Verhalten? Die schlüssigste Erklärung ist folgende: Die Aussicht in der Hitparada genannt zu werden, verhinderte das sich die Hausbesitzer voll und ganz mit dem Bild des energiesparenden, umweltbewussten Bürgers identifizierten. Nachdem dieses Hindernis wegfiel, konnten sie sich selbst als energiesparende, umweltbewusste Bürger sehen – und ihre Anstrengungen sogar noch steigern.

4.1.2.1 *Abwehrstrategien*

Konsistentes Verhalten ist, wie auch alle anderen automatischen Reaktionen, von großem Nutzen für uns und die Gesellschaft. Es ermöglicht uns, Situationen schnell und meist effizient einzuschätzen und uns angemessen zu Verhalten. Die Gesellschaft hingegen kann das Verhalten einzelner Personen besser beurteilen und voraussagen, wenn sich diese konsequent verhalten.

Daher fällt es schwer, auf konsistentes Verhalten zu verzichten und jede Entscheidung erneut abzuwägen. Es gibt allerdings Signale, auf die man achten sollte. Zum einen ist es das Bauchgefühl, zum anderen der »Grund des Herzens«.

Meist merken Menschen im Magen oder im Herzen ob sie betrogen oder ausgenutzt werden sollen. Es gibt einige Studien, die belegen, das wir Gefühle Sekundenbruchteile vor unserer verstandesgemäßen Antwort wahrnehmen (**zajonc**).

Wenn Sie also Zweifel an der Ehrlichkeit einer Person haben, stellen Sie sich folgende Frage: »Bei dem was ich jetzt weiß, wie würde ich mich entscheiden?«. Achten Sie dabei auf ihre erste Gefühlsregung (oder Intuition, wenn Sie das so nennen wollen).

Spätestens wenn sich jetzt Zweifel einstellen, sollten sie alle jetzt bekannten Fakten erneut überdenken und auf Basis dieser Informationen die Entscheidung treffen.

4.1.3 *Soziale Bewährtheit*

Eine weitere Möglichkeit psychische und Denk-Arbeit einzusparen ist das Prinzip der sozialen Bewährtheit. Das Verhalten anderer wird als richtig angenommen und gegebenenfalls kopiert bzw. adaptiert. Im Allgemeinen führt dies zu weniger Fehlern. Problematisch wird dieses Verhalten dann, wenn einzelne Auslösemerkmale dazu benutzt werden sollen, Entscheidungen aufgrund der sozialen Bewährtheit zu fällen.

Dies wird sehr häufig in der Werbung eingesetzt, häufig wird ein Produkt als das beliebteste oder das mit den höchsten Zuwachsraten angepriesen. Außerdem setzen viele Firmen – insbesondere die, die elektronischen Schnick-Schnack verkaufen – auf die »Early Adaptor«. Dies ist eine Käufergruppe, männlich, gut verdienend, 20 – 45 Jahre alt, die ein neues Spielzeug zuerst haben muss. Sie stellen in der Regel zwar nur 5% der gesamten Käuferschaft dar, erfüllen aber eine wichtige Vorbildfunktion.

tesser wiesen nach, das soziale Bewährtheit wichtiger wird, wenn wir unsicher sind und/oder die Situation unklar oder mehrdeutig ist. Dabei kann es zu einem problematischen Verhalten kommen, das sogenannte *kollektive Nicht-sehen-Wollen*. Eine Gruppe von ratlosen Menschen sucht bei den anderen Personen nach Hinweisen wie sie sich verhalten soll. Dabei dreht sich die Situation quasi im Kreis und im Endeffekt tut niemand etwas. Leider passiert dies häufig in Extremsituationen wie Unfällen oder Unglücken.

Nach dem es in New York im Jahre 1964 zu einem aufsehenerregendem Mord gekommen war – eine Frau wurde unter den Blicken von 38 Zeugen mehrfach mit einem Messer angegriffen und schließlich erstochen (**ganzberg**) – richtete sich das Augenmerk der Sozialpsychologie auf dieses Phänomen.

Die Psychologieprofessoren Latané und Darley untersuchten den Fall auf das Prinzip der sozialen Bewährtheit und den sogenannten Effekt des kollektiven Nicht-sehen-Wollens. Niemand half dem Mordopfer *weil* es 38 Zeugen gab, nicht *obwohl* es so viele waren. Diese These untersuchten sie und andere Psychologen anschließend in verschiedenen Experimenten (**latane2**; **latane1**; **latane3**; **ross**).

Ausschlaggebend für das Verhalten der Zeugen war ihre eigene Unsicherheit. Diese führt dazu, sich an anderen Mitmenschen zu orientieren. Da aber alle Anwesenden orientierungslos waren, rührte sich keiner. Es lag also soetwas wie ein »psychologischer Deadlock« vor: Jeder wartete darauf, das jemand anderes etwas unternimmt.

clark1; **clark2** und **latane4** wiesen in mehreren Versuchen nach, das die Hilfeleistung der Zeugen von deren Anzahl abhing. War nur ein Zeuge anwesend, wurde in ausnahmslos jedem Fall geholfen. Befand sich das »Versuchskaninchen« aber in Gesellschaft mehrerer desinteressierter Assistenten, sank die Hilfequote dramatisch ab. Der Effekt des kollektiven Nicht-sehen-wollens ist in Gruppen von Fremden am stärksten. Da man möglichst souverän wirken möchte, lassen die Wenigsten offene Anzeichen von Unsicherheit erkennen. Somit wird der oben beschriebene »psychologische Deadlock« verstärkt.

Der wichtigste Hinweis an Opfer eines Notfalls ist: Beseitigen Sie alle Unsicherheiten hinsichtlich Ihres Zustandes! Machen Sie unmissverständlich klar, das Sie nicht etwa betrunken sind, sondern einen Infarkt haben! Greifen Sie sich dazu eine Person aus der Menge der Zeugen heraus und sprechen Sie sie konkret an: »Sie da, der Mann mit der blauen Jacke. Ich glaube ich habe einen Herzinfarkt, bitte rufen Sie einen Notarzt!«. Dies erhöht die Chance auf Hilfe ungemein.

Neben der Unsicherheit gibt es noch ein weiters Merkmal, das die Stärke der sozialen Bewährtheit beeinflusst: Ähnlichkeit. Je ähnlicher uns jemand ist, desto eher kopieren wir sein Verhalten, oder lassen uns zumindest von ihm beeinflussen (**abrams**; **burn**; **schultz**; **hornstein**).

Dies haben natürlich inzwischen auch die Werbeagenturen bemerkt, und so kommt es nun, das immer öfter der »einfache Mann auf der Straße« oder die normale Hausfrau, und nicht dürrgehungerte Klappergestelle mit 20 Schönheits-OPs, Produkte an den Mann bringen wollen.

Das sollten Sie auch beachten, wenn Sie eine Kampagne für Sicherheitsbewusstsein durchführen wollen. So gern Sie auch den grauen Dreiteiler mit Hemd und Krawatte tragen mögen, er ist unangebracht, wenn Sie die Hausmeister und Reinigungskräfte schulen.

Ein in der Psychologie bekanntes Phänomen ist der sogenannte Werther-Effekt. Goethes Buch von 1774 über Werthers gescheiterte Liebe und anschließenden Selbstmord löste zunächst in Deutschland und später in Europa einen Nachahmeeffekt an Suiziden aus. Teilweise trugen die Selbstmörder ein Exemplar des Buches bei sich, oder waren sogar wie Werther gekleidet.

D. P. Phillips untersuchte in den 70er/80er-Jahren den Einfluss der Medien auf Suizid-Verhalten in mehreren Studien (Phillips 1974, 1979, 1980; Phillips und Carstensen 1986, 1988). Seine Ergebnisse belegen, das es eine Art Vorbildfunktion von Suiziden gibt. Nach der Berichterstattung von Suiziden, stieg die Suizidzahl an. Dabei beeinflusste die Art der öffentlich diskutierten Selbsttötungen die »Nachahmertaten«. Tötete sich ein junger weißer Mann, stieg die Suizidrate unter jungen weißen Männern. Beging jemand einen Suizid indem er mit seinem Auto einen Unfall herbeiführte, stieg die Unfallrate proportional an.

4.1.3.1 *Abwehrstrategien*

Abwehrstrategien gegen die soziale Bewährtheit sind schwierig zu erreichen. Schließlich kann man sich nicht dagegen verwahren, Informationen aus dem Verhalten anderer zu ziehen (Hill 1982; Laughlin 1980; Warnick und Sanders 1980).

Daher besteht die beste Abwehrstrategie gegen Manipulationsversuche darin, auf offensichtlich falsche Eingabedaten zu achten.

Viele Manipulateure wissen um die Macht der sozialen Bewährtheit, gehen aber dilettantisch vor. Bestes Beispiel sind die Lachereinspielungen in Fernsehsendungen (z. B. in »Eine schrecklich nette Familie«), die gekünstelter kaum wirken könnten. Ebenso verpfuscht sind Werbespots in der »der kleine Mann auf der Straße« zu Wort kommt. Hölzerner und durchschaubarer als diese Amateur-Normalos wirken eigentlich nur noch deutsche Seifenopern.

Erkennt man derartig primitive Vorgehensweisen, kann man den Autopiloten getrost abschalten und Rachemaßnahmen einleiten. Ignorieren Sie einfach die Werbebotschaft oder sonstige Verführungsmaßnahmen. Überdenken Sie mögliche Entscheidungen und überprüfen Sie, wie rational Sie bisher vorgegangen sind.

Die soziale Bewährtheit kann aber auch zu falschen Entscheidungen führen, obwohl niemand bewusst manipuliert. Im oben bereits vorgestellen Effekt des kollektiven Nicht-sehen-Wollens führt die Passivität einiger Zeugen dazu, das Unfallopfern nicht geholfen wird. Auch auf den Finanzmärkten lässt sich ein Phänomen namens *Herdentrieb* beobachten. Anleger richten ihre Entscheidung am Kaufverhalten anderer Anleger aus. Sie vermuten, dass die anderen Händler über mehr Informationen verfügen und so ein gutes Geschäft abschließen. Diesem wollen sich die »Kälber« in der Herde anschließen und verstärken somit auch noch den Effekt. Derartige Phänomene führten sogar schon fast zu Bankzusammenbrüchen (News 1988).

4.1.4 *Sympathie*

Nicht wirklich überraschend ist die Tatsache, das uns sympathische Menschen eher zu etwas verleiten können. Jeder vernünftige Verkäufer[3] versucht dem Kunden gegenüber besonders sympathisch zu erscheinen. Sympathie verstärkt alle anderen eingesetzten Überzeugungstricks.

Da die Sympathie eine sehr große Rolle spielt, wird sie oft als Waffe eingesetzt – aber auch sehr gut erforscht. Daher gibt es sehr gute Anhaltspunkte dafür, was uns

[3] Berliner Kellner sind *keine* vernünftigen Verkäufer.

jemanden als sympathisch erscheinen lässt.

4.1.4.1 *Attraktivität*

Besonders stark lassen wir uns von besonders attraktiven Menschen beeinflussen. Hier führt der sogenannte *Halo*-Effekt dazu, dass die herausstechende Eigenschaft *Attraktivität* alle anderen Eigenschaften überstrahlt. Testpersonen schrieben einer besonders attraktiven Person automatisch besonders positive Eigenschaften zu (Eagly u. a. 1991).

Verschiedene weitere Studien zeigen, das attraktivere Menschen bei Vorstellungsgesprächen bevorzugt werden (Mack und Rainey 1990), mehr verdienen (Hammermesh und Biddle 1994), von einem Geschworenengericht bevorzugt werden (**downs**; Moore u. a. 1994; Stewart II. 1980) oder eher Wahlen gewinnen (Efran und Patterson 1976). Es ist daher nicht weiter verwunderlich, das Verkäufer und andere Betrüger attraktiv erscheinen wollen.

4.1.4.2 *Ähnlichkeit*

Neben der Attraktivität gibt es noch weitere Faktoren, die uns beeinflussen können. Und diese lassen sich auch wesentlich stärker anpassen als Attraktivität.

Einer dieser Faktoren ist Ähnlichkeit. Sympathie und Hilfsbereitschaft steigen gegenüber Menschen, die uns ähnlich gekleidet sind (Byrne 1971; Emswiller, Deaux und Willits 1971; Suedfeld, Bochner und Matas 1971).

Neben der äußeren Erscheinung können auch die Herkunft oder ähnliche Interessen Sympathie erzeugen.

4.1.4.3 *Komplimente*

4.1.4.4 *Kooperation*

4.1.4.5 *Attraktivität*

4.1.4.6 *Abwehrstrategien*

4.1.5 *Autorität*

4.1.5.1 *Abwehrstrategien*

4.1.6 *Knappheit*

4.1.6.1 *Abwehrstrategien*

4.2 Mitarbeiter authentifizieren

Mitarbeiter korrekt zu authentifizieren ist die wichtigste Maßnahme gegen Social-Engineering-Angriffe. Eine Authentifikation ist die Überprüfung der »Identität« einer Person oder einer Sache. Um eine Person zu authentifizieren, kann man auf 3 Verfahren zurückgreifen:

- Besitz: Die Person besitzt etwas, wie einen Schlüssel oder eine Kennkarte.
- Wissen: Die Person weiß etwas, wie ein Passwort oder eine Geheimzahl.
- Merkmal: Die Person hat ein biometrisches Merkmal, wie ein Irisscan oder Fingerabdruckvergleich ergibt.

Alle drei Verfahren haben ihre eigenen Vor- und Nachteile: Besitz ist ein Gegenstand, und der muß erst hergestellt werden. Außerdem muss er mitgeführt und geschützt werden. Stiehlt ein Angreifer den Besitz, kann er sich damit authentifizieren. Wissen hingegen lässt sich leicht erzeugen (Passwortgeneratoren) und verteilen. Es muss aber gelernt werden, d.h. man muss sich das Passwort merken können. Es ist unwahrscheinlich, das man einen Diebstahl des Wissens bemerkt. Somit kann sich ein

Angreifer Zugriff auf das Wissen verschaffen und sich damit unentdeckt authentifizieren. In der Theorie sind biometrische Merkmale äußerst sicher, praktisch kommt es jedoch immer wieder zu Pannen und Einschränkungen. Die benötigten Geräte sind teuer und fehleranfällig. Außerdem muss das Referenzmuster an jede einzelne Kontrollstation verteilt werden.

Die Sicherheit der Verfahren lässt sich durch kombinieren mehrerer Merkmale erhöhen. So kann man eine Tür mit einem normalen Schlüssel und einer Geheimnummer sichern. Ein Einbrecher benötigt dann beides: den *Besitz* des Schlüssels und das *Wissen* der Geheimnummer.

4.2.1 *Mitarbeiterausweise*

Zur korrekten Authentifizierung aller Mitarbeiter kann es ab einer bestimmten Unternehmensgröße notwendig sein, Mitarbeiterausweise einzuführen.

Deutsche Personalausweise und Reisepässe zählen – auch ohne esoterische RFID-Chips oder »Biometrie« – zu den sichersten Ausweisdokumenten weltweit. In den Jahren 2001 bis 2006 wurden nach (Bundesregierung 2007) lediglich 6 gefälschte und 344 verfälschute deutsche Reispässe bundespolizeilich aufgedeckt. Es ist wesentlich einfacher einen *echten* Personalausweis für eine *falsche* Person zu bekommen, als einen Ausweis zu fälschen. Trotzdem eignen sich beide Dokumente zur Authentifizierung von Personen. Denn ab dem vollendeten 16. Lebensjahr muss jeder Bundesbürger einen Personalausweis besitzen. Man muss ihn aber *nicht* ständig mitführen, wie oft falsch angenommen wird.

Einen Führerschein sollten Sie nicht zur Authentifizierung verwenden. Diese Dokumente sind nicht fälschungssicher.

Einige Staaten, wie die USA oder England, haben keine generelle Ausweispflicht. Wenn Sie Mitarbeiter aus diesen Staaten angestellt haben, verlangen Sie, das diese einen gültigen Reisepass ihres Heimatlandes besitzen.

Ein Personalausweis reicht aus um eine Person zu authentifizieren, nicht jedoch, um eine Person als Mitarbeiter auszuweisen. Hierzu benötigen Sie zwingend eine Art Mitarbeiterausweis. Dieser muss, je nach Sicherheitsanforderung des Unternehmens oder der Abteilung, verschiedenen Anforderungen genügen.

Grundsätzlich benötigt ein funktionierender Ausweis ein aktuelles und großes Farbfoto. Ideal wäre hier das selbe Foto wie im Personalausweis oder Reisepass. Achten Sie auch darauf daß das Foto nicht zu alt ist. Nicht umsonst laufen die ersten beiden Personalausweise nur 5 Jahre und danach jeweils 10 Jahre. Man sieht mit 26 Jahren

nun mal anders aus als mit 16. Wenn der Mitarbeiterausweis offen getragen werden soll, muss das Foto entsprechend groß sein. Die übliche Briefmarkengröße wie im Personalausweis reicht hier nicht aus.

Versehen Sie den Mitarbeiterausweis mit Vor-, Nach- und Geburtsnamen des Trägers, außerdem mit Geburtsdatum und -ort sowie der Personalausweisnummer. Größe und Augenfarbe können ebenfalls übernommen werden. Wird der Ausweis offen getragen, sollten aus datenschutzrechtlichen Gründen alle weiteren Angaben außer dem Namen auf der Rückseite angebracht sein.

Jeder Ausweis muss mit dem Austellungs- *und* Verfallsdatum gekennzeichnet sein. Legen Sie eine sinnvolle Laufzeit für alle Ausweise fest und lassen Sie sie so regelmäßig erneuern. Dies gilt besonders für Ausweise mit hoher Schutzstufe oder für spezielle Abteilungen.

Weitere Angaben, wie Schutzstufe oder Abteilung des Mitarbeiters, sollten ebenfalls auf dem Ausweis enthalten sein. Diese Information kann auch als Hintergrundfarbe des Ausweis hinterlegt sein, so das man Zutritts- und Zugriffsberechtigungen auf den ersten Blick erkennt.

Alle Mitarbeiterausweise benötigen fälschungssichere Elemente. Dies können Hologramme, Silberstreifen und Wasserzeichen sein, wie sie auch auf Geldscheinen oder dem Personalausweis enthalten sind. Ein einfacher Ausdruck mit gescanntem Foto eignet sich definitiv nicht als Ausweis. Es können auch Chipkarten verwendet werden, auf denen eine kryptographische Signatur liegt. Erstellen Sie eine einfache, strukturierte Textdatei mit allen Angaben – inklusive der Personalausweisnummer, die sich auf dem Ausweis befinden. Erzeugen Sie dann eine PGP-Signatur und speichern Sie beide Dateien auf dem Chip ab. Um einen Ausweis zu überprüfen, können Sie den Chip auslesen und die PGP-Signatur überprüfen. Neben der Textdatei können Sie auch ein Foto signieren und auf dem Chip ablegen.

Legen Sie einen Katalog mit allen ausgestellten Ausweisen an. Verzeichnen Sie jeden ausgestellten Ausweis mit allen Daten. Legen Sie außerdem ein Foto des Mitarbeiters ab und seine Personalausweisnummer. Schützen Sie diesen Katalog unbedingt! Er ist eines der lohnendsten Angriffsziele für jeden Social Engineer. Lagern Sie ihn am besten auf einem Rechner ohne Netzwerkverbindung und in verschlüsselter Form. Dem Wachpersonal oder Werkschutz können Sie Ausdrucke der aktuellen Ausweise zur Verfügung stellen. Somit können Ausweise eines Mitarbeiters mit der Liste verglichen werden. Entwerten Sie im Katalog und im Ausdruck auch die Mitarbeiterfotos mit einem »Stempel«, sonst hat ein gewiefter Einbrecher gleich eine Bastelvorlage zur Hand. Pflegen Sie außerdem auch eine Liste aller zurückgezogenen, verlorenen oder sonstwie ungültig gewordenen Ausweise. Halten Sie Ihre Mitarbeiter dazu an, verlorene Mitarbeiter- und/oder Personalausweise anzuzeigen. Informieren Sie darüber sofort alle notwendigen Kontrollstellen.

Stellen Sie hierbei aber sicher, das der meldende Mitarbeiter zweifelsfrei authentifiziert wird. Sonst ruft ein Social Engineer beim Pförtner an und behauptet Dr. Müller-Lüdenscheid hätte seinen Mitarbeiterausweis verloren und der echte Dr. Müller-Lüdenscheid wird nicht mehr hineingelassen.

4.2.2 *Der deutsche Personalausweis*

Der deutsche Personalausweis ist einer der sichersten weltweit. Dies beruht auf dem Einsatz verschiedener, voneinander unabhängiger, Sicherheitsmerkmale.

Diese sind, (entnommen aus (Bundesdruckerei 2007)):

1. Identigram®: Holographisches Portrait (HOLOGRAPHIC SHADOW PICTURE, HSP®) Das Lichtbild der Inhaberin/des Inhabers des Passes oder Personalausweises wird bei Betrachtung der Pass- oder Ausweiskarte unter flacherem Winkel rechts neben dem herkömmlichen Bild ein zweites Mal in holographischer Form sichtbar. Durch das holographische Abbildungsverfahren ergibt sich eine stilisierte Hell-Dunkel-Wiedergabe der Bildinhalte des Ausweisfotos, in die links vier Bundesadlermotive eingearbeitet sind.

2. Identigram®: 3D-Bundesadler Eine dreidimensionale Darstellung des Bundesadlers ist unter definiertem Betrachtungswinkel in roter Farbe erkennbar.

3. Identigram®: Kinematische Bewegungsstrukturen Die über dem herkömmlichen Lichtbild angeordneten Bewegungsstrukturen zeigen als zentrales Element einen von zwölf Sternen umgebenen Bundesadler. Bei Kippbewegung der Pass- oder Ausweiskarte von links nach rechts verwandelt sich das in der Mittelposition sichtbare Adlermotiv über eine Sechseckstruktur in den Buchstaben »D«. Die Sterne werden abwechselnd größer und kleiner. Die Sechsecke oberhalb und unterhalb des Adlermotivs wandern auf und ab. Eine Kette von Sternen am rechten Bildrand geht in ein »D« über.

4. Identigram®: Makroschrift und Mikroschrift Am linken Rand des herkömmlichen Lichtbildes bestehen die kinematischen Bewegungsstrukturen aus einem geschwungenen Makroschriftband mit dem Text »BUNDESREPUBLIK DEUTSCHLAND«, an das sich links mehrere, parallel verlaufende Mikroschriftzeilen mit gleichem Text anschließen.

5. Identigram®: Kontrastumkehr Beim Abkippen der Pass- oder Ausweiskarte erfährt die Fläche des zentralen Adlermotivs eine Kontrastumkehr, so dass der zunächst helle Adler nun dunkel in einer hellen Sechseckfläche erscheint.



Normalansicht



Ansicht unter flachem Winkel

Abbildung 4.1.: Muster eines deutschen Personalausweises. Quelle: (Bundesdruckerei 2007)

6. Identigram®: Holographische Wiedergabe der maschinenlesbaren Zeilen Auch die beiden maschinenlesbaren Zeilen der Pass- oder Ausweiskarte werden zusätzlich holographisch wiedergegeben. Sie liegen jeweils über den herkömmlichen maschinenlesbaren Zeilen.

7. Identigram®: Maschinell prüfbare Struktur Das Identigram® enthält eine maschinell prüfbare Struktur, die neben dem maschinellen Lesen der Ausweise zur Unterstützung der Sichtkontrolle auch eine maschinelle Echtheitsprüfung ermöglicht. Diese Struktur beinhaltet keine personen- oder dokumentenbezogenen Daten.

8. Oberflächenprägung: Die Pass- bzw. Ausweiskarte ist in ein Spezial-Laminat eingebettet. Eine Sicherheitsprägung verleiht dem Dokument am rechten Rand eine reliefartig fühlbare Oberfläche. Die umgestaltete Prägung beinhaltet den Buchstaben »D«, das Bundesadlermotiv und die Schriftzüge »BUNDESREPUBLIK DEUTSCHLAND« sowie »REISEPASS« bzw. »PERSONALAUSWEIS«.

9. Sicherheitsdruck mit mehrfarbigen Guillochen: Guillochen sind Schutzmuster aus feinen, miteinander verschlungenen Linien, wobei sich Strukturen in verschiedenen Farben passgenau zu einem ausgewogenen Gesamtbild ergänzen. Bei Reproduktionen (z. B. als Farbkopie) werden die Linienstrukturen des Originals in punktierte Rasterstrukturen aufgelöst.

10. Laserbeschriftung: Name und Vorname der Inhaberin/des Inhabers des Passes oder Personalausweises werden am rechten Lichtbildrand durch Lasertechnik in das Ausweismaterial eingebracht.

11. Wasserzeichen: Im durchscheinenden Licht ist im Papier der Pass- und Ausweiskarte ein mehrstufiges Wasserzeichen in Form von über die Fläche verteilten stilisierten Adlern zu erkennen.

Unter http://www.bundesdruckerei.de/de/support/download/ident_72.pdf finden Sie ein Faltblatt, in dem die Sicherheitsmerkmale beschrieben sind. Nutzen Sie es zur Schulung Ihrer Mitarbeiter!

4.2.3 *Entfernte Authentifizierung von Mitarbeitern*

Eine besondere Herausforderung ist die Authentifizierung von Mitarbeitern am Telefon.

Eine Möglichkeit ist die Verwendung einer ITAN-Liste. Hier werden mehrere TANs generiert und indiziert, d.h. mit einer laufenden Nummer versehen. Ein Exemplar

wird dem Mitarbeiter übergeben, ein anderes dem Rechenzentrum oder einer anderen Abteilung. Meldet sich nun ein Mitarbeiter von außerhalb, fragt der Mitarbeiter im Rechenzentrum eine, oder besser noch mehrere, der ITANs ab.

In Abbildung 4.2 wird die Generierung von ITANs gezeigt. Auf NetBSD können Sie shuffle(1) verwenden, das eine Eingabeliste permutiert. Verwenden Sie in der Eingabeliste alle Zeichen die Ihre Tastatur hergibt! Neben Shuffle können Sie auch den APG verwenden. Er erzeugt im Beispiel komplett zufallsgenerierte Passwörter. Benötigt aber dazu jeweils einen IV für die Zufallsgenerierung. Daher wird vorher einfach ein find-Lauf gestartet, der etwas Last erzeugt. Die dritte Schleife erzeugt eine ITAN-Liste, in der keine laufende sondern eine zufällige Nummer als Index verwendet wird. Dies kann quasi als Challenge/Response-Authentifikation des Rechenzentrenmitarbeiters durch den anrufenden Mitarbeiter eingesetzt werden. Ein Social Engineer könnte sich ja durchaus als RZ-Mitarbeiter ausgeben und versuchen von einem Mitarbeiter ITANs zu erbeuten. Verwendet man statt laufender Nummern Zufallszahlen, kann der Social Engineer nicht einfach nach den ITANs Nummer 23 und 42 fragen, sondern muss ersteinmal die Indexnummer selbst in Erfahrung bringen.

Grundsätzlich ist die Authentifizierung eines Mitarbeiters über ungeschützte Kanäle immer riskant. Bei Verwendung von normalen Passwörtern, die einen längeren Zeitraum gültig sind, können diese abgehört und von einem Angreifer verwendet werden. Selbst die oben vorgestellten ITAN-Listen sind nicht gegen eine Man-in-the-Middle-Attacke gefeit. Bringt der Angreifer einen Mitarbeiter dazu, ihn anzurufen und für einen Mitarbeiter des Rechenzentrums zu halten, kann er vom Opfer ITANs erfragen. Unter Einsatz von Social-Engineering-Methoden kann er auch mehr als die benötigten 2 ITANs erschleichen – er muss bspw. lediglich die übergebenen ITANs als falsch deklarieren und kann danach vom Mitarbeiter ein weiteres Paar anfordern. Um dieses Risiko zu verkleinern, sollten Challenge/Response-Listen verwendet werden. Diese werden im dritten Beispiel in Abbildung 4.2 erzeugt. Der einzige Unterschied zur normalen ITAN-Liste ist die andere Indizierung. Anstatt einer laufenden Nummer, werden auch hier Zufallszahlen verwendet. Damit kann ein Angreifer, der die Man-in-the-Middle-Attacke fahren möchte, nicht mehr einfach nach bestimmten ITANs fragen, sondern muss selbst erstmal eine der Zufallszahlen in Erfahrung bringen. Natürlich kann der Angreifer als Außendienstler »verkleidet« im Rechenzentrum anrufen und sich nach einer ITAN fragen lassen. Er hat dann eine der Indexnummern erfahren. Er wird aber keine Antwort auf die Frage nach der ITAN wissen und das Gespräch unter einem Vorwand abbrechen. Anschließend kann er versuchen dem echten Mitarbeiter die passende ITAN zu entlocken. Allerdings wird ein sicherheitsbewusster RZ-Mitarbeiter merken das etwas nicht stimmt, schließlich authentifiziert sich der Anrufer nicht korrekt. Anschließend wird er die eben genannte ITAN von der Liste streichen und entwerten – damit nicht ein anderer Kollege diese ITAN verwendet.

```
$ for i in ‘seq 1 5‘;
  do
    printf "$i: "; shuffle -0p6 1 2 3 8 9 a b c x y z - _ :  ;
    echo;
  done;
  1: a953:6
  2: 725acb
  3: c46_de
  4: cb6ae7
  5: 64c38-
$ find /tmp/ 1>/dev/null &
$ for i in ‘seq 1 5‘;
  do
    printf "$i: ";
apg -n 1 -c ‘shuffle -0p6 1 2 3 4 5 6 7 8 9 10 11 12 ‘ -M SNCL;
  done
  1: Don9Glolg<
  2: Eraw6quok<
  3: gars?Twug9
  4: +Groljonn4
  5: 1?quiSiph
$ for i in ‘seq 1 5‘;
  do
    printf "‘shuffle -0p4 0 1 2 3 4 5 6 7 8 9‘: ";
    shuffle -0p6 1 2 3 4 5 6 7 8 9 a b c x y z  ;
    echo;
  done | sort
  2968: 3zba12
  3679: 673ba2
  5082: 89cb65
  6483: 5x27c6
  7498: 9xzb62
$
```

Abbildung 4.2.: Erzeugung von ITANs

- Halten Sie die ITAN-Liste geheim!
- Halten Sie den Algorithmus bzw. das ITAN-Schema geheim!
- Eine angesagte Indexnummer ist zu streichen! Insbesondere wenn der Anrufer versucht das Gespräch zu beenden.
- Wählen Sie *immer* eine zufällige ITAN aus! Lassen Sie sich *niemals* dazu überreden, eine verbrauchte oder bestimmte ITAN zu verwenden!
- Begrenzen Sie die Haltbarkeit der ITAN-Liste. Eine Liste pro Monat ist eine gute Dauer. Entwerten Sie eine ITAN-Liste wenn nur noch 20 der 100 ITANs gültig sind. Je weniger ITANs noch existieren, desto einfacher kann ein paar erraten oder ergaunert werden.
- ITAN-Listen müssen sicher vernichtet werden. Verpflichten Sie Ihre Mitarbeiter diese nach Ablauf der Haltbarkeit *sicher* zu vernichten oder ziehen Sie die Ausdrucke ein. Sofern Ihre Mitarbeiter regelmäßig in einer Filiale sind, können sie die alte gegen eine neue ITAN-Liste austauschen lassen.

5

GRUNDLAGEN DER VERSCHLÜSSELUNGSTECHNIK

Die Natur macht keine Sprünge.

(Gottfried Wilhelm Leibniz)

5.1 WARUM SOLLTE MAN KRYPTOGRAPHIE EINSETZEN?

Kryptographie ist z. Zt. das wohl einzige Mittel, um zuverlässig die Privatsphäre zu schützen oder die Integrität von Daten sicherzustellen. Jede Email oder Datei, die man verschickt, wird im Klartext versandt und ist somit – wie eine Postkarte – lesbar. Die Administratoren aller Mailserver oder Gateways, die die Email passiert, können diese lesen oder noch schlimmer – manipulieren. Es ist für den Empfänger sogar unmöglich nachzuweisen, daß die eMail auf dem Wege manipuliert wurde.

> *Das Briefgeheimnis sowie das Post- und Fernmeldegeheimnis sind unverletzlich.*
> Grundgesetz, Artikel 10, Abs 1.

Trotzdem versuchen insbesondere Politiker, die nicht den geringsten mathematischen oder gar technischen Sachverstand haben, kryptographische Methoden verbieten zu lassen, um somit der Überwachung der Bevölkerung weiter Vorschub zu leisten.

Kryptographische Methoden sind des weiteren auch hervorragende Mittel, um Rechnersysteme vor Manipulation und Einbruch zu schützen bzw. diesen nachzuweisen. Man kann mittels kryptographischer Methoden – wie z. B. der kryptographischen Signatur – die Integrität von Dateien prüfen und so Manipulationen an diesen ausschließen oder gar das Ausführen manipulierter Binaries verhindern – und damit das Einspielen von Rootkits unmöglich machen.

5.2 PRÜFSUMMEN

Prüfsummen sind ein Verfahren, in der über eine Basiszahl, die die Information wiedergibt, nach einer bestimmten mathematischen Vorschrift eine Prüfziffer gebildet wird.

Dies diente ursprünglich dazu, die Übertragung von Daten in Netzwerken sicherzustellen, da der Empfänger nach Erhalt der Daten und der Prüfziffer erneut die Prüfziffer berechnete und mit der Erhaltenen verglich. Stimmten beide überein, verlief die Übertragung problemlos.

Prüfsummen begegnen uns in der heutigen Welt fast ständig, denn sie werden beispielsweise in der ISBN, EAN13 oder als Personalausweisnummer eingesetzt.

Als einfaches Beispiel möchte ich hier die EAN13 beschreiben. Die EAN ist die Europäische Artikelnummmer,/ also die Zahl, die in der Regel unter dem Strichcode steht, der über Scannerkassen gezogen wird.

Um den häufigsten Eingabefehler – nämlich Zahlendreher – zu verhindern, werden die ersten 12 Ziffern, die die Basisinformationen enthalten, gewichtet und addiert:

$$a + 3b + c + 3d + e + 3f + g + 3h + i + 3j + k + 3l = I$$

Man erhält nun daraus die Summe I, deren Differenz zum nächsten Vielfachen von 10 berechnet und als Prüfziffer m angefügt wird.

Ein einfaches Beispiel an einer echten EAN13:

$$EAN = 4025839396595$$

$$4 + 3 * 0 + 2 + 3 * 5 + 8 + 3 * 3 + 9 + 3 * 3 + 9 + 3 * 6 + 5 + 3 * 9 = 115$$

$$120 - 115 = m = 5$$

Die Kassensysteme berechnen bei der Eingabe einer EAN nun diese Prüfziffer erneut und können so falsche Eingaben aufdecken.

5.2.1 *Das MD5-Prüfsummenverfahren*

MD5 ist in (Rivest 1992) definiert und erzeugt nach einem mathematisch recht anspruchsvollen Verfahren 128-Bit-Prüfsummen. Der Algorithmus ist als sogenannte Falltürfunktion konzipiert, d. h. man kann zwar vom Datum aus eine Prüfsumme

erzeugen, zu einer gegebenen Prüfsumme das entsprechende Datum aber nicht konstruieren.

Dies prädestiniert MD5 unter anderem auch als Passworthash, indem man beim Anlegen eines Passworts mittels passwd(1) das übergebene Passwort mit MD5 in eine Prüfsumme überführt und diese Prüfsumme abspeichert. Selbige Prozedur wird angewandt so man sich einloggen will, und es wird der neu erzeugte Hash mit dem Abgespeicherten verglichen und bei entsprechender Übereinstimmung der Hashs das Login durchgeführt. Der Vorteil bei diesem Verfahren ist, daß das Passwort nicht im Klartext auf der Festplatte liegt. Sollte also ein Einbrecher die Passwortdatei `/etc/master.passwd` erbeuten, verfügt er nicht über die Passwörter der Benutzer, sondern lediglich über die entsprechenden Hashs.

Auch wenn es (zum jetzigen Zeitpunkt) keinen Weg gibt, aus den erbeuteten Hashs die Passwörter zu rekonstruieren, ist es aber ohne weiteres möglich, eine Wörterbuchattacke zu fahren, indem man einfach ein Wörterbuch (bspw. `/usr/share/dict/words`) an `md5` verfüttert und die so erzeugten Hashs mit den erbeuteten aus der Passwortdatei vergleicht.

Eine weitere klassische Anwendung ist der Einsatz von MD5 als Prüfsumme insbesondere auf FTP-Servern. So wird in der Regel neben der eigentlichen Datei noch eine Datei mit Prüfsummen zur Verfügung gestellt, die eine Überprüfung der Datei nach dem Download ermöglicht.

5.2.2 *Verified Executables*

Ähnlich wie AIDE, allerdings proaktiv, funktioniert Verified Executables.

Verified Executables implementiert einen Prüfsummenvergleich in den `exec()`-Pfad des Kernels. Dazu wird ebenfalls eine initiale Datenbank (bspw. mit `/usr/share/examples/veriexecctl/gen_sha1`) erzeugt und im Bootprozess via `veriexecctl(8)` in den Kernelspeicher geladen. Ruft man nun ein Binary auf, wird die Prüfsumme erneut gebildet und mit der Datenbank verglichen. Im Securitylevel 1 warnt der Kernel bei abweichender Prüfsumme vor Ausführung, im Level 2 wird die Ausführung des Binaries verweigert.

5.3 symmetrische verfahren

Symmetrische Verschlüsselungsverfahren verwenden sowohl zum Verschlüsseln als auch zum Entschlüsseln denselben Schlüssel. Dies bedeutet im Falle eines Nachrichtenaustauschs, das neben dem Chiffre, das die Nachricht enthält, auch der entsprechende Schlüssel ausgetauscht werden muss.

Dies bedeutet im Falle einer einfachen eMail zwischen Alice und Bob, die bspw. mit mcrypt(1) (siehe Kapitel **??**) und dem Passwort GEHEIM verschlüsselt wurde, daß dieses Passwort über einen sicheren Kanal ausgetauscht werden muss. In der Praxis bedeutet dies, daß das unbefugte Abhören und Verändern des Schlüssels verhindert werden muss - was quasi unmöglich ist.

Ein weiteres Manko der symmetrischen Verfahren ist die Anzahl benötigter Schlüssel in der Kommunikation größerer Gruppen: Man benötigt dort nämlich $n * (n-1)/2$ Schlüssel, was bei 10 Personen exakt 45 Schlüssel sind.

Sinnvoll ist die symmetrische Kryptographie jedoch, wenn man die Schlüssel nicht über unsichere Kanäle austauschen will, wenn man z. B. das eigene Dateisystem oder Backups lokal verschlüsseln möchte.

5.3.1 *mcrypt(1)*

Mcrypt umfasst ein Nutzerprogramm und eine entsprechende Bibliothek, die mehrere symmetrische Kryptoverfahren zur Verfügung stellt. Mcrypt lässt sich recht einfach bedienen und bietet eine Vielzahl an Algorithmen und Modi. Die mitgelieferte Bibliothek `libmcrypt` bietet Entwicklerzugriff auf eine Vielzahl symmetrischer Kryptoalgorithmen von DES bis AES.

5.4 asymmetrische verfahren

Ein asymmetrisches Verfahren wird durch die Verwendung eines Schlüsselpaares, bestehend aus öffentlichem und privaten Schlüssel, gekennzeichnet. Zur Erzeugung des Schlüsselpaares gibt es verschiedene Möglichkeiten, die verbreitetsten sind RSA und DSA, welche auf der Faktorisiserung des Produkts zweier Primzahlen basieren oder ElGamal bzw. Diffie/Hellman, die zwei große Zahlen potenzieren und den diskreten Logarithmus dieser Zahl verwenden.

Die so erzeugten Schlüssel ermöglichen eine komfortable asymmetrische Kommu-

nikation, da ein Beteiligter nur den öffentlichen Schlüssel des jeweiligen Kommunikationspartners benötigt und dieser bedenkenlos über unsichere Kanäle verschickt werden kann.

Prinzipell ist die Verwendung asymmetrischer Kryptoverfahren recht einfach:
Alice und Bob haben je einen öffentlichen $(A_{ö}, B_{ö})$ und einen privaten (A_p, B_p) Schlüssel erstellt und möchten miteinander verschlüsselt kommunizieren:

$$Chiffrierung(Klartext + A_p + B_{ö}) \rightsquigarrow Code$$

Alice verwendet ihren eigenen privaten Schlüssel und den öffentlichen Schlüssel von Bob um einen kryptographischen Algorithmus über den Daten anzuwenden. Nachdem sie das Chiffre, zusammen mit ihrem öffentlichen Schlüssel, an Bob geschickt hat, kann dieser das Chiffre mit seinem privaten und Alice öffentlichen Schlüssel wieder in Klartext wandeln:

$$Dechiffrierung(Code + B_p + A_{ö}) \rightsquigarrow Klartext$$

5.4.1 *Ein asymmetrisches Verfahren genauer betrachtet: RSA*

Durch das entwickelte Prinzip des öffentlichen und privaten Schlüssels ist es möglich geworden verschlüsselte Daten und öffentlichen Schlüssel über ungesicherte Kanäle auszutauschen und so starke kryptographische Verfahren quasi für Jederman zugänglich zu machen. RSA wird unter anderem in OpenSSH oder GnuPG verwendet und ist offiziell in RFC3447[1] definiert.

Grundlage des Verfahrens ist das Erzeugen eines Schlüsselpaares welches dann zur Verschlüsselung der Daten verwendet wird. Erzeugt werden die Schlüssel in dem die zufälligen Primzahlen p und q generiert werden. Aus diesen Primzahlen wird die Modulzahl $modulus$ gemäß Vorschrift $modulus = p * q$ sowie ϕ mit $\phi = (p-1) * (q-1)$ erzeugt. Danach generiert man eine große (empfohlen werden vom US DoD mindestens 1024 Bit) Primzahl e welche als privater Schlüssel genutzt wird und daher geheim zu halten ist. Abschließend wird der öffentliche Schlüssel d mittels Auflösen der Gleichung $(e * d) mod\ \phi = 1$ nach d erzeugt. Da mit der Kenntnis von p und q die Zahlen $modulus$ und ϕ berechnet werden können ist es somit auch möglich den privaten Schlüssel zu generieren, daher ist es notwendig diese Zahlen entsprechend zu schützen bzw. nach Erzeugung des Systems zu vernichten.
Die Verschlüsselung eines Textes erfolgt mittels

$$Chiffre = (Klartext^{\ öffentlicher\ Schlüssel}) mod\ modulus$$

Die Entschlüsselung vollzieht sich also folgendermaßen:

$$entschlüsselt = (Chiffre^{\ privater\ Schlüssel}) mod\ modulus$$

[1] ftp://ftp.rfc-editor.org/in-notes/rfc3447.txt

Die Implementierung des RSA Algorithmus ermöglicht es eine gewünschte Bitlänge zur Erzeugung von p und q zu übergeben. Dies ist wünschenswert, da nur eine entsprechend hohe Bitlänge (mindestens 1024 Bit, besser noch 2048) die Sicherheit des Schlüsselsystems ermöglicht, denn der Aufwand zur Zerlegung von ϕ in die entsprechenden Primfaktoren ist zu hoch.

5.5 kryptographische signaturen

Die kryptographischen Signaturen bilden einen Sonderfall der asymmetrischen Verfahren. Mit ihnen ist es möglich neben der Integrität der signierten Daten auch die Integrität und Authentizität der Prüfsumme sicherzustellen. Hierzu wird vom Absender eine Prüfsumme erzeugt und mit dem eigenen privaten Schlüssel verschlüsselt, so daß das Chiffre mithilfe des öffentlichen Schlüssels entschlüsselt werden kann.

Schematisch funktioniert eine Signatur, die von Alice durchgeführt wird so:
Alice bildet eine Prüfsumme über ihre Nachricht und erhält:

$$h(N)$$

Dieser Hash entspricht der in Kapitel 5.2 besprochenen einfachen Prüfsumme. Um diese nun zu schützen, verschlüsselt Alice ihre Prüfsumme mit ihrem privaten Schlüssel A_p:

$$S = \{h(N)\}_{A_p}$$

Nun kann ein Empfänger der Nachricht, beispielsweise Bob, mit Hilfe des öffentlichen Schlüssels von Alice die Signatur und die damit signierten Daten überprüfen. Hierzu bildet Bob zuerst selbst eine Prüfsumme über die Nachricht ($h'(N)$) und entschlüsselt anschließend die Signatur S mit

$$A_{\ddot{o}} \rightsquigarrow h(N)$$

und erhält somit die von Alice gebildete Prüfsumme $h(N)$. Nun ist lediglich zu prüfen ob $h'(N) = h(N)$ gilt. Wenn dem so ist, sind Nachricht und Signatur korrekt.

Als Beispiel ein von mir mit GnuPG mit `0xB3FBAE33` signierter Klartext:
Der Block wird mit »`-----BEGIN PGP SIGNED MESSAGE-----`« ein- und »`-----END PGP SIGNATURE-----`« ausgeleitet. `Hash: SHA1` kennzeichnet das verwendete Prüfsummenverfahren als SHA1. Der Block beginnend mit `iD8D...` ist die Signatur und enthält auch die entsprechenden Informationen zu meinem Schlüssel.

Jeder Text, der vor der Ein- bzw. nach der Ausleitung steht, wurde nicht signiert und sollte demzufolge auch als nicht vertrauenswürdig betrachtet werden!

[!h]

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Von Zeit zu Zeit seh ich den Alten gern,
Und hüte mich, mit ihm zu brechen.
Es ist gar hübsch von einem großen Herrn,
So mit dem Teufel selbst zu sprechen.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.4 (NetBSD)

iD8DBQFAA8ySEfTEHrP7rjMRAmLxAJ0ePbnStfzK4NVSZFMSpvhhIp2NqACdEuFA
NChplstKID03nVEDEDdOUmA=
=+lsL
-----END PGP SIGNATURE-----
```

Listing 5.1: Beispiel einer PGP-Signatur

5.5.1 *Schlüsselsignatur*

Auf verschiedenen Veranstaltungen, wie bspw. den Chemnitzer LinuxTagen oder dem Chaos Communication Congress, werden sogenannte Keysigningparties durchgeführt. Nun kann man sich verständlicherweise Fragen warum man einen PGP Schlüssel signieren sollte, da dieser doch offensichtlich keine Nachricht enthält. Dies ist aber so nicht korrekt, denn immerhin enthält der Schlüssel die Informationen über seinen Besitzer wie Name, Kommentar, eMailadresse, Verfallsdatum etc. und kann manipuliert bzw. gefälscht werden.

Der große Vorteil der asymmetrischen Verfahren ist die Möglichkeit der Kommunikation über unsichere Kanäle, d. h. es ist möglich, mit Jemandem den man nicht persönlich kennt verschlüsselt zu kommunizieren. Dazu ist es notwendig den öffentlichen Schlüssel des Kommunikationspartners zu importieren und zu verwenden.

Doch wie kann ich in diesem Falle sicher sein, das der importierte Schlüssel auch wirklich dem gewünschten Partner gehört? Eigentlich gar nicht, es sei denn man verwendet eben die Signatur einer anderen Person oder Authorität als Bestätigung der korrekten Identität eines Schlüssels. Dies heißt in der Praxis eines PGP Keysignings zwischen Bob und Charlie nichts weiter, als das Bob die korrekte Identität Charlies prüft, bspw. anhand des Bundespersonalausweises, Charlie anschließend seinen Schlüsselfingerabdruck als Ausdruck an Bob übergibt und Bob nun den öffentlichen Schlüssel mit seinem Privaten signiert. Der signierte Schlüssel wird von Bob an Charlie verschlüsselt per eMail verschickt. Charlie muß nun sowohl Zugriff auf den privaten Schlüssel als auch auf den genannten eMailaccount haben.

Betrachtet man nun die Signaturen eines öffentlichen Schlüssels, kann man dessen Vertrauenswürdigkeit festlegen. Hierbei ist es dem Nutzer selbst überlassen inwiefern er den Signaturen vertraut, man sollte trotzdem davon ausgehen können, das eine Anzahl von Signaturen vertrauenswürdig ist.

Die in PGP eingesetzte Methode der gegenseitigen Signaturen wird gemeinhin auch als Web of Trust bezeichnet, da die graphische Darstellung der Signaturen einem einfachen gerichteten Graphen entspricht, hierbei geben die inzidenten Kanten eines Knoten die erhaltenen und die adjazenten Kanten die gegebenen Signaturen wieder. Es existieren Programme, wie gpg2dot.pl, um das Web als Graphen zu zeichnen. Ein Beispiel findet sich unter http://www.uni-magdeburg.de/steschum/local-web.jpg in einer ca. 1,3 MB grossen JPEG-Datei, die mein persönlichers Web of Trust darstellt.

Eine andere Form der Signaturverteilung ist der Einsatz von Zertifizierungsinstanzen (auch CA - Certificate Authorities) in einer Hierarchie. Hierbei wird eine oberste CA festgelegt, die weitere CAs siginiert, welche wiederum Endnutzer zertifizieren. Hierbei bildet sich kein beliebiger Graph, sondern ein Baum. Da jeder Baum genau eine Wurzel und nur einen Weg von dieser zu einem Blatt hat, wirkt sich ein Vertrauensbruch in einer CA auf alle Knoten des entsprechenden Unterbaumes aus.

Teil II

DAS BASISSYSTEM HÄRTEN

6

DAS BASISSYSTEM HÄRTEN

Der Computer rechnet mit allem, nur nicht mit seinem Besitzer.

(Dieter Hildebrandt)

6.1 HARDWARE TESTEN

Manchmal kann es vorkommen das ein Rechner nicht-reproduzierbare Fehler erzeugt oder unerklärlicherweise abstürzt[1]. In solchen Situationen ist es notwendig die Fehlerquelle aufzuspüren.

Um die Hardware zu untersuchen gibt es verschiedene Möglichkeiten bzw. Programme. Besonders anfällige Bauteile sind der Arbeitsspeicher und der Hauptprozessor. Beide können bei Defekten unerklärliche Fehler und Abstürze verursachen. Um beide zu testen gibt es eine einfache Möglichkeit, die in Bsp. 6.1 dargestellt ist. Das Shellskript erzeugt aus `/dev/urandom` eine Datei der Größe $2 * \$groesse$, berechnet die passende SHA1-Prüfsumme der Datei, verschlüsselt, löscht und entschlüsselt die Datei wieder und erzeugt wieder eine SHA1-Prüfsumme. Das ganze wird `$durchlaeufe`-mal durchgeführt, ist die Schleife beendet, werden die erzeugten Prüfsummen verglichen. Sind die Prüfsummen nicht gleich wird »Errors found!« ausgegeben, dann können Sie davon ausgehen das RAM und/oder CPU ein Problem haben.

Ein weiteres geeignetes Testprogramm ist Prime95, welches Mersenne-Primzahlen berechnet. Eigentlich ist es ein Projekt zum verteilten Rechnen, das sehr große Primzahlen finden will. Das Programm kann aber neben einer Leistungsmessung auch einen Belastungstest durchführen, indem Primzahlen berechnet und mit einer eingebauten Tabelle verglichen werden. Prime95 verfügt über verschiedene Test-Modi in denen CPU und RAM unterschiedlich belastet werden. Damit kann man mögliche

[1] Abstürze in NetBSD sind äußerst selten, in stabilen Versionen nahezu ausgeschlossen. Wenn gehäufte Abstürze auftreten, liegt es meist an der Hardware.

Fehler genauer einzukreisen. Auf der Projektseite[2] gibt es das Programm als statisches FreeBSD-Programm zum herunterladen, auf einem NetBSD mit FreeBSD-Emulation (ist in jedem GENERIC-Kernel aktiviert) läuft das Programm problemlos.

[!h]

```
#!/bin/ksh
# Anzahl der Runden
typeset -i durchlaeufe=5

# Größe der zu erzeugenden Datei, wird verdoppelt, da bs=2m
# sollte Größer als RAM sein
typeset -i groesse=3

typeset -i cnt=1
while [[ $cnt -le $durchlaeufe ]]
do
dd if=/dev/urandom of=temp.wipe bs=2m count=$groesse 2>/dev/null
sha1 temp.wipe » temp.sha1.1
openssl des3 -in temp.wipe -out temp.wipe.enc -e -salt -k passwd
rm -Prf temp.wipe
openssl des3 -d -in temp.wipe.enc -out temp.wipe -k passwd
rm -Prf temp.wipe.nc
sha1 temp.wipe » temp.sha1.2
cnt=cnt+1
done
diff temp.sha1.1 temp.sha1.2
[ $? -eq 0 ] && { echo "No Error"; } || echo "Errors found!";
rm temp.wipe.enc
rm temp.wipe
```

Listing 6.1: lasterzeuger.sh, ein Skript um CPU und RAM zu testen

In Pkgsrc gibt es noch das Paket memtest86, das eine Bootdiskette bereitstellt, die ein System startet, welches den Arbeitsspeicher testet. Dieser Test ist sehr zeitaufwändig, testet aber jede Speicherzelle.

Weitere Fehlerquellen können das Netzteil und Strom- bzw. Spannungschwankungen sein. Diese lassen sich nicht mit einfachen Werkzeugen überprüfen sondern erfordern spezielle Geräte.

6.2 Das System herunterfahren, neu starten oder in den Single-User-Modus bringen

NetBSD verfügt als BSD-Abkömmling auch über den sogenannten Single-User-Modus. In diesem Modus wird das System ohne Netzwerk- und Terminal-Unterstützung

[2] http://www.mersenne.org/freesoft.htm

hochgefahren, d. h. der Zugriff ist nur über die Konsole möglich. Dieser Zustand ist sehr nützlich wenn Wartungsarbeiten durchgeführt werden müssen, ohne das dabei eingeloggte Benutzer oder Netzwerkdienste stören. Erreicht werden kann er, in dem Bootloader beim herunterzählen des Countdowns mit [ESC] abgebrochen und das System mit `boot -s` gestartet wird.

Um Logins zu verbieten und Benutzer aus dem System rauszuschmeißen bietet shutdown(8) die Option `-k` für »Kick Off«. Alle Benutzersitzungen werden beendet und weitere Logins, außer für Root, verboten.

Um ein System neu zu starten, verwendet man shutdown(8). Es wertet /etc/rc.shutdown aus und fährt somit laufende Dienste wie bspw. PostgreSQL sauber herunter. Weiterhin warnt es alle eingeloggten Benutzer vor dem Neustart bzw. Herunterfahren und verhindert ab einem bestimmten Zeitpunkt Neuloggins. Die ebenfalls existierenden Programme halt(8) und reboot(8) werden zwar von shutdown(8) aufgerufen, sollten aber niemals alleine ausgeführt werden, da sie keinerlei Aufräumarbeiten vor dem Herunterfahren ausführen. Als Optionen kann Shutdown verschiedene Möglichkeiten übernehmen, zum einen sind das `-h` und `-p`, wobei die erstere Option das System anhält und die zweite das System auch ausschaltet, vorausgesetzt die Hardware unterstützt dies. Mit der Option `-r` wird das System neu gestartet, `-d` erzeugt einen System-Dump beim Herunterfahren und `-f` verwendet fastboot(8) um beim nächsten Start keine Dateisystemüberprüfung durchzuführen. Als vorletzte Option übernimmt Shutdown einen Zeitpunkt, an dem es die Operation ausführen soll, möglich sind bspw. »now« für sofort oder »08:45« für Dreiviertel Neun. Als letzte Option kann man noch eine Nachricht übergeben, die allen eingeloggten Benutzern per wall(1) zugestellt und in das Syslog geschrieben wird. Übergibt man den Bindestrich »-«, kann eine Nachricht via Stdin übergeben werden. Man sollte hier den Grund für das Herunterfahren oder den Neustart angeben und mitteilen wann das System wieder verfügbar ist. Beispiel 6.2 zeigt verschiedene Optionen, zuerst wird das System sofort heruntergefahren und ausgeschaltet, danach werden um Viertel Neun alle Logins beendet und neue verhindert, dazu wird die angegebene Nachricht an alle Benutzer gemeldet. Zuletzt wird das System sofort neu gestartet.

Shutdown kann von Root oder von jedem Mitglied der Gruppe »operator« durchgeführt werden. Weiterhin ist es möglich, Shutdown über den Ein-/Aus-Taster eines ACPI-fähigen Rechners auszuführen. Dazu muss der Kernel allerdings ACPI und den Ein-/Aus-Taster unterstützen. Ist dies der Fall, kann jeder mit physikalischem Zugriff auf den Taster das System herunterfahren.

[!h]

```
$ shutdown -p now

$ shutdown -k 08:15 "Systemaktualisierung notwendig, \
                     gegen 09:00 sollte das System wieder da sein"

$ shutdown -r now "Systemaktualisierung abgeschlossen, \
starte neuen Kernel"
```

Listing 6.2: Verschiedene Shutdown-Beispiele

6.3 ERSTE SCHRITTE

Um ein System abzusichern, gibt es einige Schritte, die sich schon auf das Basissystem beziehen.

6.3.1 *Schlankes System*

Jeder Ingenieur weiß, daß komplexere Systeme schwerer zu steuern sind. Diese Regel gilt selbstverständlich auch für Betriebssysteme, daher ist es sinnvoll, Betriebssysteme so schlank wie möglich zu gestalten.

Im allgemeinen heißt dies, daß man nicht benötigte Programme nicht installiert und laufen lässt. NetBSD folgt dieser Philosophie und installiert ein äußerst schlankes Basissystem. Dieses System kann man dann den eigenen Bedürfnissen anpassen und erweitern. Sets die man nicht benötigt, bspw. »X« oder »games«, werden bei der Installation abgewählt.

Dienste werden über den rc.d-Mechanismus in der /etc/rc.conf (nicht) gestartet. Dort kann man entsprechende Konfigurationsparameter übergeben und Dienste aufrufen.

Ein schlankes System erfordert auch einen schlanken Kernel. Dabei sollte man insbesondere »Loadable Kernel Modules« (LKM) deaktivieren. LKMs sind Module, die zur Laufzeit des Kernels zugeladen werden können. Dies erhöht die Flexibilität des Kernels, in dem bspw. Treiber nachgeladen werden können. Allerdings gibt es keinerlei Schutz der Kernel-Speicherstruktur vor einem LKM, so daß ein Modul nicht nur das System abstürzen lassen kann, sondern im Endeffekt das gesamte System übernehmen kann. LKMs werden in der Kernelkonfiguration mit
`options LKM# loadable kernel modules`

(de)aktiviert.

Ebenso entfernt man aus der Kernelkonfiguration nicht benötigte Treiber, bspw. das SCSI-System oder Audiogeräte. Denn was nicht im Kernel ist kann auch nicht kaputt gehen. Das Programm [adjustkernel] erzeugt eine Kernelkonfiguration, die nur benötigte Hardware der Maschine einbindet. Diese Konfiguration können Sie als Grundlage für einen eigenen Kernel verwenden.

6.3.2 *Das System aktuell halten*

Selbstverständlich müssen Sie ihr System auch aktuell halten.

Das NetBSD-Projekt hat verschiedene Mechanismen etabliert um Sicherheitslücken im System und in Anwendungsprogrammen zu behandeln.

Werden Sicherheitslücken im Basissystem aufgedeckt, wird eine Email mit einem »Security Advisory« an die `netbsd-announce`-Mailingliste gesendet. In dem »Security Advisory« wird die Sicherheitslücke beschrieben, außerdem wird aufgeführt welche Systeme betroffen sind und wie die Lücke geschlossen werden kann. Abonnieren Sie die `netbsd-announce`-Mailingliste und lesen Sie sie regelmäßig! Aktualisieren Sie ihr System so schnell es geht, denn spätestens mit der Warnmeldung wissen auch Einbrecher von der Sicherheitslücke.

Um Sicherheitslücken in Anwendungsprogrammen zu finden, können Sie [audit-packages] verwenden. Nachdem es installiert wurde, können Sie mit download-vulnerability-list(8) eine Textdatei mit Informationen zu unsicheren Programmen herunterladen, mit audit-packages(8) können Sie automatisch Ihre installierten Pakete überprüfen lassen. Findet audit-packages(8) ein unsicheres Programm, gibt es eine Warnmeldung aus. Aktualisieren Sie auch die betroffenen Pakete so schnell es geht.

Wenn Sie nach installierten Paketen suchen möchten, die zwar nicht sicherheitskritisch aber veraltet sind, können Sie [pkg_chk] verwenden. Es vergleicht die installierten Programmversionen mit den verfügbaren in Pkgsrc und gibt eine Liste der Programme aus, die durch neue Versionen ersetzt werden können.

6.3.3 *Niemals als Root arbeiten*

Arbeiten Sie niemals als Root. Root darf alles – und damit ist es ein Einfaches, das komplette System zu zerstören. Deschränken Sie unbedingt die Arbeitsschritte, die Sie als Root durchführen. Verwenden sie den Automounter AMD um Wechseldatenträger

einzubinden, kompilieren Sie Pkgsrc-Pakete oder das System nicht als Root, sondern als normaler Benutzer. Führen Sie nur die eigentliche Installation als Root durch. Verwenden Sie sudo [pp18] um einzelne Befehle als Root auszuführen.

Um von einem Benutzernamen zu einem anderen zu wechseln, können Sie su(1) verwenden. Übergeben Sie einfach als Argument den neuen gewünschten Benutzernamen und anschließend das Passwort. Mit diesem Befehl können Sie als neuer Benutzer weiterarbeiten. Dieser Befehl kann auch verwandt werden, um zum Systemverwalter Root zu wechseln, dazu muss ihr Benutzer aber Mitglied der Gruppe »wheel« sein. Wenn dem nicht so ist, wird ein `su - root` nicht gestattet werden. Fügen Sie ihren Benutzernamen mit usermod(8) der Gruppe »wheel« hinzu oder lassen Sie dies vom Systemverwalter erledigen.

Möchten Sie im System verhindern, das sich Root überhaupt an den Terminals oder der Konsole einloggen kann, müssen Sie in der `/etc/ttys` für die Terminals bzw. die Konsole das Schlüsselwort »secure« entfernen. Nun kann sich Root nicht mehr im normalen Betrieb an den Terminals einloggen. Es ist empfehlenswert die Option »insecure« unbedingt auf die Konsole anzuwenden, da nach einem Start in den Einbenutzermodus NetBSD nur die Konsole initialisiert – ist diese als »secure« markiert, erfolgt hier keine Passwortabfrage! Man bekommt also problemlos Root-Zugriff auf die Maschine!

Ein weiterer Grund, Root-Zugriff auf die Terminals zu unterbinden ist die Nachverfolgbarkeit von Logins. Wenn es mehrere Administratoren gibt, die sich als Root einloggen können, wissen Sie nicht wer sich zu einem gegebenen Zeitpunkt als Root eingeloggt hat. Wenn Sie das direkte Root-Login verbieten, müssen sich alle Administratoren als normaler Benutzer einloggen und dann per su(1) Root werden – was natürlich in den Log-Dateien verzeichnet wird.

Falls Sie sich fragen, was es mit »secure« auf sich hat, »secure« bedeutet das die Konsole in einem sicheren Raum steht, zu dem nur der Systemverwalter Zugang hat. Dies ist in Rechenzentren an Servern oder Mainframes der Fall, der normale PC-, Alpha- oder Sparc-Client steht aber in der Regel an unsicheren Plätzen – wo jeder Zugang zur Konsole hat.

[!h]

```
console"/usr/libexec/getty Pc"vt100on secure
ttyE0"/usr/libexec/getty Pc"vt220off secure
ttyE1"/usr/libexec/getty Pc"vt220on secure
ttyE2"/usr/libexec/getty Pc"vt220on secure
ttyE3"/usr/libexec/getty Pc"vt220on secure
```

Listing 6.3: /etc/ttys mit »sicheren« Konsolen

[!h]

```
console"/usr/libexec/getty Pc"vt100on
ttyE0"/usr/libexec/getty Pc"vt220off secure
ttyE1"/usr/libexec/getty Pc"vt220on
ttyE2"/usr/libexec/getty Pc"vt220on
ttyE3"/usr/libexec/getty Pc"vt220on
```

Listing 6.4: /etc/ttys mit »unsicheren« Konsolen

6.3.4 *Speicherauszüge vermeiden*

NetBSD erstellt standardmäßig einen Speicherauszug (*core dump*), wenn ein Programm abstürzt. Der Speicherauszug kann danach mit einem Debugger untersucht werden, um Fehler im Programm zu finden. Dabei enthält der Auszug allerdings relevante Teile des Arbeitsspeichers - und somit auch Informationen, die sensibel sind wie Passwörter, verschlüsselte Daten oder gar kryptographische Schlüssel.

Standardmäßig ist eine Dumpdatei auf die Rechte »0600« gesetzt und gehört Root bzw. Wheel. Trotzdem kann es einem Benutzer gelingen, Zugriff auf eine solche Datei zu erhalten, bspw. wenn das Dateisystem per NFS exportiert wird oder eine Live-CD gebootet werden kann.

Um Speicherauszüge auf Dateisystemen zu verhindern, können Sie die Option »nocoredump« beim einbinden verwenden.

Außerdem können Sie Sysctl verwenden, um einige Optionen zur Erstellung von Speicherauszügen abzuschalten, siehe Bsp. 6.5.

6.3.5 *Debugger Deaktivieren*

In einem GENERIC-Kernel ist standardmäßig der Kernel-Debugger ddb(4) aktiviert, der sich entweder mit einer bestimmten Tastenkombination aufrufen lässt (siehe Tab. 6.1) oder aufgerufen wird, wenn der Kernel abstürzt.

Mit dem Debugger lässt sich unter anderem auch der Speicher des Systems analysieren, daher sollten Sie ihn auf sicherheitskritischen Maschinen im Kernel deaktivieren. Ebenso kann der automatische Debugger-Aufruf bei Abstürzen mit `sysctl -w ddb.onpanic=0` abgestellt werden.

[!h]

```
$ sysctl -a | grep -i core
kern.defcorename = %n.core
kern.coredump.setid.dump = 0
kern.coredump.setid.path = /var/crash/%n.core
kern.coredump.setid.owner = 0
kern.coredump.setid.group = 0
kern.coredump.setid.mode = 0600 (rw-------)
proc.curproc.corename = %n.core
proc.curproc.rlimit.coredumpsize.soft = unlimited
proc.curproc.rlimit.coredumpsize.hard = unlimited
# sysctl -w kern.coredump.setid.path=/dev/null
kern.coredump.setid.path: /dev/null -> /dev/null
# sysctl -w proc.curproc.rlimit.coredumpsize.soft=0
proc.curproc.rlimit.coredumpsize.soft: unlimited -> 0
#
```

Listing 6.5: Speicherauszüge verhindern

6.3.6 *Regular Expressions considered harmful*™

Reguläre Ausdrücke sind ein wundervolles Werkzeug um die Arbeit zu vereinfachen – und sich das komplette Bein wegzublasen. Ein einfacher Tippfehler wie in Bsp. 6.6 kann schon dazu führen das wichtige Dateien gelöscht werden und wiederhergestellt werden müssen. Kontrollieren sie eine RegEx auf ihre Wirkung mit ls(1) und verwenden Sie sie nicht, wenn Sie nicht wissen, was sie tut. Vorallem nicht als Root[3].

[!h]

```
$ cd /
$ rm * .tgz
```

Listing 6.6: Das Bein mit einem Leerzeichen wegschießen

6.3.7 *Monitor und Tastatur schützen*

Wenn Sie an einem Arbeitsplatz arbeiten, an dem der Monitor frei einsehbar ist, sollten Sie Maßnahmen ergreifen um dies zu verhindern. Insbesondere wenn Sie sensible Daten bearbeiten und Publikumsverkehr haben, ist ein Bildschirmfilter sehr

[3] Obwohl es zum Lebenslauf eines Admins gehört einmal das System als Root mit einem Tippfehler zu zerlegen. Wenn man von 08:00 – 03:00 durcharbeiten muss, damit die Schlipsträger ihre Gigabyte-großen »Präsentationen« wiederbekommen, auch nicht weiter verwunderlich.

Architektur	Tastenkombination
alpha	<Ctrl>-<Alt>-<Esc> auf PC-Tastaturen
amiga	<LAlt>-<LAmiga>-<F10>
atari	<Alt>-<LeftShift>-<F9>
hp300	<Shift>-<Reset>
hpcarm	<Ctrl>-<Alt>-<Esc>
hpcmips	<Ctrl>-<Alt>-<Esc>
hpcsh	<Ctrl>-<Alt>-<Esc>
i386	<Ctrl>-<Alt>-<Esc>
	<Break> an einer seriellen Konsole
mac68k	<Command>-<Power>, oder der Interrupt-Schalter
macppc	<Command>-<Option>-<Power>
mvme68k	»Abort«-Schalter auf der CPU-Platine
pmax	<Do> an der LK-201-rcons-Konsole
	<Break> an einer seriellen Konsole
sparc	<L1>-A, oder <Stop>-A auf einer Sun-Tastatur
	<Break> an einer seriellen Konsole
sparc64	<L1>-A, or <Stop>-A auf einer Sun-Tastatur
	<Break> an einer seriellen Konsole
sun3	<L1>-A, or <Stop>-A auf einer Sun-Tastatur
	<Break> an einer seriellen Konsole
vax	<Esc>-<Shift>-D an einer seriellen Konsole
x68k	Interrupt-Schalter am Gehäuse
xen	+++++ (Fünf Pluszeichen)

Tabelle 6.1.: Tastenkombinationen um ddb(4) aufzurufen

sinnvoll. Derartige Filter werden vor die Mattscheibe gehängt und schränken den Sichtwinkel ein, so das man nur noch direkt vor dem Schirm das Bild sieht.

Die Tastatur kann man ebenfalls mit einem Blickwinkelkonzentrator vor neugierigen und unbefugten Blicken schützen. Solch ein Gerät lässt sich mit einfachen Spanplatten oder Karton leicht selbst bauen.

6.3.8 *Das System herunterfahren*

Die meisten NetBSD-Systeme laufen nicht rund um die Uhr, sondern werden bei Bedarf hoch- und runtergefahren. Normalerweise darf nur Root das System herunterfahren, der Befehl shutdown(8) kann jedoch von jedem Benutzer in der Gruppe »Operator« ausgeführt werden. Mit der Option `-h now` wird der Rechner sofort

heruntergefahren, mit `shutdown -r 09:45` startet den Rechner um dreiviertel Zehn neu.

Neuere PC-Hardware verfügt in der Regel über ACPI- bzw. APM-fähige Netzteile, können also über den Ein-/Aus-Taster an- und ausgeschaltet werden. Standardmäßig ist die ACPI-Unterstützung im GENERIC-Kernel deaktiviert. Sie benötigen also einen eigenen Kernel der die ACPI-Optionen aktiviert hat. Damit können Sie den Rechner über den Ein-/Aus-Taster herunterfahren und abschalten.

SIEHE AUCH:
`man 5 rc.conf`
`man 8 rc.d`
`man 1 su`
`man 4 options`
`man 4 lkm`
`man 3 regex`

7

ZEN, ODER: DIE KUNST EIN PASSWORT ZU FINDEN

Gewöhnlich glaubt der Mensch, wenn er nur Worte hört, es müsse sich dabei doch auch was denken lassen.

(Johann Wolfgang von Goethe)

7.1 GEFAHREN FÜR PASSWÖRTER

Bevor wir klären was ein starkes Passwort eigentlich ist, besprechen wir die häufigsten Gefahren für Passwörter.

7.1.1 *Klartext-Passwörter*

Einige Netzwerkdienste wie FTP, Telnet oder RSH übertragen Passwörter im Klartext. Dies ist in nicht-vertrauenswürdigen Netzen problematisch, da das Passwort auf dem Übertragunswege abgefangen werden kann. Daher müssen alle Protokolle und Dienste, die Daten im Klartext übertragen durch sichere Alternativen wie SSL oder SSH ausgetauscht werden oder in einem kryptographischen Tunnel bzw. VPN gesichert werden. Ebenso kann der Inhalt der Kommunikation verschlüsselt werden, bspw. Emails mit PGP.

7.1.2 *Social-Engineering*

Social-Engineering ist eine Angriffsstrategie, die nicht die Technik als Opfer auserkoren hat. Stattdessen wird hier viel lieber, und vor allem effizienter, das schwächste Glied in der Sicherheitskette angegriffen – der Mensch.

Ein Angreifer verwendet verschiedene Strategien und Taktiken um aus Benutzern der Systeme Informationen herauszuholen. Mithilfe dieser Informationen kann er

erfolgreiche Angriffe gegen Zielsysteme fahren.

Derartige Techniken sind äußerst erfolgreich wenn größere Organisationen wie Großunternehmen, Behörden oder Universitäten angegriffen werden sollen. Oftmals gelingt es einem Angreifer mit der Aggregation öffentlich zugänglicher Informationen und entsprechender Kaltschnäuzigkeit wichtige Informationen und Zugriff auf geschützte Systeme zu erlangen.

Eine weitverbreitete Masche ist das sogenannte Phishing, bei dem mit gefälschten Emails oder Webseiten Benutzer dazu verleitet werden sollen Benutzerdaten zu ihren Bankkonten oder anderen Konten wie Email oder in Verkaufssystemen. Es gibt keine rein-technischen Mittel gegen Phishing[1], da das Angriffsziel nunmal ein Soziales ist.

Gegen soziale Attacken können soziale Maßnahmen helfen, dies beinhaltet Verträge, Richtlinien, Gebote, Verbote und andere Maßnahmen. So kann man bspw. festlegen das am Telefon niemals das Rechenzentrum nach einem BenutzerPasswort fragen wird und Benutzer daher niemals ihr Passwort herausgeben sollen. Diese Regel erstreckt sich auch auf vertrauenswürdige Kollegen. Niemals sollte ein Passwort irgendwem anvertraut werden. Auch keinem »netten« Kollegen. Sinn des Passwortes ist es, ein Benutzerkonto einer realen Person zuzuordnen. Wenn mehr als eine Person das Passwort kennt, ist diese Maßnahme unterminiert. Und damit zumindest der offizielle Konteninhaber in Schwierigkeiten.

Zu einer Sicherheitsmaßnahme gehört immer die Etablierung einer Sicherheitsrichtlinie, die *alle sozialen und technischen Aspekte* umfasst.

Teile einer Sicherheitsrichtlinie sind beispielsweise:

- Verwundbare Teile des Netzwerkes abschotten und sichern
- Physikalischen Zugriff auf Rechner unterbinden[2].
- Rechte der Benutzer soweit es geht einschränken, dabei aber natürlich nicht übertreiben. Zu starke Einschränkungen werden gerne übergangen.
- Benutzer unbedingt über Social-Engineering-Angriffe informieren und von Gegenmaßnahmen in Kenntnis setzen.
- Proaktive Passwortprüfungen
- rechtliche Maßnahmen (Abmahnungen, Kündigungen etc.) gegen Verstöße

[1] *»If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology«* - Bruce Schneier

[2] Heckler & Koch produzieren hervorragende Sicherheitsmaßnahmen

Social Engineering ist die erfolgversprechendste Methode, und meist auch die am wenigsten aufwendigste. Warum sollte man sich die Mühe machen, ein Passwort zu knacken, wenn man mit etwas Redegewandheit Passwörter frei Haus geliefert bekommt?

7.1.3 *Tastaturrekorder*

Um an Passwörter z. B. für PGP-Schlüssel oder an vertrauliche Daten zu gelangen setzen Einbrecher oft Tastaturrekorder ein.

Diese können als Software ausgeführt sein, die im Hintergrund alle Tastatureingaben aufzeichnet und auch Schnappschüsse vom Bildschirm macht. Die gewonnenen Daten werden dann meist an den Angreifer zurückgesandt. Solche Programme werden z. Zt. häufig durch manipulierte Emails verteilt und sollen Eingaben auf Login-Seiten z. B. zur Bank oder zu einem Mailprovider abfangen.

Neben der Software-Variante gibt es noch Rekorder als Hardware. Diese werden zwischen Tastatur und Computer gesteckt und schleifen das Signal, welches dann aufgezeichnet wird, durch. Sie sind nicht größer als bspw. USB-PS2-Adapter und sind systemunabhängig. Dafür erfordern sie aber physikalischen Zugriff auf das System um sie einzuschleusen und wieder abzuschöpfen. Außerdem lassen sie sich durch einen Blick aufs Kabel sehr leicht enttarnen. Es gibt aber inzwischen auch Varianten, die *in* die Tastatur eingebaut werden. Solche Varianten sind natürlich schwer zu enttarnen.

Neben einem Tastaturrekorder kann man auch Kameras verwenden, um das Tastenfeld zu überwachen. Dies geschieht häufig an Bankautomaten und lässt sich mit den heutigen kleinen Spionagekameras problemlos auf Büroumgebungen oder Privaträume übertragen. Gegen derartige Maßnahmen helfen Stör- und Peilsender, wenn die Wanzen Funkübertragung verwenden.

7.1.4 *Kompromittierende Strahlung*

Jedes elektrische Gerät, sei es ein Fernseher, eine Mikrowelle oder eine Waschmaschine, emittiert elektromagnetische Strahlung. Insbesondere Geräte die Daten verarbeiten oder übertragen, wie Rechner, Netzwerkkabel oder Tastaturen, strahlen dabei auch die verarbeiteten bzw. übertragenen Daten ab. Diese Strahlung kompromittiert die Sicherheit des Systems, daher auch der Name der kompromittierenden Strahlung.

Das zum Abfangen notwendige Gerät ist aber im allgemeinen recht teuer und wird

nicht von Skript-Kiddies oder Phishern eingesetzt. Trotzdem ist dieser Kanal ergiebig in der Informationsbeschaffung und kann somit das gesamte Sicherheitssystem aushebeln. Als Gegenmaßnahme gibt es spezielle Rechner und Kabel die entsprechend auf Elektromagnetische Verträglichkeit geschirmt sind. Außerdem gibt es Störsender, die ein Abfangen der Emission verhindern. Diese Geräte werden meist nur im Militär und Geheimdienst bzw. speziellen Gebieten wie Luftfahrt oder Medizintechnik eingesetzt und sind daher recht teuer.

Für den einfachen Hausgebrauch gibt es inzwischen isolierende Tapeten. In diese sind Metallfasern eingearbeitet, die einen schirmenden Faradayschen Käfig erzeugen. Ein Zaun aus Hühnerdraht der rundherum um die Quelle gezogen wird, kann ebenfalls einen brauchbaren faradayschen Käfig erzeugen. Eine wirklich effektive Schirmung erreicht man mit speziellen Kupferplatten, die bspw. in Funklaboren eingesetzt werden, aber entsprechend teuer, groß und schwer sind.

Datenkabel, die nicht durch Schirmung geschützt werden können, bzw. deren Schirmung nicht bewacht werden kann, müssen kryptographische Tunnel einsetzen.

Weiterhin muss man natürlich gewünschte Strahlung eingrenzen bzw. abschaffen, dies betrifft z. B. WLAN, Bluetooth, IrDA oder Funktastaturen. Derartige Geräte sind zu meiden und idealerweise komplett zu verbieten.

7.2 Was ist ein starkes Passwort?

Um einen Rechner zu schützen, muss *jedes* Benutzerkonto mit einem *starken* Passwort versehen sein. Um festzustellen ob ein gewähltes Passwort hart ist, werden wir es wie ein Einbrecher analysieren.

Eine wichtige organisatorische Maßnahme ist die Einrichtung eines eigenen Kontos für jeden Benutzer. Nur so lässte sich ein Benutzer eindeutig identifizieren. Es ist sinnlos in einem Log einen Benutzernamen ausfindig zu machen, wenn mit diesem Benutzer 20 verschiedene Mitarbeiter arbeiten können.

Der einfachste Weg, in einen Rechner einzubrechen, ist ein Passwort zu erraten. Normalerweise tendieren Menschen dazu sich ein einfaches Passwort auszudenken, da sie es sich ja merken müssen. Also wählen sie den Namen ihres Hundes, Kindes, Autos oder den Wohnort, Geburtsjahr oder ähnliche naheliegenden Dinge. Dies machen sich Einbrecher zu nutze, in dem sie einfach versuchen das Passwort zu erraten. Außerdem nutzen Einbrecher ebenfalls Social-Engineering-Taktiken. *Social Engineering* kommt ursprünglich aus der Soziologie, bezeichnet im Hacker-Umfeld aber Methoden, mit denen man versucht Menschen zu hacken. Dazu kann sich ein Einbrecher bspw. am Telefon als Vorgesetzter oder Administrator ausgeben und

versuchen dem Opfer im Gespräch Kontodaten zu entlocken. Oder man sammelt verfügbare Informationen über Mitarbeiter, bspw. indem man deren Webseite oder Blog durchsucht. Wenn sich auf der Webseite Bilder einer Katze befinden, liegt es doch nahe das ihr Name das Passwort ist, oder?

Um derartige Passwortangriffe durch raten zu erschweren, pausiert NetBSD nach 5 fehlgeschlagenen Anmeldeversuchen für 15 Sekunden.

Hier nun einige Regeln für sichere Passwörter:

7.2.1 *Verwenden Sie kein Passwort das erraten werden kann!*

Benutzen Sie nicht den Namen ihrer Kinder, des Lieblingsvereines, Hundes oder ähnliches. Auch nicht sehr offensichtliche Wörter sind unbedingt zu vermeiden. Alles was sich in der näheren Umgebung des Rechners befindet ist ebenfalls eine schlechte Inspirationsquelle für Passwörter. The truth[3] is out there.

Ein Angreifer, der es auf Ihr Konto abgesehen hat, wird mit den oben beschriebenen Social-Engineering-Methoden ihre Umgebung aufklären. Also möglichst alle verfügbaren Informationen sammeln, die auf ein Passwort schließen lassen und diese Erkenntnisse in seine Einbruchsversuche einfließen lassen.

7.2.2 *Verwenden Sie kein Passwort das in einem Wörterbuch steht!*

Würde man Benutzernamen und Passwort im Klartext in einer Datei ablegen, könnte man das Passwort einfach auslesen. Damit dies nicht geschieht, wird es mit einer sogenannten Einwegfunktionen (engl. *one way hash*) verschlüsselt. Dazu wird das Passwort mit dieser speziellen mathematischen Funktion in einen sog. Hash umgewandelt. Es ist aber nicht möglich aus dem Hash das Passwort zu berechnen. Möchte sich jemand am System anmelden, gibt er sein Passwort an. Die Eingabe wird dann mit der entsprechenden Funktion in den Hash überführt, (Abbildung 7.1 zeigt ein Beispiel) anschließend werden beide Hashes verglichen. Wenn das Passwort korrekt ist, stimmen beide Hashes überein und der Benutzer wird am System angemeldet.

Gelingt nun ein Einbrecher in den Besitz der Passwort-Datei (`/etc/master.passwd`) kann er die Passwörter nicht einfach auslesen. Da die mathematischen Funktionen

[3] Fox Moulder hatte in »Akte X« über seinem Monitor ein Poster hängen. Das Poster zeigte eine Fliegende Untertasse mit dem Untertitel »The Truth is Out There«. Als Dana Scully an Moulders Rechner mußte, tippte sie »Truth« als Passwort ein und war drin.

normalerweise sicher sind [4] kann aus den Hashes das Passwort nicht berechnet werden.

Was man aber trotzdem tun kann, ist selber derartige Hashes zu erzeugen und diese mit den erbeuteten zu vergleichen. Dazu kann man einen Generator oder ein Wörterbuch verwenden, wobei das Wörterbuch nichts anderes als eine simple Wortliste ist. Wird bei einem Benutzerkonto nun ein Passwort verwendet, das im Wörterbuch steht, wird es der Einbrecher finden. Daher dürfen keine Passwörter verwandt werden, die in einem Wörterbuch stehen. Auch wenn die Generierung von Passwort-Hashes dauert, kann man die Berechnung auf mehrere Rechner verteilen und die Ergebnisse der Falltürfunktion in Datenbanken ablegen.

Im Netz kursieren sogenannte Regenbogenlisten (engl. *Rainbow Tables*), dies sind Dateien, in denen Passworthashes bereits berechnet wurden. So kann man beispielsweise den Inhalt aus /usr/share/dict an einen Passwortgenerator verfüttern und die entstandenen Hashes zum schnellen Cracken von Passwortdateien verwenden. Dies ist möglich, da eine Hashfunktion über einer Zeichenkette immer den selben Hashwert erzeugt. Egal wann man auf welchen Rechner den SHA1-Wert von »08/15« berechnet, er wird immer gleich[5] sein. Da dies die beschriebenen Regenbogenlistenattacken ermöglicht, wurde eine Zufallskomponente in die Hasherzeugung eingefügt. Dieser sogenannte »Salt« (dt. Salz, weil er ja die Würze gibt) ist eine einfache Zeichenkette, die zufallsgeneriert und dem Passwort angehängt wird.

```
# grep testuser /etc/master.passwd
testuser:$sha1$7$7BWO8GFC$BGwFt9i6lnQ6jB5.3nb8YnP75xAF:1009:100::0:91231:\
Cliff Alistair McLaine, Orion-8, Terrestrische Raumaufklärung:\
      /home/testuser:/bin/ksh
```

Abbildung 7.1.: Beispiel für einen Benutzereintrag aus NetBSDs `/etc/master.passwd`

7.2.3 *Verwenden Sie ein langes Passwort mit Groß- und Kleinschreibung, Zahlen und Sonderzeichen!*

Nun kann man natürlich auf die Idee kommen ein Wort mit ein, zwei Zahlen zu erweitern, da soetwas nicht im Wörterbuch steht. Statt eines Wörterbuches kann man für die oben beschriebene Attacke aber auch einen Generator verwenden, der Wörter erzeugt oder ergänzt. So findet man auch Passwörter wie »Waldi1«, »Anja85« oder »Magdeburg*«.

[4] (Das traditionelle crypt(1) bzw. DES ist nicht mehr sicher, aber nach dreißig Jahren ist das normal.)
[5] baa4ef9f76419f3361826c618418ac6a441f7c6

Wenn ein Einbrecher nun alle möglichen Passwörter durchkalkulieren möchte, muss er

$$\text{verfügbare Buchstaben}^{\text{Passwortlänge}}$$

Kombinationen berechnen.

Verwendet man nur die 26 Kleinbuchstaben a-z und das Passwort ist 5 Zeichen lang, sind dies $26^5 = 11.881.376$ Kombinationen. Verwendet man stattdessen 52 Groß- und Kleinbuchstaben (a-z, A-Z), 7 Umlaute, 10 Ziffern und 30 Sonderzeichen mit 10 Buchstaben, kommt man auf $99^{10} = 90.438.207.500.880.449.001$ Kombinationen.

Geht man nun davon aus das ein Rechner pro Sekunde 5 Kombinationen berechnen kann, benötigt man im ersten Falle $\frac{11881376}{(5*60*60*24)} = 27,5$ Tage, was durchaus ein realistischer Einbruchserfolg ist. Im zweiten Falle sind es aber schon 573555349 Jahrtausende[6] und selbst wenn statt 5 Kombinationen pro Sekunde 5000 berechnet werden, benötigt man immer noch 573555 Jahrtausende.

Diese Berechnung setzt natürlich voraus, das der Angreifer die Länge der Passwörter kennt. Ist dem nicht so, muss er alle möglichen Permutationen aller Längen durchrechnen, das ergibt dann

XXX

Permutationen. Halten Sie also ggf. eine Passwortrichtline, die derartige Vorgaben enthält, geheim.

7.2.4 *Das Passwort muss geheim bleiben!*

Eigentlich eine einfache Regel, die aber trotzdem oft missachtet und daher erfolgreich für Angriffe ausgenutzt wird.

Oft finden sich Passwörter auf einem Klebezettel unterhalb der Tastatur oder am Monitor versteckt, oder man teilt es Kollegen mit, damit diese irgendwas am Rechner tun können. Oder man verwendet überall das selbe StandardPasswort, bspw. in igendwelchen Webforen und Freemailern.

Dies muss strikt unterbleiben. Es nutzt nichts sich ein starkes Passwort auszudenken, wenn es dann in die Weltgeschichte herausposaunt wird.

[6] Nicht einmal ein NetBSD-Rechner dürfte eine derartige Uptime erreichen ;-)

7.2.5 *Verwenden Sie nicht überall das selbe Passwort!*

In der heutigen Zeit benötigt man in der Regel mehr als nur ein bis zwei Passwörter. Insbesondere in irgendwelchen Webforen oder für diverse Emailadressen, die als Spamfalle dienen, werden Passwörter benötigt. Sorgen Sie dafür das Sie nicht überall das selbe Passwort oder Passwortmuster verwenden. Wenn Sie sich bspw. an einem Webforum anmelden, müssen Sie meist noch eine Emailadresse angeben – verwenden Sie also nicht für beide das selbe Passwort, denn niemand kann garantieren daß das Passwort im Webforum geheim bleibt.

Benutzen Sie für ihre wichtigen Konten (private Email, PGP, Rechner etc.) starke Passwörter und verwenden Sie diese nirgends sonst. Notieren Sie sich gegebenenfalls Passwörter für »unwichtige«, selten genutzte Konten wie Foren oder ähnliches. Achten Sie aber darauf, daß niemand ohne weiteres an diese Notizen herankommt.

Insbesondere Webforen werden häufig Opfer automatisierter Attacken. So gelang es bspw. in ein privates Polizei-Forum einzubrechen. Die Passwörter für das Forum funktionierten auch bei einigen hinterlegten Emailadressen angemeldeter Benutzer.

Verwenden Sie also nicht überall das selbe Passwort!

7.2.6 *Passwortwechsel*

Häufig taucht in der Literatur die Forderung auf, das Passwort regelmäßig und häufig zu wechseln. Dies mag von einem rein technischen Standpunkt vertretbar sein, ignoriert aber die psychologische Komponente. Zuerst einmal stellt sich die Frage, wann es technisch notwendig ist, ein Passwort zu wechseln. Wie in Kapitel 7.2 dargelegt, gibt es einige Grundregeln für Passwörter. Wenn man sich an diese hält und hinreichend starke sowie nicht-erratbare Passwörter erzeugt, fällt der erzwungene Passwortwechsel weg. Denn rein von statistischer Warte her ist dieser nur innerhalb der Zeit notwendig, die nötig ist, um das Passwort mit einer Wörterbuchattacke zu erraten.

Verwendet man nun, wie in Kap. 7.2.3 dargelegt, ein starkes Passwort, das innerhalb des Lebenszykluses des Passwortes nicht erraten oder geknackt werden kann, ist ein Wechsel aus technischer Sicht nicht notwendig.

Es empfiehlt sich sogar auf erzwungene Passwortwechsel zu verzichten. Hat man nämlich eine restriktive Passwortrichtline in Kraft und setzt diese auch technisch um, erschwert man den Benutzern die Passwortwahl und auch die Möglichkeit, sich das Passwort zu merken. Dies hat natürlich zur Folge, das die Passwörter notiert

werden, was einen neuen Angriffsvektor eröffnet. Außerdem werden dann in der Regel schnell SerienPasswörter etabliert, die einfach durchnumeriert werden. Ein Kunde hatte für eine Arbeitsgruppe einen Laptop eingerichtet, der am Monatsersten ein neues Passwort verlangte. Diese wurden dann clevererweise durchnumeriert, von »Arbeitsgruppe1« bis »Arbeitsgruppe12«, je nachdem welcher Monat gerade war. Dies hat natürlich die ursprüngliche Intention des Passwortwechsels ad absurdum geführt.

Dies heißt natürlich nicht, das Passwortwechsel generell schlecht sind. Nur muß man beachten, daß die Benutzer sich ihre Passwörter auch merken können. Klären Sie Ihre Benutzer im Vorfeld über diese Maßnahme auf und wählen Sie die Passwörtlaufzeit nicht zu kurz. Monatliche Wechsel sind nicht vertretbar, halbjährliche schon eher. Stellen Sie Ihren Benutzern auch die Möglichkeit zur Verfügung, automatisch Passwortvorschläge zu generieren.

7.3 Passwörter von Hand generieren

Nachdem wir nun festgestellt haben was starke Passwörter sind, müssen wir uns der Aufgabe stellen Passwörter zu finden.

Eine gute Möglichkeit ein Passwort auszudenken ist es, sich einen Satz auszudenken und aus diesem Satz die Initialen zu einem Passwort zusammenzuziehen, etwa in der Art:

Wem der große Wurf gelungen,
Eines Freundes Freund zu sein.
- Friedrich Schiller

wird zu

WdgWg, EFFzs. – FS

Somit erhält man schon ein recht brauchbares Passswort, an das man bspw. noch »1805« anhängen kann. Weiter härten kann man es, in dem man bestimmte Buchstaben durch Sonderzeichen ersetzt (E=3, I=1, S=5 usw. usf.) und so *WdgWg,3FFz5.-F51805* erzeugt.

Nun muss man sich nur noch den Ursprungssatz und die Ersetzungsregeln merken.

Es gibt nahezu unendlich viele solcher Eine andere Variante ist es, Wörter mit Zahlen zu vermischen, etwa so:

Ringparabel → R1i2n3g4p5a6r7a8b9e10l

oder so:

22.01.1729 + Lessing → 2L2e0s1s1i7n2g9.

Oder mischen Sie Wörter und Daten durch. Nehmen wir als Beispiel zwei Namen und Geburtsjahre: *Lara 1970* und Tonja 1973. Man kann einen Namen umdrehen, mit dem anderen Namen vermischen und die Jahreszahlen anfügen. Dabei wird einfach bei 19 und 73 die Feststelltaste gedrückt, so das Sonderzeichen statt Ziffern verwendet werden. Das Ergebnis ist dann:

19/§a*Lj*a*n*r*o*a*t*!*)*70

Der Phantasie sind hier keine Grenzen gesetzt, so das Sie eine Vielzahl von Passwörtern generieren können, die leicht merkbar sind.

7.4 Passwörter automatisch generieren

Apg ist der »automated password generator«, der Passwortvorschläge generieren kann. Apg unterstützt zwei verschiedene Algorithmen, einen aus FIPS-181 um aussprechbare, silbenbasierte Passwörter zu erzeugen, und einen anderen, der komplett zufällige Passwörter erzeugt. Weiterhin kann Apg Wörterbuchdateien verwenden um die erzeugten Passwörter auf ihre Stärke zu prüfen oder um bestimmte Wörter als Passwort zu verhindern. Installiert werden kann er aus pkgsrc/security/apg.

Abbildung 7.2 erzeugt aussprechbare Passwörter mit Apg, Abbildung 7.3 generiert verschiedene starke Passwörter, die allerdings schwerer zu merken sind. Dabei wird mit der Option `-M SNCL` der Einsatz von Groß-, Kleinbuchstaben, Ziffern und Sonderzeichen erzwungen und mit `-E` die Zeichen »0123« ausgeschlossen. Die Option `-r` verwendet das angegebene Wörterbuch zum Vergleich der erzeugten Passwörter und schließt sie bei einem Treffer aus. Zum Schluß wird die Option `-s` verwandt, die vom Benutzer eine Eingabe erfordert, um die ersten Zufallswerte zu generieren.

Apg ist eine gute Hilfe wenn man keine Idee für ein neues Passwort hat, oder Passwörter automatisch für neue Benutzer generieren muss. Es kann manchmal vorkommen das Apg hängt, weil die Zufallsgenerierung Daten benötigt. Passiert

```
$ apg -a 0 -m 8 -x 12
DahyRijHypjo
wrisEij8
Nuphyeulvuk
itsAcugHyd9
kribazGuc
Ot0knelun
$
```

Abbildung 7.2.: aussprechbare Passwörter mit apg erzeugen

```
$ find /usr/share/man/ | xargs md5 > /dev/null &
$ apg -n 10 -m 8 -x 14 -M SNCL -E 0123 -r verbotenePasswörter -s
Please enter some random data (only first 8 are significant)
(eg. your old password):>
kriWop;sloc8
Nit7Wrejyal=
Ek6OkdiashBip‘
yedIm+VohonOm4
AleajIs!Swat6
&OroyThukPoct8
7OthivRoak?
Izvif!opIt9
AkwojyuIr*ow8
4upipAl)
$
```

Abbildung 7.3.: nicht-aussprechbare Passwörter mit Apg erzeugen

Apg erzeugt 10 Passwörter der Länge 8 - 14 und verwendet dazu Ziffern, Sonderzeichen, große und kleine Buchstaben, schließt aber "0", "1", "2", "3" aus. Das erzeugte Passwort wird anschließend auf Vorkommen in /home/stefan/verbotenePasswörter überprüft und bei einem Treffer verworfen. Die Option `-s` erfordert vom Benutzer die Angabe von 8 Zeichen, die als Initialisierungsvektor für den Zufallsgenerator verwendet werden. Der vorhergehende Find-Befehl dient dazu, Last im System zu erzeugen, um den Zufallsgenerator mit Entropie zu versorgen.

dies, bspw. wenn man sehr viele Passwörter generiert, dann hilft es im Hintergrund einfach etwas Aktivität zu erzeugen, bspw. mit `find / | xargs md5 > /dev/null &`, wie in Abbildung 7.3 angewandt.

Neben Apg existiert noch shuffle(1) im System, das einfach die übergebenen Optionen

zufällig sortiert wieder ausgibt. Diese Funktion kann bspw. genutzt werden um einen Salt zu erzeugen: `shuffle -p 6 ‘seq 1 49‘` gibt 6 zufällig ausgewählte Elemente der Liste 1 - 49 wieder aus.

7.5 PASSWORT-HASHES GENERIEREN

Um einen neuen Benutzer unter NetBSD anzulegen, verwendet man das Programm useradd(8). Dort kann man auch ein Passwort übergeben - allerdings nur als Passwort-Hash, und nicht im Klartext.

Um einen solchen Passwort-Hash zu generieren, kann man OpenSSL oder pwhash(1) verwenden. Möchte man eine Liste von Benutzerkonten einrichten, kann man mit pwhash(1) die gehashten Passwörter an useradd(8) übergeben. Abbildung 7.4 zeigt dazu mehrere Beispiele. Zuerst werden mit OpenSSL mehrere BSD-MD5-Passwort-Hashes erzeugt, anschließend wird mit pwhash ein MD5-Hash erzeugt, ein Passwort für einen neuen Benutzer erzeugt und zu guter Letzt ein einfaches Skripte gezeigt, das zwei Dateien mit Benutzernamen und Passwörtern ausliest und die Benutzer mit zufälligen Passwörtern erzeugt und ein Protokoll in die Datei »Benutzerliste« schreibt. Die Option -F von Useradd zwingt die Benutzer beim ersten Login ihr Passwort zu wechseln.

```
$ openssl passwd -salt AA MyPassword
Warning: truncating password to 8 characters
AAaV85M5Ph8j6

$ openssl passwd -table -1 -in userlist
foo     $1$C9/uTqZa$s1/X4ihpfvG/.m2EYzbZV1
bar     $1$7NDpX3zi$mBKk39Z9k6s4zbBFkPYt4.
baz     $1$bZywpxFq$xyjJAsf6MW.pX3VohvwJE.


$ pwhash -m md5test
$1$a4w7f0jp$/U7Flz7YAx77KaFl2Grnl.


# useradd -g mygroup -d /home/testuser -m -p \
      `pwhash -b 9 ?MyP455W0Rd!` testuser

# for i in `cat userlist`; do j=$i;
  k=`shuffle -0 -p 8 a b c [...] x y z A B C [...] X Y Z 0 1 [...] 8 9`
  useradd -F -g mygroup -d /home/$j -m -p `pwhash -m  $k` $j;
  echo "Benutzer '$j' mit Passwort $k angelegt" | tee -a Benutzerliste
  done
```

Abbildung 7.4.: Passwort-Hash mit OpenSSL oder pwhash(1) erzeugen

7.6 PASSWÖRTER IN NETBSD

NetBSD verwendet sogenanntes »password shadowing«, d.h. die sensiblen Passwort-Daten werden aus der `/etc/passwd` in die geschützte `/etc/master.passwd` ausgelagert. Da die `master.passwd` nur mit System-Rechten zugreifbar ist, kann ein einfacher Benutzer die Passwort-Hashes nicht mehr zu einer Wörterbuchattacke heranziehen. Steht in der `/etc/master.passwd` anstelle eines Hashes ein oder mehrere Asteriske, ist dem Benutzer ein Login nicht mehr möglich, da ein Hash niemals ein »*« sein darf. Befindet sich gar kein Zeichen im zweiten Feld, ist das Passwort leer. Das heißt man kann sich unter diesem Benutzernamen anmelden, ohne ein Passwort anzugeben. Dies darf natürlich absolut nicht vorkommen und der entsprechende Benutzer muß sofort deaktiviert werden. Finden lassen sich derartige Benutzer mit einer simplen AWK-Zeile, die als Root ausgeführt wird: `awk -F: 'length($2)<1 {print $1}' /etc/master.passwd`

Ebenfalls gefährlich sind Standardbenutzer, die mit Standardpasswörtern versehen

und bereits im Basissystem vorhanden sind. Dies sind meist Konten wie »games« oder »daemon« oder Benutzer die von Anwendungspaketen installiert werden, wie PostgreSQLs »pgsql«. Standardmäßig werden derartige Konten in NetBSD mit einem Asterisk deaktiviert und müssen vom Systemadministrator explizit freigeschaltet werden. Andere Systeme oder Anwendungsprogramme sind teilweise nicht so sorgfältig und gefährden die Systemsicherheit.

`/etc/master.passwd`:

- Benutzernamen
- Passwort-Hash
- User-ID
- Gruppen-ID
- Login-Klasse
- Zeit bis zum nächsten Passwortwechsel
- Restlaufzeit des Kontos
- Allgemeine Informationen über den Benutzer (GECOS),
- Heimatverzeichnis des Benutzers
- Login-Shell des Benutzers

`/etc/passwd`:

- Benutzernamen
- einen Asterisk (»*«) anstelle des Passwort-Hashes
- User-ID
- Gruppen-ID
- Allgemeine Informationen über den Benutzer (GECOS),
- Heimatverzeichnis des Benutzers
- Login-Shell des Benutzers

Abbildung 7.5.: Aufbau der /etc/master.passwd und /etc/passwd

NetBSD kann verschiedene Passwort-Hashes zur Passwort-Hash-Generierung benutzen. Welcher verwendet wird, kann in der Datei `/etc/passwd.conf` festgelegt werden. Zur Zeit werden »old«, »newsalt«, »md5«, »sha1« und »blowfish« unterstützt, wobei »old« nur Passwörter unterstützt, die maximal acht Zeichen lang sind. Empfehlenswert sind »sha1« oder besser noch »blowfish« mit entsprechend hohen Rundenzahlen. Blowfish mit 15 Runden benötigt auf dem Testsystem[7] bereits 15 Sekunden, wie in Tab. 7.6 dargestellt. Je höher die Rundenzahl ist, desto länger benötigt der Login-Prozeß bzw. eine Runde einer Wörterbuchattacke. Für einen Benutzer mag es vertrebar sein 4 oder 8 Sekunden auf sein Login zu warten, eine vernünftige Wörterbuchattacke wird mit diesen Werten nicht durchführbar.

Wenn Sie den Passworthash verändert haben, muss das Passwort mit passwd(1)

[7] AMD Athlon Model 4 (Thunderbird) (686-class), 1333,46 MHz, 1GB RAM, 60GB ATA100-Platte, NetBSD 4.99.3, pwhash(1) 1.14

R	r	u	
10	0.48s	0.48s	0.0
11	0.97s	0.96s	0.0
12	1.92s	1.92s	0.0
13	3.84s	3.83s	0.0
14	7.68s	7.65s	0.0
15	15.36s	15.31s	0.0
16	31.42s	30.62s	0.0
17	61.95s	61.23s	0.0
18	123.06s	122.45s	0.0
19	248.31s	244.95s	0.0
20	639.28s	490.16s	0.2

R: Rundenzahl
r: real, u: user, s: system

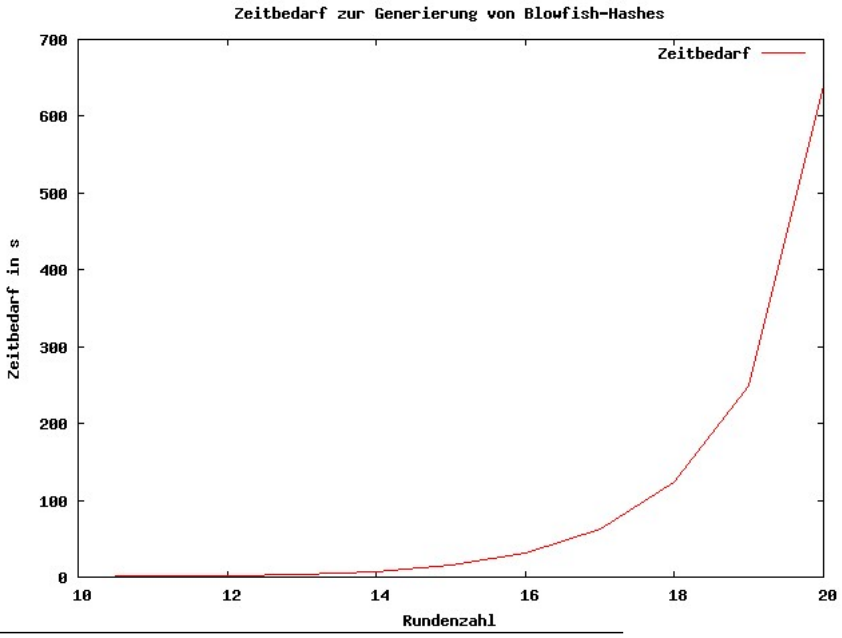


Abbildung 7.6.: Zeitbedarf zur Generierung von Blowfish-Hashes

geändert werden, damit es mit der neuen Hashmethode erzeugt wird.

```
default:
  localcipher = blowfish,7
  ypcipher = md5

myuser:
  localcipher = blowfish,9
  ypcipher = blowfish,9

:mygroup:
  localcipher = blowfish,7
  ypcipher = newsalt,9

root:
localcipher = blowfish,25
ypcipher=blowfish,25
```

Abbildung 7.7.: Konfigurationsmöglichkeiten in der /etc/passwd.conf

7.7 Passwörter cracken

Um zu erfahren wie Angreifer Passwörter cracken und um überprüfen zu können ob die eigenen Passwörter wirklich sicher sind, werden wir sie mit einem Cracker bearbeiten. Doch Achtung, das cracken fremder Passwortdateien ist definitiv verboten! Auch wenn Sie als Administrator die Passwörter ihrer eigenen Server cracken wollen, holen Sie sich vorher vom Verantwortlichen eine *schriftliche* Erlaubnis! Arbeiten Sie am besten vorher eine Richtlinie aus, die den Umgang mit den gecrackten Daten klärt. Es kann auch nicht schaden eventuell vorhandene Arbeitnehmervertreter hinzuzuziehen.

7.7.1 *Mit Crack*

Crack von Alec Muffet ist einer der verbreitetsten Passwortcracker. Er verwendet Wortlisten, kann die erzeugten Wörter aber anhand von Reglen manipulieren. Man kann so beispielsweise Zahlen an Wörter anhängen, Wörter verdoppeln, umdrehen oder die Groß- und Kleinschreibung durchpermutieren.

Die meisten Cracker unterstützen verschiedene Angriffs-Modi, so kann beispielsweise der Inhalt der Benutzerdaten in den Gecos-Feldern als Passwort durchprobiert werden. Weiterhin benutzen Cracker Wortlisten um die oben beschriebene Wörterbuchattacke zu fahren. Dazu benötigen Sie Wortlisten. Einige finden sich hier[8] bzw. im Wortschatz der Uni Leipzig unter[9] sowie zahlreiche weitere Verweise in der Wortdatenbank[10]

7.7.2 *Mit John the Ripper*

John the Ripper ist ein umfangreicher Passwortcracker, der verschiedene Passwort-Hashes angreifen kann. Dazu verwendet John verschiedene Modi, die die Passwortdatei unterschiedlich analysieren.

Im Single-Modus (`--single`) benutzt John die Gecos-Felder der Passwortdatei um das Passwort zu raten. Mit dieser Variante findet man sehr schnell alle Benutzerkonten, deren Passwort vom Benutzernamen, Büronummer oder ähnlichem abgeleitet ist. Ein weiterer Modus ist der Wortlisten-Modus (`--wordlist`), bei dem der Inhalt der Wortlisten-Datei als Passwort durchprobiert wird. Der umfangreichste und

[8] ftp://coast.cs.purdue.edu/pub/dict/wordlists/
[9] http://wortschatz.uni-leipzig.de/html/wliste.html
[10] http://www.wortdatenbank.de/

mächtigste Modus ist der inkrementelle Modus (`--incremental`), hierbei fährt John Brute-Force-Attacken, das heißt es werden alle möglichen Passwortkombinationen durchprobiert. Mit diesem Modus kann jedes Passwort erraten werden, er ist aber äußerst zeitaufwändig.

```
$ time john --single passwd --session=laptoptest
Loaded 6 password hashes with 6 different salts (FreeBSD MD5 [32/32])
Guericke         (test6)
test4            (test4)
5tset            (test5)
guesses: 3  time: 0:00:00:01 100%  c/s: 2098  trying: test21900
    1.10s real     1.07s user     0.02s system

$ time john --wordlist=/usr/share/dict/web2 passwd --session=laptoptest
Loaded 3 password hashes with 3 different salts (FreeBSD MD5 [32/32])
Berliner         (test1)
guesses: 1  time: 0:00:03:47 100%  c/s: 2152  trying: Zyzzogeton
  227.25s real   226.46s user     0.01s system
```

Abbildung 7.8.: Passwörter cracken mit John

7.8 EINWEGPASSWÖRTER MIT S/KEY

S/Key implementiert EinwegPasswörter basierend auf einem Anruf-/Rückantwort-Verfahren (engl. *Challenge/Response*), wie es bspw. bei Wachposten o.ä. eingesetzt wird. Es wird ein Anruf festgelegt, auf den mit einer bestimmten Parole zu antworten ist.

S/Key generiert hierzu eine beliebige Anzahl von Passwort-Paaren. Dies muss in einer gesicherten Umgebung geschehen, bspw. an der Konsole, einem Terminal oder in einem SSH-Login.

Zuerst benötigt man ein beliebiges neues Passwort, das als privater Schlüssel im S/Key-Prozess fungiert. Dieses Passwort ist das Herz des gesamten S/Key-Prozesses und darf daher nicht kompromittiert werden. Mit diesem Passwort wird der S/Key-Prozess mit skeyinit(1) wie in Abbildung 7.9 initialisiert.

Anschließend wird mit einem kryptographischen Hash (MD4, MD5 oder SHA1) eine Liste von Passwort-Paaren generiert. Das erste Passwort wird dazu auf dem Server gespeichert, alle anderen Hashes und Passwörter werden verworfen. Diese Liste kann man mit skey(1) anzeigen lassen. Die angezeigten Passwortpaare muss

man ausdrucken oder aufschreiben und kann sie so später für ungesicherte Logins verwenden. Im Beispiel wird zuerst mit Sha1 der S/Key-Prozess initialisiert und anschließend werden 10 Passwörter, von 99 an abwärtsgezählt, angezeigt.

```
# skeyinit -t sha1
Password: (NetBSD-Passwort eingeben)
[Updating stefan]
Old key: [sha1] balm15624
Reminder - Only use this method if you are directly connected
           or have an encrypted channel. If you are using telnet
           or rlogin, exit with no password and use skeyinit -s.

Enter secret password: (S/Key-Passwort eingeben)
Again secret password: (S/Key-Passwort eingeben)

ID stefan skey is otp-sha1 99 balm15625
Next login password: GIVE HEAL KITE BALD SIR GILT

$ skey -n 10 99 balm15625
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password: (S/Key-Passwort eingeben)
 90: COMA AM RIG CARE WAGE RUNG
 91: SAYS DUNK SHIM MINK RON IDA
 92: DUSK MEG DISK SMOG COWL GUT
 93: SOON AIDA SLOW BEAK RAW TOY
 94: JIBE RACE CROW GIFT TWO AWE
 95: CLOD WAD TOTE DEER JAM JAW
 96: WARD BERT WIT VALE HAP JAKE
 97: CLOT MOVE CUBE TEAM TO ARID
 98: RUNG SELL CHOW TEAR CAM PIE
 99: BARK ROWE MAID VIEW ABE SEN
$
```

Abbildung 7.9.: S/Key-Schlüssel erstellen

7.9 OPENSSH

OpenSSH ist das Mittel der Wahl, wenn es um verschlüßelte Logins auf entfernte Rechner geht. Mit OpenSSH kann man neben Passwörtern auch kryptographische Schlüssel zur Authentifikation verwenden. Die Schlüssel-Datei kann mit einem Passwort gesichert werden, muss sie aber nicht. So kann man Loginmöglichkeiten ohne Passworteingabe bereitstellen , die bspw. in Skripten verwendet werden. Hierzu muss

auf dem Client ein Schlüssel erstellt werden, dessen öffentlicher Teil auf den Server übertragen wird. Dabei kann man verschiedene Schlüssel-Typen verwenden, RSA1 für RSA im Protokoll 1 und RSA oder DSA für Protokoll 2. Ich empfehle den Einsatz von DSA-Schlüsseln der Länge 1024. RSA mit 1024 Bit sollte nicht mehr als sicher betrachtet werden, 2048 Bit sind empfehlenswert, so man denn RSA überhaupt noch einsetzen muß.

```
stefan@local $ ssh-keygen -b 1024 -t dsa
stefan@local $ scp .ssh/id_dsa.pub stefan@remote:/home/stefan/.ssh
stefan@local $ ssh stefan@remote
stefan@remote $ cd .ssh && cat id_dsa.pub » authorized_keys
stefan@remote $ chmod 600 authorized_keys ; rm id_dsa.pub
```

Abbildung 7.10.: SSH mit Schlüsselauthentifikation

7.9.1 *SSH-Logins verbieten*

Generell sollten SSH-Logins für sicherheitskritische Benutzer verboten werden. OpenSSH unterstützt dazu verschiedene Optionen in den Konfigurationsdateien[11]. Mit der Option `PermitRootLogin no` wird standardmäßig das Login für den Benutzer Root deaktiviert – es sollte unter keinen Umständen reaktiviert werden. Weiterhin kann man mit den Optionen »AllowUsers«, »DenyUsers«, »AllowGroups« und »DenyGroups« einstellen welche Benutzer und Gruppen sich (nicht) einloggen dürfen. Davon sollte man Gebrauch machen und wenn möglich unerwünschte Benutzer explizit aussperren. Man kann dabei auch explizit Hostnamen übergeben, so das sich ein Operator bspw. nicht von einem Windows-Rechner aus einloggen kann. Außerdem kann mit der Option »PermitEmptyPasswords« ein leeres Passwort verboten werden.

Wenn man die Konfiguration von OpenSSH geändert hat, muß man unbedingt die aktuelle Verbindung offen halten und sich ein weiteres mal einloggen. Dadurch überprüft man ob die Konfiguration funktioniert und sperrt sich nicht unbeabsichtigterweise selbst aus.

7.9.2 *Verschlüsselte Tunnel*

OpenSSH kann dazu verwandt werden, verschlüsselte Netzwerktunnel zu entfernten Rechnern zu erstellen. Dies ist nützlich, wenn ungesicherte Verbindungen (SMTP,

[11] `/etc/ssh/sshd_config` für OpenSSH aus dem Basissystem bzw. `/usr/pkg/etc/ssh/sshd_config` für die Pkgsrc-Variante.

POP, NNTP, IMAP oder ähnliches) durch nicht-vertrauenswürdige Netze geleitet werden. Im Beispiel 11.2 werden zwei Beispiele zu SSH-Tunneln gezeigt, das erste leitet den entfernten Port 5432 von 192.168.2.5 auf den lokalen Port 5000 weiter. Das heißt, SSH loggt sich erst auf 192.168.2.1 als stefan ein, und erstellt dann die verschlüsselte Verbindung vom lokalen Rechner zu 192.168.2.5 über die 192.168.2.1. Dabei ist nur die Strecke vom lokalen Rechner zur 192.168.2.1 mit SSH gesichert, der Teil von 192.168.2.1 zur 192.168.2.5 ist nicht verschlüsselt. Das zweite Beispiel loggt sich direkt auf der 192.168.2.5 ein leitet den dortigen 127.0.0.1:5432 direkt auf den lokalen Rechner weiter.

```
$ ssh -2 -c aes128-cbc -C stefan@192.168.2.1
                  -L 50000:192.168.2.5:5432

$ ssh -2 -c aes128-cbc -C stefan@192.168.2.5
                    -L 50000:127.0.0.1:5432

$ psql -U stefan -h 127.0.0.1 -p 50000 MyDatabase
```

Abbildung 7.11.: PostgreSQL-Ports mit SSH tunneln

8

SYSTEMAUFRUFE MIT SYSTRACE REGLEMENTIEREN

Narrensichere Systeme sind meist nicht von Narren geprüft.

(Erhard Blanck)

8.1 SYSTEMAUFRUFE MIT SYSTRACE STEUERN

Wie inzwischen allgemein anerkannt ist, setzt eine erfolgreiche Verteidigung eines Rechner-Systems mehrere Verteidigungslinien voraus. Diese Linien müssen sich überlappen ohne jedoch voneinander abzuhängen. Dies können beispielsweise Paketfilter, Application-Level-Gateways oder biometrische Zugangskontrollen sein. Allerdings bietet jede Verteidigungslinie wiederum neue Angriffspunkte, so könnte beispielsweise ein Einbruchserkennungssystem mittels Speicherüberlauf übernommen und mit Root-Rechten mißbraucht werden. Daher ist es notwendig, die Schadmöglichkeiten von Programmen einzugrenzen.

Nahezu jedes heute eingesetzte Programm ist zu komplex und umfangreich, um sorgfältig auf Fehlerquellen und Sicherheitslücken überprüft zu werden. Selbst wenn es als quelloffenes Programm vorliegt, überprüft in der Regel niemand den Quellcode auf absichtliche oder fahrlässige Sicherheitsprobleme. Quelloffenheit ist zwar ein sehr gutes Kriterium für sicherheitsrelevante Programme, schützt aber definitiv nicht vor Lücken oder Hintertüren, wie (1984) beweist. Teilweise ist man aber auch auf den Einsatz geschlossener Software angewiesen und hat dann keinerlei Möglichkeit mehr, diese zu prüfen und muss ihr zwangsläufig trauen.

Angriffe gegen Systeme konzentrieren sich in der Regel auf Systemaufrufe (auch System Calls oder Syscalls genannt), die unter anderem auch dazu verwendet werden können um im Kernel privilegierte Operationen durchzuführen. Um derartigen Vorgehensweisen zu begegnen, wird die Angriffsfläche reduziert. Dazu wird ein Programm eingeführt, das die Ausführung von Systemaufrufen begrenzt und die Zugriffsrechte darauf feiner als bisher granuliert. Hierzu benötigt man eine Richtlinie, die die Zugriffe reglementiert. Die Richtlinie muss alle möglichen Fälle abdecken

und auch Kenntnisse von allen möglicherweise auftretenden Pfaden haben, was Dank symbolischer Links nicht besonders einfach ist. Die Richtlinie beschreibt das Normalverhalten eines Prozesses und dient so als Vergleich zum laufenden System. Weicht ein Prozess von der beschriebenen Richtlinie ab, wird dies als Einbruchsversuch erkannt und verhindert. Außerdem soll das überwachende Programm auch Überwachungsprotokolle der überwachtenProgramme erzeugen, um so Anomalien erkennen und analysieren zu können.

Ein derartiges System lässt sich auf verschiedene Arten implementieren. Einerseits kann es komplett im Kernel arbeiten, andererseits auch komplett im User-Space. Im Kernel ist die Ausführung recht schnell, aber das System selbst ist äußerst komplex und nur sehr schwer auf andere Betriebssysteme zu portieren. Im User-Space hingegen ist das Programm portabel, aber sehr langsam und unsicher, da es zu Race-Conditions zwischen dem Zeitpunkt der Analyse und der Ausführung eines Systemaufrufes kommen kann.

Systrace, vom seinem Entwickler Niels Provos in (2006) und (Eriksen und Provos 2003) beschrieben, verwendet einen hybriden Ansatz. Es wird ein kleiner Teil im Kernel implementiert und der größte Teil im User-Space. Der Kernel-Teil ermöglicht die sehr schnelle Behandlung von kontextinsensitiven Systemaufrufen, die bspw. stets abgelehnt oder erlaubt werden. Weiterhin ermöglicht dies, auszuführende Programme in einem Sandkasten zu kapseln und geforkte Prozesse mit der vererbten Richtlinie weiter zu kontrollieren. Der Teil im User-Space überwacht die Systemaufrufe auf kontextsensitive Entscheidungen und trifft sie anhand der definierten Richtlinie. Während der Entscheidungsfindung blockiert der Kernel den fraglichen Prozess. Weiterhin kann der User-Space-Dæmon über eine Schnittstelle Informationen wie Zustandsübergänge, PID-Änderungen oder Forks vom Kernel-Teil anfordern.

Um Funktionen des Kernels zu nutzen, können Anwenderprogramme Systemaufrufe verwenden. Mit diesen kann sich ein Programm bspw. an einen Port binden oder eine Logdatei öffnen, da solche Operationen System-Rechte erfordern. Damit ein Programm auf derartige Systemaufrufe zugreifen darf, muß es mit Root-Rechten gestartet werden - und eröffnet damit eine riesige Sicherheitslücke. Systrace von Niels Provos umgeht dieses Problem, indem die Rechtezuteilung nicht mehr auf Programmebene vorgenommen wird, sondern auf Ebene der Systemaufrufe heruntergebrochen wird. Ein Programm kann also von einem normalen Benutzer gestartet werden und bekommt gemäß einer vorher erstellten Richtlinie von Systrace entsprechende Zugriffsrechte auf Systemaufrufe zugeteilt.

8.1.1 *Die Grammatik der Richtlinie*

Die Richtlinie für ein zu überwachendes Programm wird in einer recht einfachen Grammatik definiert, in der Ausdrücke aneinandergereiht werden. Ein Ausdruck besteht dabei aus einem booleschen Ausdruck und der auszuführenden Aktion. Die Aktion kann aus folgenden Aktionen bestehen: *ask* (frage), *deny* (verbiete) oder *permit* (erlaube) in Verbindung mit optionalen Argumenten. Ergibt der boolesche Ausdruck *wahr*, wird die definierte Aktion ausgeführt. Ist als Aktion *ask* definiert, wird der Benutzer befragt, um die Aktion zu erlauben oder zu verbieten.

Der boolesche Ausdruck setzt sich aus verschiedenen Variablen und den Logik-Operatoren *and* (logisches Und), *or* (logisches Oder) und *not* (logisches Nicht) zusammen. Die Variablen bestehen aus den normalisierten Systemaufruf-Namen, den dem Systemaufruf übergebenen Argumenten und einem Logik-Operator, der beide Argumente verknüpft.

Die Filterausdrücke auf Argumente verwenden verschiedene Operatoren:

- *match* Wahr, wenn der Dateiname gemäß den Regeln in fnmatch(3) getroffen wird.
- *eq* Wahr, wenn das Argument des Systemaufrufes genau der Vorgabe entspricht.
- *neq* Die logische Negation des *eq*-Operators.
- *sub* Wahr, wenn die angegebene Teilzeichenkette im Argument des Systemaufrufes vorkommt.
- *nsub* Die logische Negation des *sub*-Operators.
- *inpath* Wahr, wenn das Argument des Systemaufrufes im Pfad der Vorgabe vorkommt.
- *re* Wahr, wenn das Argument des Systemaufrufes dem angegebenen Regulären Ausdruck entspricht.

Ein paar Ausdrücke zur Veranschaulichung der Möglichkeiten:

- `netbsd-execve: permit` Erlaubt alle execve(2)-Aufrufe.
- `netbsd-execve: true then permit log` Erlaubt alle execve(2)-Aufrufe und protokolliert sie an Syslog. Das »true then« ist nötig, damit »log« eingesetzt werden kann.

- `netbsd-fsread: filename eq ''/etc/passwd'' then permit` Erlaubt alle Lese-Operationen auf /etc/passwd.

- `netbsd-fsread: filename match ''/etc*'' then deny [eaccess] log` Verbietet alle Lese-Operationen auf /etc* mit dem Fehlercode EACCESS und protokolliert sie an Syslog.

- `netbsd-seteuid: uid eq ''1007'' or uname eq ''systraced'' then permit` Erlaubt das setzen der effektiven UID wenn die UID des aufrufenden Benutzer 1007 ist oder er der Benutzer »systraced« ist.

- `netbsd-connect: sockaddr re ''inet-.192\.168\.[0,4,8]\.[4-6].:22'' then permit log` Erlaubt eine Socket-Verbindung, wenn die Zieladresse des Sockets im angegebenen Adressbereich liegt. Auch dieser Aufruf wird protokolliert.

Um über einen Systemaufruf zu entscheiden, traversiert Systrace alle Ausdrücke und bricht beim ersten Ausdruck ab, der zum Systemaufruf passt. Dieser Ausdruck entscheidet dann, ob der Systemaufruf ausgeführt oder abgelehnt wird. Wird kein passender Ausdruck gefunden, wird die Entscheidung an den Benutzer delegiert. Wird ein Systemaufruf abgelehnt, kann Systrace an das aufrufende Programm einen spezifizierten Fehlercode zurückgeben.

Um die Richtlinie auf Benutzer- bzw. Gruppenebene granulieren zu können, werden Ausdrücke mit einem Prädikat versehen. Dieses Prädikat genügt der Form *", if" {"user", "group"} {"=", "!=", "<", ">" } {Bernutzername, numerische UID}*. Somit lassen sich Ausdrücke der Art `netbsd-fsread: filename eq ''/etc/master.passwd'' then deny[eperm], if group != wheel` erzeugen. Hier wird der Zugriff auf die Datei /etc/master.passwd mit dem Fehlercode EPERM abgelehnt, wenn der aufrufende Benutzer nicht Mitglied der Gruppe »wheel« ist.

Soll ein Systemaufruf unter anderen Benutzerrechten ausgeführt werden, kann an den Ausdruck die Direktive *as user:group* angehängt werden. Beispiele dazu werden im Kapitel 8.4 aufgeführt.

An jeden Ausdruck einer Richtlinie kann die Log-Direktive `log` angehängt werden. Damit wird der Ausdruck und die durch ihn implizierte Entscheidung vom Betriebssystem geloggt. Mit dieser Option lassen sich Programme komplett überwachen und analysieren. Protokolliert man beispielsweise alle exec(3)-, execve(2)- und connect(2)-Aufrufe, erfährt man welche Programme ein Benutzer ausgeführt und welche Sockets er geöffnet hat. Beachten Sie hierbei aber unbedingt Datenschutzrechtliche Bestimmungen und andere Regelungen.

8.1.2 *Erzeugung einer Richtlinie*

Ziel der Richtlinie ist es, alle erlaubten Systemaufrufe einer Anwendung zu erfassen und zu erlauben. Nicht erfasste Aufrufe sind als Angriff zu werten und zu verbieten. Somit lässt sich eine Richtlinie erstellen, in dem ein spezielles Programm die zu erfassende Anwendung bei einem Probelauf überwacht und die abgesetzten Systemaufrufe mitschneidet. Die ausgeführten Systemaufrufe werden dabei in die kanonische Form normalisiert und in Richtlinien-Ausdrücke übersetzt. Existiert in der bisherigen Richtlinie kein Ausdruck, der den aktuellen Systemaufruf behandelt, wird ein neuer Ausdruck angehängt, der den Aufruf erlaubt. Manuelles Eingreifen in die erzeugte Richtlinie ist in der Regel nicht erforderlich, es sei denn, die überwachte Anwendung arbeitet mit Zufallsnamen für Dateien. Dann muss der entsprechende Ausdruck dahingehend abgeändert werden. Bei dieser automatisierten Richtlinienerstellung wird davon ausgegangen, daß das zu überwachende Programm per sè sicher ist. Kann dies nicht gewährleistet werden, ist eine so erstellte Richtlinie nicht als sicher zu betrachten. Außerdem ist die Automatik ebenfalls nicht anwendbar, wenn es nicht möglich ist, alle auftretenden Code-Pfade durchzuexerzieren. Trotzdem dient eine so erstellte Richtlinie als Basis für eine händische Anpassung, oder als Basis für weitere Übungsläufe. Bei diesen wird die Richtlinie nur dann erweitert, wenn neue Systemaufrufe erfolgen. Hier kann der Benutzer wieder die resultierende Aktion festlegen. Nachdem eine Richtlinie fertiggestellt wurde, kann sie von Systrace gegenüber dem gewünschten Programm durchgesetzt werden.

8.2 IMPLEMENTIERUNG

Systrace kann in einem von drei verschiedenen Modi laufen:

- Initialisierungsmodus: Systrace überwacht ein Programm automatisch und generiert eine Richtlinie. Diese ist ein guter Startpunkt um eine angepaßte und verfeinerte Richtlinie zu erzeugen.

- Nachfragemodus: Hier wird ebenfalls ein Programm überwacht und eine Richtlinie erzeugt, allerdings wird der Benutzer bei jedem Systemaufruf um Zustimmung gebeten. Dies ist sinnvoll, wenn dem zu überwachenden Programm nicht unbedingt von vornherein vertraut werden kann. Die Nachfrage erfolgt über das X-Programm xsystrace(1) oder im Textmodus ohne X.

- Überwachungsmodus: Systrace überwacht ein Programm und setzt die definierte Richtlinie durch. Nicht erlaubte Systemaufrufe werden abgelehnt und protokolliert.

Systrace verfügt über eine Reihe von Optionen:

- `-A` Initialisierungsmodus, erzeugt eine Richtlinie, in der alle Systemaufrufe erlaubt sind.
- `-a` Überwachungsmodus, setzt die definierte Richtline durch.
- `-c UID:GID` Spezifiziert eine numerische User- und Gruppen-ID. Unter diesen wird das zu überwachende Programm ausgeführt. Nur als Root machbar.
- `-d Verzeichnis` Setzt ein Verzeichnis für die Richtliniendateien. Standard ist /.systrace.
- `-f Datei` Systrace verwendet die Richtlinie, die in der Datei angegeben ist.
- `-g gui` Aktiviert eine alternative GUI.
- `-i` Vererbt die Richtlinie des Elternprozesses an die Kinder.
- `-p` Systrace bindet sich an den bereits laufenden Prozess mit der angegebenen PID. Der komplette Programmpfad muß ebenfalls angegeben werden.
- `-t` Textmodus, läuft auch ohne X.
- `-U` Benutzt nur globale Richtlinien (/etc/systrace) statt lokaler.
- `-u` Deaktiviert das Zusammenfassen von Systemaufrufen zu Aliasen.

8.3 Sicherheit des Systems

Ein System wie Systrace selbst hat auch mit einigen Sicherheitsproblemen zu kämpfen, z. B. Aliasen auf Systemressourcen, Dateinamen von Programmen, die in einem Chroot laufen oder der Verfolgung von Prozess-IDs.

Der Zugriff auf ein und dieselbe Datei ist unter Unix dank symbolischer Links und relativer Pfadnamen auf unendlich viele Arten möglich. Außerdem können Dateien von verschiedenen Diensten, wie z. B. Proxies, NFS oder CFS, bereitgestellt werden. Derartige Dienste sind für Systrace nicht sichtbar, müssen aber trotzdem korrekt funktionieren.

Weiterhin ist es möglich, eine Race-Condition zu produzieren, die eine Systrace-Sandkiste aushebelt. Systrace benötigt für die Überprüfung eines Systemaufrufes

eine gewisse Zeitspanne. Während dieser Zeitspanne kann ein anderer Prozess den eigentlichen Systemaufruf des zu überwachenden Programmes ändern. Systrace erkennt diese Änderung nicht und gestattet die Ausführung des inzwischen geänderten Systemaufrufes.

Um diesen Problemen zu begegnen, verwendet Systrace nur normalisierte Dateinamen und Systemaufrufe. Alle Dateinamen und Parameter für Systemaufrufe werden von Systrace normalisiert, indem Dateinamen in absolute Form – also ohne Symlinks oder relative Pfadangaben – umgewandelt werden. Diese normalisierten Werte werden dem Betriebssystem wieder übergeben. Ausgenommen hiervon sind nur einige Systemaufrufe, wie `readlink`. Weiterhin werden diese Werte auf einem nur-lesbaren Puffer zwischengespeichert, sodaß kein anderer Prozess die Werte manipulieren kann. Der Kernel verweigert die Ausführung von Systemaufrufen, die Symlinks als Argumente enthalten. Somit werden nur noch von Systrace normalisierteAufrufe, die erlaubt sind, ausgeführt. Alle anderen Aufrufe werden abgelehnt.

Werden Systemaufrufe abgelehnt, müssen die überwachten Programme entsprechende Fehlermeldungen bekommen. Da nicht alle Programme eine funktionierende Fehlerbehandlung implementieren, kann in der Richtlinie ein bestimmter Fehlercode spezifiziert werden.

Ebenfalls zu betrachten ist ein Richtlinien-Wechsel und Prozess-Beendigung. Wenn ein überwachter Prozess einen neuen Prozess startet, wird der alte Prozess vom System aus dem Speicher entfernt. Der neue Prozess wird stattdessen ausgeführt. Dieser neue Prozess kann ein vertrauenswürdiges Programm sein, sodaß keine weitere Überwachung notwendig ist. Es kann aber auch ein Prozess sein, der mit einer anderen Richtlinie besser überwacht wird. Systrace überwacht Systemaufrufe auf Erfolg und kann so nach einem geglückten `execve`-Systemaufruf den neuen Prozess mit einer anderen Richtlinie überwachen oder die Überwachung beenden.

Mithilfe des `log`-Befehls kann in einer Richtlinie jeder ausgeführte Systemaufruf protokolliert werden. Somit ist es möglich, mit einer Systrace-überwachten Shell alle `execve`-Aufrufe eines Benutzers zu überwachen. Hierbei muss man aber unbedingt datenschutzrechtliche und andere rechtliche Regelungen beachten.

Die gesamten Richtlinien werden in einzelnen Dateien gespeichert. Kann ein Einbrecher die Richtlinien-Dateien manipulieren, kann er Systrace aushebeln. Daher sind die Richtlinien-Dateien unbedingt zu schützen. Dazu kann man neben restriktiven Schreibrechten die NetBSD-Fileflags (`schg`) in Verbindung mit den Security-Leveln verwenden. Möchte man diese Methode nicht einsetzen, sollten die Richtlinien zumindest regelmäßig mit einem Integritätsprüfer wie mtree(8), Aide oder Tripwire überprüft werden.

8.4 Apache überwachen

Mit den Befehlen aus Abbildung 8.2 wird Apache gestartet. Dabei wird er von Systrace überwacht, sodaß eine Richtlinie erstellt wird. Diese Richtlinie wird geschrieben, nachdem Apache wieder beendet wurde. Ein Teil der Richtlinie wird in Abbildung 8.3 gezeigt. In den Zeilen 11, 13 und 37 wird auf eine Datei im Verzeichnis /var/run zugegriffen. Wie man leicht erkennt, enthält der Dateiname die Prozess-ID bzw. Semaphoren-Nummern. Diese müssen durch den Joker »*« ersetzt werden, außerdem muß der Operator »eq« durch »match« ersetzt werden, wie in Abbildung 8.4 gezeigt wird. In den Zeilen 31 und 32 bindet sich Apache an die angegebenen IP-Adressen und Port 80. In den restlichen sowie ausgelassenen Zeilen liest Apache diverse Konfigurationsdateien ein. Startet man nun Apache unter Systrace-Überwachung, kann kein Benutzer auf die Webseiten zugreifen, da die Richtlinie keinen Lese-Zugriff für /home/www beinhaltet. Die Verstöße werden in /var/log/messages protokolliert. Um dieses Problem möglichst komfortabel zu beheben, wird Apache wieder im Initialisierungsmodus von Systrace gestartet: `systrace apachectl start`. Wird nun in einem Browser die Adresse http://127.0.0.1/ aufgerufen, meldet sich xsystrace (Ein Beispiel findet sich in Abbildung 8.1) zu Wort und verlangt vom Benutzer die Bestätigung oder Verweigerung der benötigten Systemaufrufe.

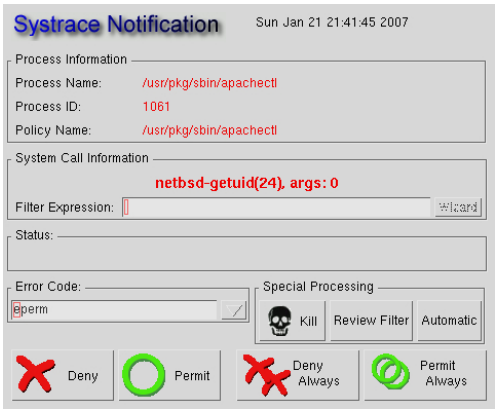


Abbildung 8.1.: Xsystrace bittet den Benutzer um eine Entscheidung

Da diese Prozedur für jede einzelne von Apache geladene Datei durchlaufen wird, brechen wir nach dem Laden der Index-Seite ab und beenden Apache mit `apachectl stop`. Systrace hat die Richtlinie bereits um die Lesezugriffe auf /home/www erweitert. Allerdings für jede Datei einzeln, sodaß hier wieder Reguläre Ausdrücke mit »match« und »*« eingesetzt werden. Abbildung 8.5 zeigt die automatisch generierte Richtlinie, die jede aufgerufene Datei einzeln behandelt. In Abbildung 8.6 wurde die Richtlinie mit Regulären Ausdrücken verfeinert, sodaß Zugriffe auf /home/www/public/* gestattet und auf /home/www/institutsintern/* verboten wer-

den. Weiterhin wurden die Zugriffe auf CGI-Dateien erlaubt bzw. verboten. Zugriffe auf die verbotenen Dateien werden mit ENOENT abgelehnt, so das für Apache die Dateien nicht existieren. Bis jetzt wurde die Richtlinie soweit konfiguriert, das Apache unter Überwachung normal funktioniert.

Trotzdem muß Apache noch als Root gestartet werden. Systrace kann mit der Option `-c` eine beliebige numerische Benutzer- und Gruppen-ID übernehmen, als die der zu überwachende Prozess ausgeführt werden soll. Dazu muss allerdings noch die Richtlinie angepasst werden, da ein nicht-privilegierter Benutzer keine privilegierten Operationen ausführen darf. Systrace kann jede Aktion in der Richtlinie als ein anderer Benutzer ausführen. Somit kann eine Anwendung als normaler Benutzer ausgeführt werden, und nur die benötigten Systemaufrufe werden als Root ausgeführt. Dann müssen Systrace und das zu überwachende Programm aber als Root gestartet werden. Startet man Apache nun mit `systrace -c 1002:1001 apachectl start` als Benutzer und Gruppe »www« mit der bisherigen Richtlinie, wird sich Systrace erneut zu Wort melden. Es werden alle Systemaufrufe angezeigt, die als Benutzer »www« nicht ausgeführt werden können. Am einfachsten editiert man daher vorher die Richtlinie mit vi(1) und sucht zuerst nach Aktionen in denen Dateien mit »netbsd-fswrite« geschrieben werden. Das umfaßt in diesem Beispiel die Log-, PID- und Semaphoren-Dateien. Die nächsten offensichtlichen Kandidaten sind alle »netbsd-bind«-Aktionen, in denen sich Apache an die Netzwerkgeräte bindet. Die angepaßten Aktionen der Richtlinie für nicht-privilegierte Läufe finden Sie in Abbildung 8.4. Mit dieser Richtline kann Apache via Systrace als Benutzer »www« gestartet werden.

```
# systrace -A apachectl start
/usr/pkg/sbin/apachectl start: httpd started
# apachectl stop
/usr/pkg/sbin/apachectl stop: httpd stopped
# ls /root/.systrace/
usr_pkg_sbin_apachectl usr_pkg_sbin_httpd
```

Abbildung 8.2.: Automatisch eine Richtline für Apache erstellen

```
[...]
netbsd-pread: permit
netbsd-fsread: filename eq "/etc/group" then permit
netbsd-fsread: filename eq "/usr/pkg/etc/httpd/httpd.conf" then permit
netbsd-fsread: filename eq "/usr/pkg" then permit
netbsd-fsread: filename eq "/usr/pkg/etc/httpd/srm.conf" then permit
netbsd-fsread: filename eq "/usr/pkg/etc/httpd/access.conf" then permit
netbsd-gettimeofday: permit
netbsd-fsread: filename eq "/etc/etc.network/resolv.conf" then permit
netbsd-fsread: filename eq "/etc/hosts" then permit
netbsd-chmod: filename \underline{eq}"/var/run/httpd.mm.\underline{2872}.sem"
                          and mode eq "600" then permit
netbsd-chown: filename \underline{eq}"/var/run/httpd.mm.\underline{2872}.sem"
                          and uid eq "1002" and gid eq "-1" then permit
netbsd-fswrite: filename eq "/var/log/httpd/error_log" then permit
netbsd-dup2: permit
netbsd-select: permit
netbsd-fsread: filename eq "/usr/pkg/etc/httpd/mime.types" then permit
netbsd-fsread: filename eq "/usr/pkg/etc/httpd/magic" then permit
netbsd-fswrite: filename eq "/var/log/httpd/access_log" then permit
netbsd-chdir: filename eq "/" then permit
netbsd-fork: permit
netbsd-exit: permit
netbsd-setsid: permit
netbsd-fsread: filename eq "/dev/null" then permit
netbsd-fswrite: filename eq "/dev/null" then permit
netbsd-socket: sockdom eq "AF_INET" and socktype eq "SOCK_STREAM" then permit
netbsd-setsockopt: permit
netbsd-bind: sockaddr eq "inet-[0.0.0.0]:80" then permit
netbsd-listen: permit
netbsd-bind: sockaddr eq "inet-[192.168.0.5]:80" then permit
netbsd-bind: sockaddr eq "inet-[127.0.0.1]:80" then permit
netbsd-fsread: filename eq "/var/run/httpd.pid" then permit
netbsd-umask: permit
netbsd-fswrite: filename eq "/var/run/httpd.pid" then permit
netbsd-write: permit
netbsd-fswrite: filename \underline{eq}"/var/run/httpd.lock.\underline{1827}" then per
[...]
```

Abbildung 8.3.: Automatisch erzeugte Richtlinie für Apache (Auszug)

```
[...]
netbsd-chmod: filename \underline{match}"/var/run/httpd.mm.\underline{*}.sem"
                          and mode eq "600" then permit
netbsd-chown: filename \underline{match}"/var/run/httpd.mm.\underline{*}.sem"
                          and uid eq "1002" and gid eq "-1" then permit
netbsd-fswrite: filename \underline{match}"/var/log/httpd/error_log" then permit
[...]
netbsd-fswrite: filename \underline{match}"/var/run/httpd.lock.\underline{*}" then permit
[...]
```

Abbildung 8.4.: Automatisch erzeugte Richtlinie korrigiert für Apache (Auszug)


```
[...]
netbsd-fsread: filename eq "/home/www" then permit
netbsd-fsread: filename eq "/home/www/.htaccess" then permit
netbsd-fsread: filename eq "/home/www/index.html" then permit
netbsd-fsread: filename eq "/home/www/index.pl" then permit
netbsd-fsread: filename eq "/home/www/index.mhtml" then permit
[...]
```

Abbildung 8.5.: Automatisch generierte Richtlinie für Apache, die jede Datei einzeln aufführt.

```
[...]
netbsd-fsread: filename match "/home/www/public/*" then permit
netbsd-fsread: filename match "/home/www/institutsintern/*"
                             then deny [enoent]
netbsd-fsread: filename eq "/home/www/cgi-bin/cvsweb.cgi" then permit
netbsd-fsread: filename eq "/home/www/cgi-bin/postgresql.cgi"
                              then deny [enoent]
[...]
```

Abbildung 8.6.: Angepasste Apache-Richtline

```
# grep 'as root' usr_pkg_sbin_httpd
  netbsd-fswrite: filename match "/var/run/httpd.mm.*.sem" then permit as root
  netbsd-fswrite: filename match "/var/run/httpd.mm.*.sem" then permit as root
  netbsd-fswrite: filename match "/var/run/httpd.mm.*.sem" then permit as root
  netbsd-chmod: filename match "/var/run/httpd.mm.*.sem" and mode eq "600"
                                                          then permit as root
  netbsd-chown: filename match "/var/run/httpd.mm.*.sem" and uid eq "1002"
                                        and gid eq "-1" then permit as root
  netbsd-fswrite: filename eq "/var/log/httpd/error_log" then permit as root
  netbsd-fswrite: filename eq "/var/log/httpd/access_log" then permit as root
  netbsd-bind: sockaddr eq "inet-[0.0.0.0]:80" then permit as root
  netbsd-bind: sockaddr eq "inet-[192.168.0.5]:80" then permit as root
  netbsd-bind: sockaddr eq "inet-[127.0.0.1]:80" then permit as root
  netbsd-fsread: filename match "/var/run/httpd.pid" then permit as root
  netbsd-fswrite: filename eq "/var/run/httpd.pid" then permit as root
  netbsd-fswrite: filename match "/var/run/httpd.lock.*" then permit as root
```

Abbildung 8.7.: Angepasste Apache-Richtline

8.5 SYSTRACE-ÜBERWACHTE SHELL

Matthias Petermann beschreibt in (2005) die Einrichtung einer Shell, die von Systrace komplett überwacht wird.

Um eine Shell von Systrace überwachen zu lassen, müsste die Shell über Systrace in der Art `systrace -a -i ksh` gestartet werden. Da solch ein Befehl nicht in der /etc/master.passwd eingetragen werden kann, muß er in einem kleinen C-Programm (Abbildung 8.8) gekapselt werden. Diese Kapsel wird nach Erstellung der Richtlinie in /etc/shells und /etc/master.passwd für die jeweils zu überwachenden Benutzer als Shell eingetragen. In diesem Beispiel soll der Benutzer *systraced* eine eingeschränkte Korn-Shell erhalten.

Um die Richtlinie automatisch zu erstellen, loggen wir uns mit `login systraced` ein und starten eine Shell mit `systrace -A /bin/ksh`. Nach Beendigung der Shell haben wir wieder das Grundgerüst unserer Richtlinie in systraced/.systrace/bin_ksh vorliegen. Nun wird die Shell-Kapsel aus Abbildung 8.8 als Shell für den Benutzer aktiviert und wieder mit login(1) als *systraced* eingeloggt. Da nun die Systrace-Überwachung aktiv ist, werden Systemaufrufe überwacht und Fehler via syslog(3) protokolliert. Es ist in der Anfangsphase recht praktisch, in einem weiteren X-Terminal `tail -f /var/log/messages` laufen zu lassen, um so in Echtzeit die fehlgeschlagenen Systemaufrufe analysieren zu können. Die Richtlinien unter systraced/.systrace/ sind mit chown(8) Root und Wheel zuzuordnen und mit chmod(1) auf 644 oder gar 444 zu setzen. Selbiges gilt für das .systrace-Verzeichnis, allerdings mit den Rechten 755.

In der Beispiel-Richtlinie aus Abbildung 8.9 befinden sich vier Blöcke von Ausdrücken. Im ersten Block wird der Zugriff auf verschiedene Geräte und Konfigurationsdateien geregelt. Außerdem wird der Zugriff auf /tmp/, /var/tmp/ und das Heimatverzeichnis des Benutzers geregelt.

Im zweiten Block sind verschiedene Ausdrücke, die die Erstellung von Sockets regeln. Dabei werden mit den Regulären Ausdrücken die Zugriffe auf verschiedene IP-Adressen bzw. Adress-Bereiche unter Port 22 erlaubt. Alle derartige Ausdrücke werden protokolliert.

Im dritten Block werden die exec(3)- und execve(2)-Aufrufe behandelt. Aufrufe von /usr/bin/ftp und /usr/bin/telnet sowie von Programmen, die in /home/systraced/, liegen werden verboten, alle anderen erlaubt. Der Benutzer kann somit auch keine Programme in seinem Benutzerverzeichnis ablegen und von dort aus starten. Er kann allerdings in dieser Richtlinie noch Programme in /tmp/ oder /var/tmp/ ausführen.

Im letzten Block finden sich alle Systemaufrufe, die ohne Argumente aufgeführt werden und von Systrace während des ersten Initialisierungslaufes geschrieben

wurden.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
  puts("Führe Systrace-überwachte Shell aus");
  execlp("/bin/systrace","systrace","-a","-i","/bin/ksh",NULL);
  return 0;
}
```

Abbildung 8.8.: Kapselung des Shell-Aufrufes

```
Policy: /bin/ksh, Emulation: netbsd
\textit{## Zugriffe auf Konfigurationsdateien erlauben}
netbsd-fsread: filename eq "/etc/man.conf" then permit
netbsd-fsread: filename eq "/etc/passwd" then permit
netbsd-fsread: filename match "/etc*" then deny
netbsd-fsread: filename match "/home/systraced/*" then permit
netbsd-fsread: filename match "/home*" then deny
netbsd-fsread: filename match "/tmp/*" then permit
netbsd-fsread: filename match "/var/tmp/*" then permit
netbsd-fswrite: filename match "/dev/tty" then permit
netbsd-fswrite: filename match "/tmp/*" then permit
netbsd-fswrite: filename match "/var/tmp/*" then permit
netbsd-fswrite: filename eq "/dev/crypto" then permit

\textit{## SSH auf bestimmte Adressen erlauben}
netbsd-socket: sockdom eq "AF_INET" and socktype match "*" then permit
netbsd-connect: sockaddr eq "inet-[127.0.0.1]:22" then permit log
netbsd-connect: sockaddr re "inet-.192\.168\.[0,4,8]\.[4-6].:22" then permit log
netbsd-connect: sockaddr re "inet-.192\.168\.0\.5\.:22" then permit log
netbsd-setuid: uid eq "1007" and uname eq "systraced" then permit
netbsd-seteuid: uid eq "1007" and uname eq "systraced" then permit
netbsd-recvfrom: permit
netbsd-setsockopt: permit

\textit{## Ausführbare Dateien in $HOME und ftp + telnet verbieten, sonst erlauben}
netbsd-exec: filename sub "/home/systraced" then deny log
netbsd-execve: filename sub "/home/systraced" then deny log
netbsd-execve: filename eq "/usr/bin/ftp" then permit log
netbsd-execve: filename eq "/usr/bin/telnet" then permit log
netbsd-exec: true then permit log
netbsd-execve: true then permit log

netbsd-mmap: permit
netbsd-fsread: permit
netbsd-__fstat13: permit
netbsd-close: permit

[...]
```

Abbildung 8.9.: Kapselung des Shell-Aufrufes

8.6 weiterer nutzen

Systrace eignet sich nicht nur zur Absicherung eines Systemes, sondern auch zur einfachen Überwachung und Analyse von Programmen. Indem man automatisch eine Richtlinie erstellen läßt, erfährt man welche Systemaufrufe vom Programm durchgeführt werden. Diese Richtlinie kann man zu Testzwecken manipulieren und so überprüfen wie sich ein Programm verhält, wenn es beispielsweise nicht mehr auf bestimmte Dateien zugreifen darf. Diese Vorgehensweise ist sehr nützlich bei großen Programmen, die nur als Binärversion verfügbar sind, wie Opera oder Acrobat Reader. Ebenso kann man damit selbst entwickelte Programme auf Fehlerbehandlungen hin überprüfen.

8.7 fazit

Systrace ist ein umfangreiches Programm, das bei korrekter Konfiguration die Sicherheit eines Systemes dramatisch erhöhen kann. Mit Systrace lassen sich Dienste als nicht-privilegierter Benutzer ausführen. Nur bestimmte Systemaufrufe müssen Root-Rechte erhalten. Damit ist das potentielle Schadensrisiko, das von einem SETUID-Programm ausgeht, dramatisch reduziert. Weiterhin ermöglicht Systrace eine feinere Granulierung von Programmaufrufen oder Sockets man kann beispielsweise SSH nur auf bestimmte IP-Adressen erlauben und auf alle anderen verbieten.

Die Konfiguration einer Systrace-Richtlinie setzt allerdings gute Kenntnisse des Betriebssystemes – insbesondere der Systemaufrufe – und der Anwendung voraus. Außerdem ist Systrace eine technische Maßnahme, die ohne umfassende Sicherheitsrichtlinie allein zu kurz greift.

siehe auch:
(Ioannidis, Bellovin und Smith 2006)
(Goldberg u. a. 1996)

9 DATEN SICHER LÖSCHEN

Das ist die Wunderwelt der Technik
Faszinierend doch sonderbar
Wir werden kleiner und kleiner
Und am Ende sind wir nicht mehr da.

(Welle: Erdball, »Wunderwelt der Technik«)

Die Hauptaufgabe eines Systemadiministrator ist Schutz und Verfügbarmachung von Daten. Doch Daten müssen teilweise wieder sicher vernichtet werden. Dies ist beispielsweise der Fall wenn alte Rechner oder Datenträger ausgesondert werden oder gesetzliche Aufbewahrungsfristen überschritten werden. In diesem Kapitel wird gezeigt welche Methoden zur sicheren Datenlöschung existieren, was bei den verschiedenen Datenträgern zu beachten ist und wie man die Datenvernichtung bereits im Vorfeld plant.

9.1 WIE WERDEN DATEN GESPEICHERT UND GELÖSCHT?

Um Daten vor unbefugtem Zugriff zu schützen, kann man kryptographische Methoden einsetzen. So kann man, wie in (Schumacher 2006c) beschrieben, CGD unter NetBSD einsetzen um Dateisysteme zu verschlüsseln. Aber Dateien müssen nicht nur aufbewahrt und gesichert, sondern auch gelöscht werden. Wenn es sich dabei um sensible Daten handelt, sind spezielle Löschverfahren notwendig, die die Daten unwiederbringlich Löschen.

Wie man Daten vernichtet hängt vom Speichermedium bzw. der verwendeten Speichertechnologie ab. Für elektromagnetische Datenträger (Festplatten, Disketten, Magnetbänder) gibt es drei Möglichkeiten, die physikalische Vernichtung mittels Curie-Temperatur oder Magnetfeld, oder die vernichtungsfreie Löschung der Daten durch Überschreiben.

Um diese Verfahren zu verstehen, muss man wissen wie elektromagnetische Datenträger prinzipiell funktionieren. Dateien werden als Bitfolge abgespeichert, die physikalisch auf der Festplatte durch magnetische Flußwechsel gespeichert werden. Magnete bzw. ferromagnetische Stoffe haben einige physikalische Eigenschaften, die

für dieses Thema relevant sind. Diese Grundlagen werden in jedem Einführungsbuch in die Elektrotechnik oder Nachrichtentechnik, wie beispielsweise (Steinbuch 1967, Kap. 2.3 *Magnetische Bauelemente*), (Lindner†, Bauer† und Lehmann 1999, Kap. 2 *Elektrische und Magnetische Felder*) oder dem für Informatiker bzw. Administratoren hervorragend geeignetem (Messmer 1998, Kapitel 30.1, *Grundlagen magnetischer Datenaufzeichnung – Ferromagnetismus und Induktion*), vertieft dargelegt, so das ich hier lediglich einen kurzen Überblick geben werde.

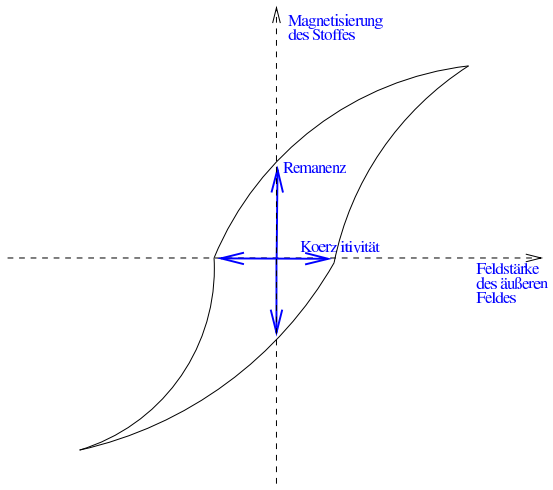
In ferromagnetischen Stoffen, wie Eisen, Nickel oder Cobalt, gibt es mikroskopisch kleine, sogenannte *Weißsche Bezirke* oder *Domänen*. Diese Bezirke sind vollständig magnetisiert, im Normalzustand aber nicht gleich ausgerichtet. Dadurch, das der Magnetismus der Bezirke quasi kreuz und quer im Stoff wirkt, ist der Stoff nach außen an sich nicht magnetisch. Allerdings können die Bezirke gleichgerichtet werden, in dem man sie einem äußeren Magnetfeld aussetzt. Das Magnetfeld richtet alle Bezirke in der selben Richtung aus, nämlich der Flußrichtung des äußeren Magnetfeldes. Da nun alle Weißschen Bezirke in der selben Richtung ausgerichtet sind, wirkt der Elementarmagnetismus des Stoffes und der Körper ist nach außen magnetisch. Im Physikunterricht wurde dies oft demonstriert, in dem ein einfacher Eisennagel mehrfach mit einem Tafelmagneten in der selben Richtung bestrichen wurde. Der Nagel wurde magnetisiert, und wirkte für kurze Zeit selbst wie ein Magnet. Die Weißschen Bezirke ferromagnetischer Stoff bleiben unterschiedlich lange gleichgerichtet, der Körper also unterschiedlich lange magnetisch. Dieses »verbleiben« des äußeren Magnetismuses im Körper bezeichnet man als *Remanenz*[1]. Die *Sättigungsmagnetisierung* gibt an, welche Feldstärke notwendig ist um alle Weißschen Bezirke auszurichten.

Da elektromagnetische Datenträger ihre Daten lange speichern müssen, verwendet man hier spezielle Stoffe mit hoher Remanenz. Dies ist natürlich schön wenn man Daten speichern will, allerdings problematisch, wenn man die Daten sicher löschen möchte. Aufgrund der Remanenz der Datenträger kann man auch nach mehrfachem Überschreiben der Sektoren noch vergangene Ausrichtungen mit teuren Laboreinrichtungen rekonstruieren. Dies ermöglicht es bereits gelöschte und sogar überschriebene Daten wiederherzustellen.

Remanente Weißsche Bezirke können durch ein entgegengesetztes Magnetfeld in die andere Richtung ausgerichtet werden. Die dazu notwendige Stärke des äußeren Magnetfeldes wird als *Koerzitivfeldstärke* bezeichnet. Legt man ein äußeres Magnetfeld an, das sich einmal komplett dreht, werden auch die einzelnen Bezirke mitgedreht. Dies ist quasi so als würde man einen Kompaß nehmen und mit einem Magneten die Nadel nach Norden ausrichten. Dann führt man den Magneten am Kompaß entlang und dreht die Nadel von Nord über Ost und Süd nach Westen und wieder nach Norden. Den Prozess des kompletten Herumdrehens um 360° bezeichnet man als *Hysterese* bzw. *Hysteresekurve*. Diese Kurve ist ebenfalls wieder Material-spezifisch, ermöglicht aber sehr gute Aussagen über das magnetische Verhalten des Stoffes.

[1] von lateinisch: remare, verbleiben

Die nachfolgende Abbildung zeigt eine Freihandskizze einer Hysteresekurve. In ihr sind die Feldstärke des äußeren Magnetfeldes und die Magnetisierung des Stoffes an den Achsen abgetragen. Außerdem wurden die Remanenz und die Koerzitivität eingezeichnet.



Hysteresekurve

Um Daten auf einem elektromagnetischen Datenträger abzulegen, richtet ein Schreib-/Lesekopf die Weißschen Bezirke aus. Ein logisches Bit muß also immer mindestens so groß wie ein Weißscher Bezirk sein. In der Praxis besteht es aber aus mehreren zusammengefassten Bezirken. Die benötigte Feldstärke, die der Schreib-/Lesekopf zum Ausrichten aufwenden muß, entspricht der Koerzitivfeldstärke des Materials. Die Magnetisierung der Weißschen Bezirke des Datenträgers entspricht der Remanenz. Unterschiedlich ausgerichtete Weißsche Bezirke können sich gegenseitig beeinflussen, daher muß mit steigender Kapazität der Datenträger die Remanenz des verwendeten Materials ebenfalls steigen.

Ein interessantes Phänomen magnetischer Stoffe ist die *Bit-Verschiebung* bzw. engl. *Bit Shifting*. Dabei wandern die Weißschen Bezirke, die für die Aufzreichnung verwendet werden, mehrere Mikrometer auf der Platte herum, die Lesespuren verschieben sich also. Technisch kann man diesem Problem einfach begegnen, entweder werden die inneren Spuren auf einem Zylinder weiter, oder man baut eine sogenannte Vorkompensation ein. Dabei verwendet man einen Controller, der den Lesekopf die Verschiebung der Bits ausgleichen lässt. Interessant ist dieses Phänomen für Löschverfahren durch Überschreiben.

Ebenfalls von Bedeutung für das gewählte Thema ist die sogenannte Curie-Temperatur. Auch sie ist eine spezifische Eigenschaft ferromagnetischer Stoffe und bezeichnet die Temperatur, bei der schlagartig alle ferromagnetischen Eigenschaften verschwinden. Setzt man einen Datenträger der Curie-Temperatur aus, verschwindet die Ausrichtung

der Weißschen Bezirke – und damit auch alle Daten. Für die Eisen-Kobalt-Legierung $Fe_{65}Co_{35}$ liegt die Curie-Temperatur bei 920°C und die Sättigungsfeldstärke bei satten 2,45 Tesla. Zum Vergleich: Kernspintomographen arbeiten mit ca. 1 Tesla, das Erdmagnetfeld hat gerade einmal 50 Mikro-Tesla.

Um die Packungsdichte auf Festplatten zu vergrößern, wurden die Codierungsverfahren optimiert. Es existieren inzwischen verschiedene Verfahren, um durch magnetische Flußwechsel logische Bitfolgen zu repräsentieren. Einige Optimierungen vollzogen sich auf physikalisch-elektrotechnischer Ebene, andere in der Codierungstheorie der Nachrichtentechnik. Es existieren beispielsweise verschiedene Verfahren Bitfolgen zu codieren, so das die Anzahl magnetischer Flußwechsel reduziert werden kann. Bekannt und am besten dokumentiert sind die FM- und MFM-Formate – was auch daran liegt das sie für Disketten entwickelt wurden und vergleichsweise alt sind.

Im FM-Format werden Bits in einer sogenannten *Bit-Zelle* abgelegt. Dies ist ein definierter Bereich von Weißschen Bezirken, die insgesamt meist 3 Mikrometer groß sind. Eine solche Bit-Zelle wird halbiert in ein sogenanntes Takt-Bit und ein Daten-Bit. Das Takt-Bit wird immer gesetzt, so das ein Daten-Bit quasi immer zwischen zwei logischen Einsen eingekeilt ist. Ein Daten-Bit repräsentiert dann je nach Ausrichtung eine logische Null oder Eins. Um die Daten auszuwerten, benötigt man lediglich einen Zähler. Jedes ungerade Bit ist ein Takt-Bit und immer gesetzt. Jedes gerade Bit ist ein Daten-Bit und damit entweder eine Null oder eine Eins. Allerdings ist dieses Verfahren recht primitiv und die Datenmenge je Fläche nicht sehr groß. Das MFM-Verfahren verbessert die Datendichte, in dem die Taktung geändert wird. Statt in jeder Bitzelle ein Takt-Bit zu setzen, wird es nur gesetzt wenn das vorige und das folgende Daten-Bit Null gesetzt ist. *Zwei* Daten-Nullen in Folge werden also nur noch durch *ein* Takt-Bit repräsentiert. Damit reduziert sich der Platzbedarf und die Packungsdichte wächst. Das verwendete Codierungsformat ist von Bedeutung, wenn man Daten durch Überschreiben mit statischen Bitmustern löschen will.

Weiterführende Informationen sowohl zu Speicherverfahren als auch zu Codierungstypen finden Sie in (Messmer 1998).

Das Betriebssystem selbst verwendet ein Dateisystem, um Daten abzulegen. Das heißt, die physikalischen und elektrotechnischen Vorgänge der Datenspeicherung bleiben vor dem Betriebssystem und damit auch dem Anwender verborgen. Trotzdem muß das Betriebssystem Daten und *Metadaten* organisieren. Mit Daten bezeichnet man die eigentlichen Nutzdaten, also beispielsweise einen Brief, der als LaTeX-Datei abgespeichert wird. Metadaten bezeichnen »Daten über den Daten« – also Dinge wie Eigentümer der Datei, Zugriffsrechte, letzte Änderung und so weiter. Das Dateisystem organisiert die Ablage der Dateien in den *logischen* Blöcken des Dateisystems und den *physikalischen* Blöcken des Datenträgers. Für gewöhnlich wird beim Erzeugen des Dateisystems eine logische Blockgröße angegeben, meist sind dies 16kB oder mehr. In diese logischen Blöcke werden Dateien geschrieben. Die Datei wird dazu je nach

ihrer Größe in die benötigten Blöcke verteilt. Eine Datei von 100kB Größe wird auf einem Dateisystem mit 16kB Blockgröße auf $100/16 = 6,25$ Blöcke verteilt. Da nur ganze Blöcke angesteuert werden können, wird die Datei in 7 Blöcke geschrieben. Der letzte Block enthält dann nur die restlich verbleibenden 4kB der Datei. In den freibleibenden 12kB des 7. Blockes können noch die Daten einer Datei liegen, die vorher hier abgelegt war. Diese Reste nennt man *File Slacks* bzw. Restblöcke. Auch sie sind von Bedeutung für das sichere Löschen, da sich unter Umständen in diesen Restblöcken« Fragmente vertraulicher Dateien finden können. Im schlimmsten Fall ist ein solcher Block 64kB groß und mit nur einem einzelnen »neuen« Byte belegt, dann lassen sich noch 65535 Byte »alte« Daten rekonstruieren.

Neben der Aufteilung in logische Blöcke sorgt das Dateisystem auch für die Zuordnung der belegten Blöcke zu den Dateien sowie für die Aufteilung der logischen Blöcke auf die physikalischen Blöcke der Datenträger. In den klassischen BSD-Dateisystemen (LFS, FFS) werden Indexknoten (*Index Node*) bzw. Inodes verwendet, um Dateien im Dateisystem zu organisieren. In ihnen finden sich die oben erwähnten Metadaten (Besitzer, Zugriffsrechte, Zugriffszeiten etc.) sowie eine Liste der belegten logischen Dateisystemblöcke. Soll eine Datei gelesen oder geschrieben werden, wird über entsprechende Prozeduren (ffs_read() und ufs_bmap() für das BSD FFS) aus dem Inode die Liste der belegten logischen Blöcke ausgelesen. Die Identifikatoren der logischen Blöcke werden in die Physikalischen übersetzt. Anschließend werden die benötigten Blöcke vom Datenträger angefordert. Soll eine Datei gelöscht werden, Wird lediglich der verwendete Inode und die belegten logischen Blöcke als leer markiert. Die eigentlichen Daten in den Blöcken verbleiben auf dem Datenträger und können beispielsweise mit fsdb(8) rekonstruiert werden. Dazu muss man lediglich von Hand den entsprechenden Inode wieder aufbauen. Sollen Daten unwiderbringlich gelöscht werden, sind spezielle Löschverfahren notwendig, die in den folgenden Kapiteln erläutert werden. Weitere Informationen zu Dateisystem finden Sie im »Daemon-Book« (McKusick u. a. 1999, Kap. 7, »Local Filesystems« und Kap. 8, »Local Filesystores«), das den Aufbau der 4.4BSD-Dateisysteme (LFS, FFS, MFS) sehr detailliert beschreibt.

9.1.1 *Zerstörungsfreie Datenvernichtung*

Um Daten auf magnetischen Datenträgern, wie Festplatten, Disketten oder Bändern, sicher zu löschen, müssen die Blockinhalte der Datei überschrieben werden. Dazu gibt es verschiedene Standards und Methoden, die unterschiedliche Sicherheitsstufen bieten.

Problematisch sind allerdings hierbei grundsätzliche Probleme. Die bereits erwähnten Datei-Reste in Blöcken sowie die Existenz nicht bekannter Kopien der Daten. Das Betriebssystem kann eine Datei unter anderem in den Swap-Speicher ausgelagert

haben, so das sich die Daten auf der Swappartition befinden. Ebenso kann die Datei im temporären Verzeichnis /tmp abgelegt werden. Daher kann man zwar dafür sorgen, das die bekannte Datei sicher gelöscht wird, nicht jedoch alle anderweitig existierenden Kopien der Daten. Außerdem können nur Dateien überschrieben werden, die noch existieren. Bereits unsicher gelöschte Dateien können nicht ohne weiteres überschrieben werden.

Außerdem funktioniert Überschreiben nicht, wenn das Dateisystem die Blockgröße bei jedem Speichervorgang neu zuordnet. Dies tut beispielsweise LFS. Um Daten sicher zu überschreiben, muß der Schreib-Cache der Platte deaktiviert werden. Verwenden Sie dazu unter NetBSD dkctl(8) in der Form `dkctl` *`GERÄT`* `setcache none`. Leider schalten nicht alle IDE-Platten den Schreib-Cache ab, obwohl dies implementiert sein müsste. SCSI-Platten können generell ihren Schreibcache abschalten.

Weiterer Kritikpunkt ist die Abhängigkeit der statischen Bitmuster vom Speicherverfahren der Festplatte. Eine Platte mit FM benötigt andere Bitmuster als MFM. Inzwischen gibt es eine Vielzahl unterschiedlicher Verfahren – auch proprietäre und daher unbekannte. Bisher existiert kein spezifisches Schema der Bitfolgen für die neueren Verfahren wie EEPRML, MTR oder Trellis, da sie kaum oder gar nicht öffentlich dokumentiert sind. Hier hilft nur der Einsatz von Zufallsdaten, die das Signal der Daten verrauschen.

Alles in allem sind Überschreibeverfahren im Dateisystem kritisch zu beurteilen. Sie sind nicht hinreichend wenn der Datenträger ausgesondert wird. Sie sind nur dann nützlich, wenn der Datenträger weiter im Besitz bleibt und möglichst verschlüsselnde Dateisysteme eingesetzte werden.

9.1.1.1 *Einfaches Überschreiben mit Nullen*

Beim einfachen Überschreiben des Datenträgers wird jeder Block mit Nullen überschrieben. Damit schützt man sich zwar vor Programmen, die die Daten wiederherzustellen versuchen, nicht jedoch vor anderen Maßnahmen. Durch das gleichmäßige Bitmuster lassen sich die vorher gespeicherten Datenmuster relativ einfach wieder rekonstruieren. Dies wird zwar schwerer, je höher die Datendichte (und damit Kapazität) der Festplatte ist, ist aber nicht unmöglich.

Diese Methode ist nicht empfehlenswert.

9.1.1.2 *US DoD 5220.22-M (E) und (ECE)*

Ein vom US DoD (United States Department of Defense, US-amerikanisches Verteidigungsministerium) herausgegebener Standard. In 3 Durchläufen werden die Daten erst mit 0xFF, dann 0x00 und abschließend Zufallsdaten überschrieben.

Eine Erweiterung des (E)-Standards ist die (ECE)-Version, mit insgesamt 7 Durchläufen. Zuerst und zuletzt werden die 3 Durchläufe der (E)-Version (0xFF, 0x00, Zufallsdaten) geschrieben. Als 4. Durchlauf werden wiederum Zufallsdaten geschrieben.

Die Zufallsdaten erhöhen die Sicherheit des (ECE)-Verfahrens, schützen jedoch nicht vor aufwendigen Laboranalysen. Außerdem werden beide Standards im (National Security Agency 2001) nicht *im* Dateisystem eingesetzt, sondern direkt auf den Datenträger geschrieben.

Weiterhin wurde befohlen, daß das US-Militär mittels Überschreiben keine Daten löschen darf, die als *Geheim* oder *Streng Geheim* eingestuft wurden. Derartige Daten dürfen nur über Degaußer (siehe 9.2.2.1) vernichtet werden.

9.1.1.3 *Richtlinie zum Geheimschutz von Verschlussachen beim Einsatz von Informationstechnik*

Die Richtlinie zum Geheimschutz von Verschlussachen beim Einsatz von Informationstechnik des Bundesamtes für Sicherheit in der Informationstechnik spezifiziert 7 Durchläufe, je dreimal 0xFF und 0x00 im Wechsel und abschließend ein Durchlauf mit 0xAA.

Aufgrund der statischen Bitmuster nicht empfehlenswert.

9.1.1.4 *Bruce Schneier*

Bruce Schneiers Algorithmus aus (Schneier 1997) überschreibt die Daten erst mit 0xFF, 0x00 und 5-mal mit im streng kryptographischen Sinne zufälligen Daten. Streng zufällige Daten sind jedoch aus signaltheoretischer Sicht nicht zwingend erforderlich, da die zu löschenden Daten »lediglich« durch unregelmäßige Signale verrauscht werden sollen. Hierzu genügen auch pseudo-zufällige Daten wie aus /dev/urandom generiert oder aus Anwendungsdateien wie Videos oder MP3-Dateien.

9.1.1.5 *Peter Gutmann*

Peter Gutmann ist natürlich kein Standard, sondern Autor von (Gutmann 1996). In diesem Artikel analysiert er trefflich die Speichermethoden auf magnetischen Datenträgern. Sein Vorschlag zum sicheren Löschen ist ein 35-faches Überschreiben der Daten. Hierbei werden in den ersten und letzten 4 Läufen Zufallsdaten geschrieben, ansonsten verwendet er spezielle Bitmuster. Diese speziellen Bitmuster sind an die Speicherverfahren der Festplatten (RLL, MFM) angepasst. Ziel ist es hierbei, mithilfe der Bitmuster das rekonstruierbare Signal der Nutzdaten zu verrauschen. Dabei werden bspw. erst die Folge 01010101 und danach das Komplement dazu, 10101010, geschrieben. Damit ist sichergestellt, das jedes »Bit« einmal gedreht wurde.

Die *Idee* des Standards kann als sicher betrachtet werden. Allerdings ist die *Implementierung* durch die Einführung neuer Codierungsstandards veraltet, da nur MFM, (1,7)RLL und (2,7)RLL-kodierte Festplatten betrachtet werden. Anhand stochastischer Berechnungen (siehe (Reinke 2004)) kann man davon ausgehen, das 33 Schreibvorgänge mit zufälligen Daten mit einer Wahrscheinlichkeit von 99% jeden Bit der Festplatte zweimal »umschalten«, die Nutzdaten also sehr sicher von der Platte schrubben. Sieben Durchläufe reichen aus, um jedes Bit der Platte einmal umzuorientieren. Abbildung 9.2 zeigt ein Skript, das eine Festplatte 33-mal mit Zufallsdaten überschreibt. Dies kann mit den unten beschriebenen Programmen wipe(1) und dd(1) oder neb-wip(1) problemlos realisiert werden.

9.2 PROGRAMME

Im folgenden werde ich einige Programme vorstellen, die Daten durch mehrfaches Überschreiben löschen. Diese Verfahrensweise muß aber kritisch betrachtet werden, da es einige Probleme zu beachten gilt. Überschreibeverfahren funktionieren nicht, wenn das zugrundeliegende Dateisystem die Blockgröße dynamisch alloziert. Es ist schlicht nicht möglich beim Überschreiben genau die Blöcke zu treffen in denen die Nutzdaten liegen.

Empfehlenswert sind die angegebenen Löschprogramme im Dateisystem nur, wenn der Datenträger weiterhin unter physikalischer Kontrolle bleibt. Soll er ausgesondert werden, sind zerstörende Maßnahmen sinnvoller oder zumindest wie in Kapitel 9.2.0.4 beschrieben der Einsatz von dd(1), um direkt auf die Platte zu schreiben.

9.2.0.1 *rm(1)*

NetBSDs rm(1) kann mit der Option -P zu löschende Dateien mit 0xFF, 0x00 und Zufallsdaten überschreiben. Damit können Daten gelöscht werden, die vor einfacher Wiederherstellung durch Software geschützt werden sollen. Der Standard schützt nicht vor Laboranalysen. Sicheres Löschen von Dateien im Dateisystem ist in NetBSD mit Bordmitteln nicht zu bewerkstelligen. Lediglich komplette Datenträger oder zumindest Partition lassen sich mit Zufallsdaten schrubben.

9.2.0.2 *destroy(1)*

In Pkgsrc findet sich das Programm destroy(1) von Shane Kinney. Destroy überschreibt Dateien je Durchlauf einmal mit Null und einmal mit Zufallsdaten. Es können 1 bis 20 Durchläufe durchgeführt werden. Dieses Verfahren kann bei hinreichend vielen Durchläufen (mindestens 7) als sicher angesehen werden.

9.2.0.3 *wipe(1)*

Wipe von Tom Vier ist als pkgsrc/sysutils/wipe installierbar. Es löscht komplette Verzeichnisbäume und ist wesentlich flexibler konfigurierbar als Destroy. Wichtige Optionen sind:

- -I unterbindet Bestätigungsanfrage an den Benutzer
- -r löscht rekursiv Verzeichnisse
- -s schaltet Fortschrittsanzeige aus
- -v zeigt Fortschrittsanzeige für jede Datei an
- -d löscht Dateien nach dem Überschreiben
- -D behält Dateien nach dem Überschreiben
- -n löscht spezielle Dateien (Fifos, Links, etc.)
- -N überspringt spezielle Dateien (Fifos, Links, etc.)
- -k oder -K sperrt Dateien während des Überschreibens bzw. sperrt sie nicht

- -o oder -O schreibt die Bitmuster and STDOUT oder in eine Datei
- -p x wiederholt alle Läufe x-mal
- -x x führt x Durchläufe mit Zufallsdaten durch, gilt pro Lauf, der mit -p festegelegt wurde, -p4 -x2 führt $4 * 2 = 8$ Zufallsläufe durch.
- -b überschreibt einmal mit angegebenem Byte
- -l x Sicherheitsstufe
  - 0 Der Pseudozufallsgenerator wird nur einmal initialisiert
  - 1 Der Pseudozufallsgenerator wird für jede Datei initialisiert
  - 2 Der Pseudozufallsgenerator wird für jeden Durchlauf initialisiert
- -a oder -A überschreibt Daten bis der Plattenplatz verbraucht ist. bzw tut dies nicht
- -z oder -Z überschreibt Dateien einmal mit Nullen oder deaktiviert diesen Modus
- -t oder -T aktiviert oder deaktiviert statische Durchläufe
- -B und -S setzt sie Anzahl und Größe der Sektoren eines Block-Gerätes
- -C maximale Dateipuffergröße

Standardmäßig läuft wipe mit den Optionen `-ZdntVAk0 -S512 -C4096 -l1 -x8 -p1`, d.h. keine Nullen, löscht Dateien und spezielle Dateien, mit statischen Durchläufen, Fortschrittsanzeige, sperrt dabei die Dateien, schreibt in eine Datei mit Blockgröße von 512, Dateipuffergröße von 4096, der Pseudozufallsgenerator wird für jede Datei initialisiert, es wird achtmal mit Zufallsdaten überschrieben und das ganze Muster einmal durchlaufen.

Wipe kann auch als Quelle für andere Prozesse verwendet werden, so kann man mit `wipe -T -x1 -o 20971520 | dd of=cont2 bs=20971520` 20971520 Bytes, also 20MB, Pseudozufallsdaten generieren und mit dd(1) in eine Datei schreiben lassen. Mit Wipe als Quelle dauert der Prozess knapp 1,5 Sekunden, mit /dev/urandom 26 Sekunden.

9.2.0.4 *dd(1)*

Wie in Kapitel 9.1.1 beschrieben, haben Überschreibe-Verfahren einige grundsätzliche Probleme. Sie funktionieren nicht auf Dateisystemen mit dynamischer Größenanpassung, können nicht-bekannte Kopien nicht überschreiben und erwischen keine Datei-Reste in Blöcken.

Möchte man sicherstellen das alle Daten auf einer Festplatte – oder zumindest einer Partition – überschrieben werden, muß man zu des Administrators Superwaffe – dd(1) – greifen. Dd kann man verwenden, um direkt auf Raw-Geräte zu schreiben, also unterhalb des Dateisystems. Ein simples `dd if=/dev/wd0h of=/tmp/wd0h.dd` schreibt den *gesamten* Inhalt der Partition wdoh in eine Datei. Umgekehrt kann man mit `dd if=/tmp/container of=/dev/wd0h` eine Datei auf die Partiton schreiben lassen. Mithilfe der beiden Geräte `/dev/zero` und `/dev/urandom` kann man eine unendliche Folge von Nullen bzw. Pseudozufallsdaten erzeugen lassen und auf eine Partition oder Festplatte schreiben lassen. Mit den Befehlen `dd if=/dev/zero of=/dev/wd0h ; dd if=/dev/urandom of=/dev/wd0h` wird die Partition wdoh zuerst mit Nullen und dann mit Zufallsdaten überschrieben. Dabei wird *jeder* Block der Partition, auf den die Festplatte Zugriff hat, überschrieben. Ein derartiges Verfahren ist von der Technik her das sicherste, da die Daten direkt über das Raw-Gerät überschrieben werden. Es müssen lediglich hinreichend viele Durchläufe benutzt werden, wie beispielsweise in Abbildung 9.1 gezeigt wird. Hierbei wird eine Schleife eingesetzt, die 5-mal durchlaufen wird. In der Schleife wird die gesamte zweite IDE-Festplatte (*wd1* ist die zweite Platte, die Partition *d* die gesamte Platte), dreimal mittels dd(1) beschrieben. Zuerst mit Nullen aus /dev/zero, dann mit Pseudozufallsdaten aus /dev/urandom und abschließend mit wipe(1) als Pseudozufallsgenerator. Die Option bs=2m setzt die Blockgröße auf 2 Megabyte. Dies erhöht den Schreibdurchsatz auf die Festplatte erheblich, als Wert sollte hier immer die Größe des Platten-Caches verwendet werden. Die Option progress gibt alle 50 Blöcke (also 50 * 2m = 100MB) einen kleinen Punkt als Fortschrittsanzeige aus. In der letzten Zeile wird das Programm wipe(1) als Zufallsgenerator verwendet, da es wesentlich schneller ist als /dev/urandom. Die Option `-o 2097152` setzt zwar die Größe der zu erzeugenden Zufallsdaten auf 2097152 Byte, also genau 2MB, aber die Option `-a` überlädt `-o` und generiert solange Daten bis das Gerät voll ist und dd(1) abbricht. Die meisten Festplatten lassen sich schneller schrubben wenn beide Programme mit einer Blockgröße von 2MB arbeiten.

Abbildung 9.2 zeigt ein einfaches Shellskript, das insgesamt 33-mal eine Festplatte mit Zufallsdaten überschreibt. Dazu wird wipe(1) als Quelle für Zufallsdaten verwendet. Mit der Option `-o 2097152` wird wipe(1) angewiesen 2MB große Blöcke zu schreiben. Dies erhöht den Durchsatz dramatisch gegenüber anderen Einstellungen. Auf meinem »IBM Thinkpad X24«-Laptop steigt der Schreibdurchsatz von 6MB auf 10MB, wenn die Blockgröße auf 2MB gesetzt wird. Der einzustellende Wert hängt jedoch von der verwendeten Festplatte ab, so das man ein paar Blockgrößen ausprobieren sollte. Das

```
# for i in ‘seq 1 5‘
> do
> dd if=/dev/zero of=/dev/wd1d bs=2m progress=50
> dd if=/dev/urandom of=/dev/wd1d bs=2m progress=50
> wipe -T -x 1 -o 2097152 -a | dd of=/dev/wd1d bs=2m
> done
```

Abbildung 9.1.: Eine Partition mit dd(1) überschreiben

Programm dd(1) gibt beim Beenden immer die durchschnittlich geschriebenen Bytes je Sekunde aus, so das sich etwas experimentieren lohnt. Außerdem wird im Skript der Schreibcache jeder Platte abgeschaltet und zusätzlich nach jedem Durchlauf auf die Platte geschriebenen. Die Maßnahme ist notwendig, da nicht alle IDE-Platten den Schreibcache wirklich abschalten.

```
#!/bin/sh

if [ ! $DURCHLAEUFE ]; then
DURCHLAEUFE=33
fi

for i in $@
do
if [ ! -b /dev/$i ]; then
printf "\nDas Geraet /dev/$i existiert nicht!\n\n"
else
/sbin/dkctl $i setcache r

for j in ‘seq 1 $DURCHLAEUFE‘
do
printf "\n$j. Durchlauf auf Platte /dev/$i"

/usr/pkg/bin/wipe -T -x1 -o 2097152 -a | \
/bin/dd of=/dev/$i bs=2m

/sbin/dkctl $i synccache force
done
fi
done
```

Abbildung 9.2.: Eine Partition 33-mal mit wipe(1) und dd(1) überschreiben

9.2.0.5 *neb-wipe(1)*

NetBSD-Wipe ist über Pkgsrc-WIP erhältlich. Es schreibt direkt auf ein Raw-Gerät. Dazu kann es Gutmann-Bitmuster verwenden oder Zufallsdaten erzeugen. Es können beliebig viele Durchläufe geschrieben werden. Der verwendete Pseudozufallszahlengenerator ist wesentlich schneller als /dev/urandom.

9.2.0.6 *buffer(1)*

Buffer ist ein Zwischenpuffer, der für Magnetbänder entworfen wurde. Der Eingabefluß wird geglättet und zwischengespeichert, so daß das Bandlaufwerk nicht ständig vor- und zurückspulen muß, falls gelesene Daten ausbleiben.

Anstatt dd(1) sollte man buffer(1) verwenden um direkt auf Bänder zu schreiben, insbesondere wenn erst Zufallsdaten generiert werden müssen. Bei schnellen Bändern empfiehlt es sich die Zufallsdaten erst in eine Datei zu schreiben und diese dann auf das Band zu bringen.

Mit `wipe -T -x1 -o 20971520 -a | dd of=wipedatei bs=1m count=2048 ; buffer -i wipedatei -o /dev/nrst0` wird zuerst eine 2GB große Datei mit Zufallsdaten generiert und anschließend mit buffer auf das erste Bandlaufwerk geschrieben.

9.2.1 *Die NetBSD/Schrubber Live-CD*

Um eine Festplatte komplett zu putzen, benötigt man ein System mit bestimmten Werkzeugen. Zu diesem Zwecke habe ich eine NetBSD/LiveCD erstellt, die auf einem minimalistischen NetBSD 3.1 basiert und alle notwendigen Werkzeuge mitbringt.

Mit der CD kann man einen beliebigen PC booten und zum Platten schrubben einsetzen. Einzige Voraussetzung ist ein bootfähiges CDROM-Laufwerk und unterstützte IDE- bzw. SCSI-Controller, was aber meist der Fall ist. Das System ist sehr minimalistisch, es gibt keine Manpages, keine einkompilierten Netzwerkkarten und sonstige Hardware, die nicht benötigt wird. Im Kernel sind alle verfügbaren IDE- und SCSI-Controller aktiviert, so das nahezu jedes Massenspeichergerät angesprochen werden kann.

Auf der CD finden sich neben der Allzweckwaffe dd(1) auch die oben vorgestellten Programm Wipe und Neb-Wipe, sowie das Shell-Skript `wipe-33.sh`. Es verwendet Wipe um 33-mal alle übergebenen Partitionen mit Zufallsdaten zu überschreiben. Die Anzahl der Durchläufe kann mit der Umgebungsvariable `DURCHLAEUFE` festgelegt

werden, die zu überschreibenden Partitionen werden als Argument übergeben. Das Skript prüft alle Geräte auf Existenz und gibt ggf. eine Warnmeldung aus. Falls das Gerät existiert, wird der Schreibcache mit dkctl(8) deaktiviert und eine Schleife mit der entsprechenden Anzahl der Durchläufe gestartet. Nach jedem Durchlauf wird mit dkctl(8) der Schreibcache zum synchronisieren gezwungen. Dies ist leider notwendig, da einige IDE-Platten den Schreibcache nicht abschalten können. Mit diesem Behelf wird zumindest am Ende eines Durchlaufs dafür gesorgt, das der Cache auf die Platte geschrieben wird. Das Skript gibt einige Statusmeldungen nach jedem Durchlauf aus, um den Fortschritt anzuzeigen. Mit `export DURCHLAEUFE=7 && ./wipe-33.sh wd0d sd0d` wird die erste IDE- und die erste SCSI-Platte komplett in 7 Durchläufen geschrubbt.

Die CD lässt sich über das Passwort-lose Root-Konto benutzen. Es werden 8 Terminals initialisiert, die über STRG+ALT+F[1-8] erreicht werden können. Die Terminals F4-F8 sind mit 50 statt 25 Textzeilen initialisiert, was allerdings auf einigen sehr alten Grafikkarten (z.B. ATI) zu Problemen führt. Das erste Terminal (STRG+ALT+F1), auf dem die Bootmeldungen erscheinen, ist die Konsole. Dort schlagen auch die Kernel-Meldungen an den Syslogd auf. Von daher ist es ratsam, eine anderes Terminal zum Arbeiten zu verwenden.

9.2.2 *Physikalische Datenvernichtung*

Ein sehr sicherer Weg der Datenvernichtung ist die physikalische Vernichtung des Datenträgers. Wenn das Medium oder Laufwerk nicht mehr funktioniert, kann es auch nicht mehr ausgelesen werden. Für verschiedene Laufwerks- bzw. Datenträgertypen gibt es unterschiedliche Arten und Wege Daten zu vernichten.

Eine Warnung vorneweg: ich beschreibe den Einsatz von Feuer, Strom und mechanischer Gewalt, um Datenträger zu vernichten. Wenn Sie von diesen Dingen keine Ahnung haben, lassen Sie die Finger davon! Wenn Sie Datenträger mechanisch zerstören wollen, achten Sie auf Ihren Schutz! Tragen Sie zumindest ordentliche Arbeitshandschuhe und eine Schutzbrille! Ein umherfliegender Splitter einer Festplatten-Scheibe kann problemlos einen Augapfel zerstören.

9.2.2.1 *Magnetische Datenträger*

Wie in Kapitel 9.2.2.1 beschrieben, speichern magnetische Datenträger ihre Daten in dem sie kleine magnetische Bereiche magnetisieren. Verliert der Datenträger nun seinen Magnetismus, sind schlagartig alle Daten unwiderruflich vernichtet.

Für jeden Magneten existiert als spezifische Eigenschaft die Curie-Temperatur. Bei

der Curie-Temperatur verliert jeder Magnet schlagartig alle magnetischen Eigenschaften. Die Daten sind also definitiv und unwiderbringlich vernichtet. Einziger Nachteil ist die Höhe der Curie-Temperatur. Bei handelsüblichen Festplatten liegt sie bei ca. 1100 Kelvin bzw. 800C, ist also mit dem heimischen Backofen schwer zu erreichen. Hat man jedoch direkten Zugang zu einem Hochofen, Walzwerk oder Müllverbrennungsanlagen, kann man diesen Weg der Datenvernichtung wählen.

Eine zweite Möglichkeit der physikalischen Datenvernichtung ist der sogenannte Degaußer. In ihm wird ein starkes Magnetfeld mit einer Flußdichte von mindestens 1 Tesla aufgebaut. Zum Vergleich: das Erdmagnetfeld hat ca. 50T also 0,000.050 Tesla, der stärkste in Europa verfügbare Kernspintomograph des Leibnitz-Institutes Magdeburg gigantische 7 Tesla. Dieses starke Magnetfeld wird um den Datenträger herum aufgebaut und durchläuft die Hystereseschleife (rotiert also mindestens einmal rundherum) und richtet somit *alle* Weißschen Bezirke neu aus. Damit ist jeglich Form von Daten in den Magnet-Partikeln vernichtet. Es gibt auch keine Daten in irgendwelchen Nebenspuren oder Remanenz bereits gelöschter Dateien. Dieses System ist definitiv sicher, zerstört aber die Festplatte. Denn das starke Magnetfeld vernichtet auch die Steuerelektronik und Servospuren der Festplatten.

Problematisch ist bei beiden Verfahren, das jeder Datensektor des Mediums erreicht werden muß. Das Magnetfeld muss jede Datenscheibe komplett durchfluten bzw. die Curie-Temperatur mehrere Minuten an jedem Punkt jeder Datenplatte anliegen. Da die Fesplattengehäuse schirmen, öffnen Sie die Gehäuse und bearbeiten Sie nur die Datenplatten im Degaußer bzw. im Hochofen.

Einen derartigen Degaußer (Bild 9.3) kann man beispielsweise von der Firma Ibas mieten. Selbstbaulösungen mit etwas gewickeltem Kupferdraht sind zwar möglich, der Konsum aller MacGyver-Folgen qualifiziert aber noch nicht zur Elektro-Fachkraft. Lassen Sie also unbedingt die Finger von derartigen Experimenten.



Abbildung 9.3.: Degaußer 2 von Ibas (Quelle: www.ibas.de)

Nachteil beider Verfahren ist die Zerstörung der Datenträger und ihre nicht-triviale Beschaffung. Zumindest ein Degaußer läßt sich jedoch gegen Entgelt mieten. Vorteil ist die hohe Sicherheit der Verfahren. Außerdem funktionieren sie auch mit defekten Platten, beispielsweise nach einem Head-Crash oder mit defekter Elektronik.

9.2.2.2 *Optische Datenträger*

Optische Datenträger speichern ihre Daten in einer Trägerschicht, die von einem Laser abgetastet wird. Wird diese Trägerschicht physikalisch vernichtet, sind auch die Daten weg. Die verbreitetsten optischen Datenträger, CDs und DVDs, bestehen prinzipiell aus drei Schichten: der unteren, durchsichtigen aber stabilen, Schutzschicht, der mittleren Trägerschicht und der oberen Deckschicht.

Eine CD lässt sich am einfachsten vernichten, wenn man die obere Deckschicht, also das was normalerweise bedruckt ist ober beschrieben werden kann, zerstört. Die darunterliegende Trägerschicht oxydiert in der Regel sofort bzw. wird durch die mechanische Einwirkung ebenfalls vernichtet. Die Trägerschicht sieht aus wie hauchdünne Alu-Folie und zerfällt sehr schnell in sehr kleine Teile. Diese verteilen sich gerne in der Gegend und bevorzugt in Teppiche. Dort kann man ihnen nur sehr schwer mit dem Staubsauger zu Leibe rücken. Zerstören Sie also eine CD in einer Umgebung in der dies nicht stört oder benutzen Sie einen großen Müllbeutel oder eine Abfalltonne als Hülle. Tragen Sie dabei Arbeitshandschuhe und schützen Sie ihre Augen! Sie können den CDs oder DVDs mit einem stabilen Schraubenzieher, robusten Messer oder auch einem Akkuschrauber bzw. einer Bohrmaschine mit Metallbohrer zu Leibe rücken. Entfernen Sie die Trägerschicht gründlich und zerbrechen Sie die CD danach.

In einem Handwerksbetrieb haben wir ein einfaches System entwickelt um massenhaft CDs zu zerstören. Wir haben eine simple Spindel aus einem Holzbrett und einer 15mm-Gewindestange gebaut. Auf die Spindel wurden die CDs aufgestapelt und von oben mit einer großen Unterlegscheibe und einer Flügelmutter befestigt. Die gesamte Spindel wurde unter einer Standbohrmaschine sicher eingespannt und mehrfach mit einem 12mm Metallbohrer durchbohrt.

Weitere Möglichkeiten sind Häcksler bzw. Aktenvernichter, die auch CDs untertützen bzw. speziell für CDs gedacht sind. Normales Zerbrechen der CD in größere Teile reicht nicht aus, um Spezialfirmen davon abzuhalten Teile der CD wieder auszulesen.

Physikalische Vernichtung lässt sich auch durch Ätzen der CD erreichen. Hierzu muss die CD in eine Säure oder ähnliches getaucht werden. Der Umgang mit Säuren und dergleichen ist aber nicht trivial und daher hier nicht weiter von Bedeutung.

Ebenfalls nicht wirksam ist ein Zerkratzen der Unterseite der CD, da derartige Kratzer mit Füllmasse korrigiert werden können. Es muß die Trägerschicht des Mediums vernichtet werden. Dies lässt sich am besten über die Oberseite erreichen.

Wirksam, aber nicht empfehlenswert, ist die Mikrowelle. Durch die Mikrowellenstrahlung wird innerhalb von Millisekunden die Trägerschicht effizient und nichtrekonstruierbar vernichtet. Dabei entstehen aber giftige und umweltschädliche Dämpfe sowie eine erhöhte Brandgefahr.

9.2.2.3 *Magneto-Optische Datenträger*

MO-Medien sind aufgrund der verwendete Speicherverfahren und technischen Umsetzungen vergleichsweise robust. Magneto-Optische Datenträger *schreiben* Daten magnetisch, *lesen* diese dann aber wieder optisch über Laser aus. Soll ein Datum auf dem Datenträger geschrieben werden, wird die Oberfläche von Laser über die Curie-Temperatur erhitzt. Damit verschwindet die Magnetisierung der Oberfläche und nach Herunterkühlen unter die Curie-Temperatur wird mit dem Magnetkopf das Datum geschrieben. Weitere Informationen zu MO-Laufwerken find sich in (Messmer 1998, Kapitel 31.8.4, *Magnetooptische Laufwerke*).

Da das Erhitzen der magnetischen Schicht über die Curie-Temperatur heraus jeglichen Magnetismus entfernt, reicht ein einmaliges Überschreiben der *gesamten* Platte aus.

9.2.2.4 *Flash-EEPROM*

Flash-EEPROMs werden in USB-Sticks oder Compact-Flash-Karten verwendet. Sie verwenden keinen Magnetismus um Daten abzulegen, sondern sogenannte »Floating Gates« der Feldeffekt-Transistoren (FETs). Vereinfacht gesprochen ist das Floating Gate ein Leiter zwischen zwei Punkten (Source und Drain, Quelle und Abfluß genannt). Durch Anlegen einer Spannung kann der Leiter blockiert werden, eine gleichhohe, aber negative, Spannung öffnet den Leiter wieder. Je nachdem ob der Leiter offen oder blockiert ist, wird eine logische 1 oder 0 repräsentiert. Ein FET ist also im Prinzip nichts anderes als ein sehr kleiner Schalter.

Daten können erst überschrieben werden, nachdem die alten Daten gelöscht wurden. Die Löschoperation kann dabei nur bei bestimmten EEPROMs auf Byte-Ebene erfolgen, meist geschieht dies Sektor-weise, woher auch der Name *Flash* rührt. Da es bei den Feldeffekt-Transistoren nicht möglich ist vorige Belegungen zu rekonstruieren, genügt ein einfaches Überschreiben der Daten mit einem beliebigen Bit-Muster. Praktisch lässt sich dies am einfachsten mit einem `dd if=/dev/zero of=/dev/sd0d` realisieren.

Beachten Sie aber, das ein Flash-Speicher systembedingt nur eine begrenzte Anzahl von Lösch-Operationen durchführen kann. Dies rührt daher, das bei jedem durchtunneln der Oxid-Schicht des Floating-Gates, diese Schicht beschädigt wird. Zur Zeit geben die Hersteller Zyklen von 100.000 bis 1.000.000 Löschvporgängen an.

9.3 VORBEUGENDE MASSNAHMEN

Die einfachste Maßnahme um die Datenvernichtung zu vereinfachen ist die Datensparsamkeit. Daten die nie gespeichert wurden, müssen auch nie gelöscht werden. Da nicht alle sensiblen Daten nicht gespeichert werden können, ist der Einsatz verschlüsselnder Dateisysteme geboten. Damit wird der direkte Zugriff auf die Daten verhindert. NetBSD bietet mit CGD und CFS zwei Varianten um Dateisysteme zu verschlüsseln, mit CGD können gesamte Partitionen – auch /tmp und die Swap-Partition – verschlüsselt werden. Die Artikel (Schumacher 2004) und (Schumacher 2006c) beschreiben die Funktionsweise verschlüsselnder Dateisystem sowie den Einsatz von CGD und CFS unter NetBSD.

Auch wenn Daten verschlüsselt wurden, müssen sie sicher gelöscht werden. Dies gilt insbesondere für die Passwort-Hashes oder Schlüssel der Dateisysteme. Außerdem kann eine Festplatte oder ein Magnetband 30 Jahre halten – und niemand kann garantieren, das die eingesetzten kryptographischen Verfahren dann noch sicher sind. Eine adäquate Vorgehensweise ist hier das mehrfache Überschreiben der Festplatte mit Zufallsdaten.

Wenn Sie große Datenmengen verwalten müssen, organisieren Sie die Datenstruktur entsprechend der Schutzstufen. Legen Sie eindeutige Kriterien zur Bestimmung der Schutzstufe fest und teilen Sie beispielsweise 5 Schutzstufen (0 – nicht schutzwürdig, 4 – Streng Geheim) ein. Legen Sie anschließend fest, wie die Schutzstufen zu löschen sind (0 – einmliges Überschreiben mit /dev/null; 4 – Sicheres Löschen der Datenträger durch mehrfaches Überschreiben und anschließende physikalische Vernichtung der Datenträger durch Degaußer). Setzen Sie von vornherein ein Verschlüsselungsgebot für alle Daten ab Stufe 2 durch. Verschlüsseln Sie ebenfalls alle Sicherungskopien, Arbeitskopien oder sonstigen Kopien der Daten. Verbieten Sie in Ihrer Sicherheitsrichtlinie, die geschützten Daten anderweitig abzulegen als erlaubt. Beachten Sie ebenfalls Medienbrüche (Ausdrucke). Tragen Sie Sorge, das auch Ausdrucke von geschützten Dateien geschützt und entsprechend vernichtet werden. Ebenfalls hilfreich ist es, die Schutzstufe im Dateinamen zu hinterlegen, so das eine Datei bspw. `Personalakte-Müller.SS4.txt` heißt. Ebenfalls hilfreich ist insbesondere in großen Organisationen eine Markierung der Datenträger, die ihre Schutzstufe wiedergibt. Eine rote »S4« mit Folienstift auf die Festplatte gemalt oder auf einem hitzebeständigen, resistenten Etikett dürfte ausreichen.

Wenn Sie einen externen Dienstleister mit der Vernichtung Ihrer Datenträger betrauen ist es empfehlenswert, die Datenträger vorher durch mehrfaches Überschreiben zu löschen. Es sind auch schon in Sicherheitsunternehmen Datenträger gestohlen worden.

9.4 Datensicherung und Datenvernichtung

Auch sensible Daten müssen gesichert werden. Dazu bedarf es unbedingt des Einsatzes von Verschlüsselung. Verschlüsseln Sie ausnahmslos *jede* Kopie der sensiblen Daten – auch jene die Sie in Ihrer geschützten Umgebung aufbewahren.

Richten Sie gegebenenfalls verschiedene Sicherungskreise ein. Dies ist notwendig wenn Sie Daten haben, deren Lagerfrist kürzer ist als Ihre gängige Archivierungsdauer. Der Gesetzgeber schreibt beispielsweise Aufbewahrungsfristen von maximal 2 Jahren für Einträge in Personalakten vor. Wenn Sie eine Personalakte mit Eintrag 5 Jahre archivieren, kann der betreffende Mitarbeiter Sie verklagen und wird problemlos gewinnen. Sorgen Sie daher dafür das Daten mit Verfallsfrist entsprechend gesondert gesichert und aufbewahrt werden. Überwachen Sie die Verfallsdaten im Zuge Ihrer regelmäßigen Inventur der Sicherungsmedien! Markieren Sie sensible Medien oder Medien mit Verfallsdatum eindeutig. Dies kann mit bunten Etiketten, einem einfachen Folienschreiber oder über Softwarelabel (bspw. dump -L 'SS4 2009-01-31') geschehen.

Mein Handbuch zur Datensicherung (Schumacher 2006a) widmet sich diesem Thema ausführlich und beschreibt wie verschiedene Programme (Bacual oder Amanda) konfiguriert werden müssen, um verschiedene Sicherungskreise einzurichten.

9.5 Fazit

Sicheres Datenlöschen ist mit etwas Aufwand möglich. Selbst nichtzerstörende Verfahren wie mehrfaches Überschreiben bietet Schutz gegen übliche Laboranalysen. Das bedeutet, das mindestens 7-mal, besser noch 33-mal, Zufallsdaten auf das Raw-Gerät geschrieben werden müssen.

Sind die Daten so wertvoll das sie ausführliche und teure Laboranalysen rechtfertigen, hilft nur die physikalische Vernichtung mittels angemietetem Degaußer oder Curie-Temperatur, falls man darauf Zugriff hat. Man kann die Datenvernichtung auch an externe Dienstleister abtreten, beachten Sie aber, das es dann zu unüberwachbaren Sicherheitsproblemen kommen kann – überschreiben Sie die Datenträger also sicherheitshalber.

Eine Richtlinie zum sicheren Löschen ist in größeren Organisationen zwingend erforderlich. Bewerten Sie anfallende Daten bezüglich ihrer Schutzstufe und legen Sie passende Löschverfahren fest. Markieren Sie sensible Datenträger und berücksichtigen Sie sensible Daten und ihre Verfallsdaten in der Sicherungsstrategie.

10

SCHADSOFTWARE BEKÄMPFEN

Viren sind das Tschernobyl der Datenverarbeitung.

(Wau Holland)

10.1 DATEISYSTEMINTEGRITÄT MIT MTREE PRÜFEN

Mtree wurde als Dateisystemintegritätsprüfer entwickelt, um zu prüfen ob Installationen erfolgreich abgeschlossen wurden. Dazu wird eine einfache Datenbank, in der Dateiname, Dateistatus und Prüfsumme über der Datei abgelegt werden, erzeugt.

Sofern man diese initiale Datenbank entsprechend schützt (z. B. auf einem schreibgeschützten Medium oder verschlüsselt), kann man diese Datenbank mit dem aktuellen System Datenbank vergleichen. Weicht die Prüfsumme bzw. der Status einer Datei ab, wurde diese offensichtlich manipuliert.

Ziel des Tests ist es, insbesondere Rootkits zu enttarnen, diese Kits werden in der Regel nach einem erfolgreichen Einbruch installiert und dienen dazu die Anwesenheit eines Einbrechers zu verschleiern. Meist werden hierzu manipulierte Binaries wichtiger Systemprogramme installiert, um z. B. zu verhindern daß `ps -ax` die laufenden Prozeße des Einbrechers anzeigt. Nachteilig ist hierbei allerdings das `mtree` erst reaktiv funktioniert, d. h. es gelingt dem Einbrecher das Rootkit zu installieren und er wird erst beim nächsten Durchlauf von `mtree` enttarnt.

10.1.1 *Konfiguration und Einsatz*

Mtree gehört zum Basissystem und muss nicht extra installiert werden. Es wird bei Systemaktualisierungen verwendet, um die Korrektheit der eingespielten Dateien zu verifizieren.

Mit der Option -c erstellt mtree ausgehend vom aktuellen Verzeichnis eine Liste mit Dateieigenschaften wie in Listing 10.1 gezeigt.

[!h]

```
$ mtree -c
#          user: stefan
#       machine: balmung.net-tex.de
#          tree: /etc/X11/rstart/contexts
#          date: Tue Apr 25 22:24:08 2006

# .
/set type=file uid=0 gid=0 mode=0444 nlink=1 flags=none
.               type=dir mode=0755 nlink=2 time=1140283765.495867000
    @List       size=1306 time=1135575735.0
    default     size=1286 time=1135575735.0
    x           type=link mode=0755 time=1135576379.0 link=x11r6
    x11         type=link mode=0755 time=1135576379.0 link=x11r6
    x11r6       size=1525 time=1135575735.0
stefan@balmung {13}
```

Listing 10.1: von mtree erzeugte Eigenschaftenliste

Welche Attribute mtree(8) anzeigt, können Sie mit Schlüsselwörtern bestimmen. Die wichtigsten sind:

IGNORE N Pfade unterhalb von N sollen ignoriert werden

GID, GNAME, UID, UNAME gibt die Gruppen/User ID bzw. Name wieder

MD5, RMD160, SHA1

MODE oktaler Dateimodus

SIZE Größe in Byte

TIME letzter Modifizierungszeitpunkt

-L folge allen Links

-P PFAD PFAD als Startpfad setzen (auch mehrfach möglich)

-X DATEI Pfad- oder Dateinamen aus DATEI sollen ignoriert werden

Um die Datenbank effektiv einsetzen zu können, muss sie vor einem möglichen Einbrecher geschützt werden, die geschieht am einfachsten mit einer kryptographischen Signatur (PGP oder OpenSSL) oder in dem sie verschlüsselt abgelegt wird. Ebenfalls

[!h]

```
$ mtree -L -c -K sha1,rmd160,gname,uname,mode -p /bin -p /sbin -X /etc/mtree.excl
#          user: stefan
#       machine: balmung.net-tex.de
#          tree: /
#          date: Tue Apr 25 22:29:08 2006

# .
/set type=file uname=root gname=wheel mode=0755 nlink=1 flags=none
.               type=dir nlink=22 time=1143916415.520797000
    .cshrc      mode=0644 size=940 time=1062527967.0 \
                rmd160=7559c5dc7215d8534c68a5e13bb8b6ffcab41d23 \
                sha1=fe2d675b3a4bc6febedde9f75ea8c7a9eac0c5e9
    .profile    mode=0644 size=519 time=1109063537.0 \
                rmd160=189f64161787c5c055bee918d9e8e7ab66c23742 \
                sha1=5a20146835d958b0cd702c401d28a3eb63a634e6
    boot        mode=0444 size=53820 time=1139597480.920000000 \
                rmd160=067a170f143c0b8e842dd3e0280b8af33bbd22dd \
                sha1=4f5af1b54b40c04ac9f277361eff7cd338d5cb86
    netbsd      size=8898275 time=1139697721.90897000 \
                rmd160=64cb2c68bce32d82e9ceca7b07e89a45f820f0ee \
                sha1=29847163d46c719ad6988fcd2f8b648b9d76e712
```

Listing 10.2: von mtree erzeugte Eigenschaftenliste (2)

möglich ist es bei einem entfernten Rechner Prüfsummen (sha1, rmd160, md5) über die Datenbank zu erstellen und auszudrucken oder aufzuschreiben.

Ist die Integrität der originalen Datenbank sichergestellt, kann man nun das laufende System gegen diese Datenbank abgleichen lassen. Mtree macht dies automatisch mit der Option `-f`, siehe Listing 10.3. Hier wurden auch Dateien gefunden die verändert, gelöscht oder neu erstellt wurden.

Sie können ihren Rechner mit mtree und cron nächtlich Prüfen lassen. NetBSD verfügt mit daily.conf(5) und security.conf(5) über Mechanismen, die per cron täglich bestimmte Verwaltungsmaßnahmen und Sicherheitsstest durchführt. Dabei gibt es in /etc/defaults/daily.conf den Schalter `run_security=YES`, der security.conf ausführt. /etc/defaults/security.conf wiederum startet mittels `check_mtree=YES` einen mtree-Lauf. Standardmäßig verwendet mtree dazu die Datenbanken in /etc/mtree/. Hier können Sie eigene Datenbanken als VERZEICHNIS.secure hinzufügen.

Wichtig ist das die Ursprungs-Datenbank gesichert wird, dies kann durch kryptographische Signaturen oder Verschlüsselung gesehen, oder indem Sie die Datenbank auf ein schreibgeschütztes Medium kopieren.

Sollte ein Einbrecher bereits in das System eingedrungen sein, kann er beliebig

[!h]

```
$ mtree -L -p / -X /etc/mtree.excl -f /root/mtree.orig
3:      modification time (Wed Feb  9 18:49:07 2005, Wed Feb  9 18:55:16 2005)
3/1:    modification time (Wed Feb  9 18:36:51 2005, Wed Feb  9 18:54:47 2005)
3/2:    modification time (Wed Feb  9 18:36:51 2005, Wed Feb  9 18:54:48 2005)
3/3:    size (6460, 6489)
        modification time (Wed Feb  9 18:48:17 2005, Wed Feb  9 18:54:52 2005)
        rmd160 (0xf30de94ae188d9114a00118e80bbedf8e2c6c6ed, 0x6a5b3781799b09e2b8d8037e89fe03957ebca247)
        sha1 (0x4c499470dcaed01c2681eb0676ddac8cf3024d5a, 0xc04159927a4e9ca5f521161d2cc8daa019fca19a)
extra: 3/new1
extra: 3/new2
missing: ./3/7
missing: ./3/8
missing: ./3/9
```

Listing 10.3: Prüflauf mit mtree

Programme austauschen. Dies trifft auch auf mtree zu, so das Sie sicherheitshalber auch die mtree-Dateien signieren oder auf eine CD brennen können. Noch besser ist es, ein verdächtiges System einfach von einem anderen, vertrauenswürdigen System, bspw. einer Live-CD aus zu prüfen.

10.2 Dateisystemintegrität mit Aide prüfen

Aide funktioniert ähnlich wie Mtree als Dateisystemintegritätsprüfer, indem eine existierende Datenbank von Dateiattributen mit dem aktuellen System verglichen wird.

Aide durchläuft dazu die vorher definierten Verzeichnisse und erstellt besagte Datenbank mit verschiedenen, ebenfalls übergebenen Optionen und Dateiattributen.

Diese Optionen sind : (Regelzeichen und Bedeutung)

p Dateirechte

i INODE

n Anzahl der Links auf die Datei

u Nutzer

g Gruppe

s Größe

b Block

m mtime, Erstellzeitpunkt

a atime, Zugriffszeitpunkt

c ctime, Veränderungszeitpunkt

s auf wachsende Größe prüfen

md5 MD5-Prüfsumme

sha1 SHA1-Prüfsumme

rmd160 rmd160-Prüfsumme

tiger Tiger-Prüfsumme

r p+n+i+u+g+s+m+c+md5

l p+n+i+u+g

e leere Gruppe

> p+u+g+i+n+S

Die ersten Optionen (p - S) sind selbsterklärend, da es aber möglich ist diese Daten zu fälschen, (z. B. mittels Kernelmodifikation oder Rootkit) verwendet Aide noch weitere Möglichkeiten, nämlich 4 verschiedene Prüfsummenverfahren (md5, sha1, rmd160, tiger). Die Geschwindigkeitunterschiede der Algorithmen sollten sich auf normalen Heimsystemen in Grenzen halten, auf einem »großen« Produktionsserver können sich durchaus Unterschiede bemerkbar machen, es empfiehlt sich hier einfach einen Testlauf zu machen und zu sehen welcher Algorithmus am schnellsten ist (ich tendiere in Richtung SHA1 und RMD160). Man kann die Prüfverfahren beliebig kombinieren und so mehrere Prüfsummendatenbanken erstellen, welche zu fälschen unmöglich sein sollte, oder verschiedene Konfigurationen erstellen, die per cron durchlaufen, z. B. einen einfachen Dateiattributecheck bei jedem Start und nachts oder am Wochenende einen Durchlauf mit allem Prüfsummen.

Listing 10.4 zeigt wie Aide eine neue Datenbank generiert, die standardmäßig unter `/usr/pkg/etc/aide.db.new` abgelegt wird. Vorgaben entnimmt Aide der Konfigurationstdatei `/usr/pkg/etc/aide.conf`. Anschließend wird mit `-C` die Datenbank

geprüft und ein Bericht generiert, mit -u wird die Vergleichsdatenbank aktualisiert und mit -c eine alternative Konfiguration angegeben.

[!h]

```
$ aide -i
$ aide -C
$ aide -u -c foo.bar
```

Listing 10.4: Einsatz von Aide

[!h]

```
database = file:///etc/aide.db
% Speicherort der Datenbank im URL-Format (also % file:///pfad/zur/datei)

database_new = file:///etc/aide.db.new
% Speicherort der zu erstellenden Datenbank

report_url = file:///root/aide.report
% Speicherort des generierten Reports, kann auch per -r übergeben werden

heavy = p+i+n+g+u+s+S+rmd160+md5
% Definition des "Makros" heavy, es können beliebige Optionen übergeben werden

/bin heavy
% Das Verzeichnis /bin soll mit den Optionen heavy geprüft werden

/dev u+g+i+s+S
% Das Verzeichnis /bin soll mit den Optionen u g i s S geprüft werden

!/var/spool/
% Das Verzeichnis /var/spool soll nicht geprüft werden
% Es bietet sich an Verzeichnisse die sich oft ändern nicht (zu
% paranoid) zu prüfen.
```

Listing 10.5: Optionen in aide.conf

Kleines Beispiel aus einer Datenbank:

[!h]

```
# This file was generated by Aide, version 0.6
# Time of generation was 2002-03-01 16:21:17
@@db_spec name uid gid size inode perm count md5 rmd160
/* verwendete Attribute */ /usr 0 0 512 2 40755 14 0 0
/usr/bin 0 0 4608 26368 40755 2 0 0
/usr/bin/Mail 0 0 110880 26369 100555 3 imGFDbIE2vmgcor1KxliHg== wz4...
/usr/bin/apply 0 0 10416 26370 100555 1 Cm8vqBULhAszAtgvD2idAQ== o2R...
/usr/bin/apropos 0 0 13040 26371 100555 1 K2mwO7ixww/UHU7l245QZA== k...
/usr/bin/at 0 0 32216 26372 104555 4 NowA6KW8osFIQ5sivNVShA== Hpi7Jp...
/usr/bin/atq 0 0 32216 26372 104555 4 NowA6KW8osFIQ5sivNVShA== HpiZJ...
/usr/bin/atrm 0 0 32216 26372 104555 4 NowA6KW8osFIQ5sivNVShA== HpdZ...
```

Listing 10.6: Beispiel aus einer Aide-Datenbank

10.3 Dateisystemintegrität mit tripwire(1) prüfen

Tripwire kann neben Mtree und Aide ebenfalls dazu eingesetzt werden Manipulationen an Dateien aufzuspüren. Aide verwendet die selben Prinzipien wie Tripwire, da es Tripwire zum Vorbild nahm. Zum Entstehungszeitpunkt von Aide war Tripwire noch nicht unter der GPL freigegeben und heute existieren weiterhin verschiedene Tripwire-Versionen, welche als proprietäre Software vertrieben werden.

10.4 Prüfsummen einzelner Dateien erstellen

Um einzelne Dateien zu vergleichen reicht ein einfaches Prüfsummenverfahren wie sha1(1) oder md5(1) aus. Dabei erstellt das Programm eine Prüfsumme, die die Integrität der Datei sicherstellt. Somit können Sie einfach testen ob die Datei verändert wurde oder ob sie bspw. korrekt durch ein Netzwerk übertragen wurde. Allerdings wird hierbei nicht die Integrität der Signatur gesichert, so das diese vor Manipulation geschützt werden muss. Kann also bei der Datenübertragung nicht ausgeschlossen werden, das Datei und Prüfsumme ausgetauscht werden können, müssen Sie die Datei oder die Prüfsumme mit einer kryptographischen Signatur versehen. Tabelle 10.1 zeigt die vorhandenen Prüfsummen, deren Größe und den Zeitaufwand für eine 206MB große Datei. In den letzten Jahren sind vermehrt Angriffe auf die kryptographische Stärken von MD5 und SHA1 aufgetreten. In den nächsten Jahren werden diese Verfahren höchstwahrscheinlich komplett gebrochen werden. Bis dahin empfiehlt es sich aus Sicherheitsgründen, wenn eine Prüfsumme eben nicht nur zum Prüfen eines sicheren Übertragungsweges gebraucht wird, mehrere verschiedene Prüfsummen zu erstellen.

Prüfsumme	Bit	Befehl	Zeit†
sum	16 Bit	cksum -o 1	4,6s
cksum	16 Bit	cksum -o 2	1,9s
MD2	128 Bit	md2(1)	127,8s
MD4	128 Bit	md4(1)	2,6s
MD5	128 Bit	md5(1)	3,3s
SHA1	160 Bit	sha1(1)	4,2s
Ripe-MD160	160 Bit	rmd160(1)	5,4s

† mit time(1) gemessene Zeit um eine Prüfsumme für eine 206MB große Dump-Datei zu erstellen

Tabelle 10.1.: eingebaute Prüfsummen

[!h]

```
$ md5 dump0-060917.asc > MD5
$ md5 -c MD5
$ dmesg > test
$ sha1 test > SHA1
$ sha1 -c SHA1
$ dmesg » test
$ sha1 -c SHA1
(SHA1) test: FAILED
$
```

Listing 10.7: Dateien mit MD5 und SHA1 vergleichen

Der erste Test gelingt, der zweite schlägt fehl, da der Inhalt der Datei geändert wurde.

10.5 Den Rechner mit Chkrootkit auf Rootkits prüfen

Chkrootkit[PP2] ist ein Paket, mit dem man einen Rechner nach Rootkits untersuchen kann. Es funktioniert prinzipiell wie ein Virenscanner, indem es Binärdaten auf Spuren von Rootkits untersucht.

Es besteht aus folgenden Programmen:

- `chkrootkit`: Ein Shellskript das System-Befehle auf Rootkits prüft
- `ifpromisc.c`: Prüft ob die Netzwerkschnittstelle im Promiscous-Modus ist
- `chklastlog.c`: Prüft auf Manipulationen in den Login-Protokollen von lastlog(5)

- `chkwtmp.c`: Prüft auf Manipulationen in den wtmp-Login-Protokollen von last(1) und ac(1)
- `check_wtmpx.c`: Prüft auf Manipulationen in den wtmpx-Login-Protokollen (nur Solaris)
- `chkutmp.c`: Prüft auf Manipulationen in den utmp-Login-Protokollen
- `chkproc.c`: Sucht nach LKM-Trojanern
- `chkdirs.c`: Sucht nach LKM-Trojanern
- `strings.c`: Einfacher strings(1)-Ersatz

Ruft man Chkrootkit ohne Optionen auf, testet es alle System-Befehle auf Rootkit-Anzeichen. Die Option `-l` zeigt alle Testoptionen an, `-r` übernimmt ein Verzeichnis als Wurzelverzeichnis, das zu prüfen ist. Nützlich vor allem wenn man eine verdächtige Systemplatte von einem anderen System aus testen möchte. Mit `-p` kann man das Verzeichnis für alle benötigten Systembefehle wie grep(1) oder ps(1) angeben, wenn man bspw. eine saubere CD für Arpwatch verwenden möchte. Mit `-x` zeigt Chkrootkit alle verdächtigen Dateien an, so das man sie hänisch überprüfen kann.

Da Chkrootkit wie ein Virenscanner arbeitet kann es nur Rootkits finden, deren Signaturen es kennt. Außerdem können Rootkits auch Routinen enthalten, um ihre Enttarnung durch Chkrootkit zu verhindern. Daher sollte man verdächtige Systeme durch ein sauberes System, bspw. einer LiveCD untersuchen.

Eine Liste von bekannten Rootkits findet sich unter http://www.chkrootkit.org/#tests.

10.6 Das Netzwerk auf Zombies untersuchen

Das Paket Ddos-scan [PP19] überprüft Rechner in einem Netzwerk auf sogenannte Zombies. Das sind Agenten von Botnetzen, die die Kontrolle über befallene Rechner übernehmen und sie so in die Gewalt von Einbrechern bringen. Ein bekanntes Beispiel ist Stacheldraht. Heutzutage werden hauptsächlich Rechner eines minderqualitativen Betriebssystems aus Redmond angegriffen und für den Spam-Versand mißbraucht. NetBSD-Rechner werden bisher nicht angegriffen, obwohl dies prinzipiell möglich ist. Dennoch kann man mit diesem System heterogene Netzwerke überprüfen.

Siehe auch:
(1999)
(2000)

[!h]

```
# dds -v -d 127.0.0.1
No mask passed, assuming host scan (/32)
Mask: 32
Target: 127.0.0.1
dds $Revision: 1.10 $ - scanning...

Probing address 127.0.0.1
Sending ICMP to: 127.0.0.1
Sending tfn ICMP to: 127.0.0.1
Sending UDP to: 127.0.0.1
Listener got an ICMP packet
Waiting 30 seconds for late responses.
Listener got an ICMP packet
Listener got an ICMP packet
#
```

Listing 10.8: Beispiel eines Ddos-Scan-Laufs

11

NETZWERKE

Erfahrung ist verstandene Wahrnehmung.

(Immanuel Kant)

11.0.1 *TCP-Wrapper*

11.1 NETZWERKVERKEHR MIT OPENSSH VERSCHLÜSSELN

11.1.1 *Konfiguration des Dæmons*

AFSTOKENPASSING
: Gibt an, ob ein AFS-Token zum Server weitergeleitet werden soll.

ALLOWGROUPS
: Spezifiziert Gruppen die sich einloggen dürfen. Wenn aktiviert, dürfen sich auch nur diese Gruppen einloggen. Joker wie *?* und * sind erlaubt, GIDs sind nicht gestattet.

ALLOWTCPFORWARDING
: Erlaubt die Weiterleitung von TCP-Ports

ALLOWUSERS
: Spezifiziert Benutzer die sich einloggen dürfen. Wenn aktiviert, dürfen sich auch nur diese Benutzer einloggen. Joker wie *?* und* sind erlaubt, UIDs sind nicht gestattet.

AUTHORIZEDKEYSFILE
: Spezifiziert die Datei, die den öffentlichen Schlüssel eines Benutzers enthält. Joker der folgenden Form sind erlaubt:
%% = %

%h = Home-Verzeichnis
%u = Username
Standardeinstellung ist /.ssh/authorized_keys.

BANNER
Banner gibt den Text in der angegebenen Datei vor dem Einloggen aus. Kann teilweise notwendig sein um z. B. juristische Warnungen auszugeben. Nur in Protokoll 2 verfügbar, standardmäßig abgeschaltet.

CHALLENGERESPONSEAUTHENTICATION
Erlaubt »Challenge Response«-Authentifizierung, Standard ist ja. Alle Authentifizierungsmechanismen aus `man 5 login.conf` werden unterstützt.

CIPHERS
Eine kommaseparierte Liste der Kryptoalgorithmen für SSH-Protokoll 2. Standard ist »aes128-cbc, 3des-cbc, blowfish-cbc, cast128-cbc, arcfour, aes192-cbc, aes256-cbc".

CLIENTALIVEINTERVAL
Zeitintervall der Inaktivität, nachdem der Dæemon überprüft, ob der Client noch lebt. »0« bedeutet keine Überprüfung, existiernt nur in Protokoll 2.

CLIENTALIVECOUNTMAX
Anzahl der Überprüfungen des Clients (siehe *ClientAliveInterval*) bevor der Client gekappt und die Sitzung beendet wird. Die ClientAlive-Meldungen werden im verschlüsselten SSH-Kanal übertragen und sind somit im Gegensatz zu KeepAlive nicht fälschbar.
Standard ist 3. Beispiel: ClientAliveInterval auf 15 und ClientAliveCountMax auf 3 bedeutet ein Ende der Session nach $3 * 15 = 45$ Sekunden Inaktivität.

DENYGROUPS
Verbietet Benutzern das Einloggen welche primär oder sekundär in der angegebene Gruppen sind. Joker sind erlaubt, GIDs nicht.

DENYUSERS
Verbietet bestimmten Benutzern das Einloggen. Joker sind erlaubt, UIDs nicht. Die Form »Benutzer@Rechner« kann verwendet werden, um bestimmte Benutzer von bestimmten Rechnern aus auszuschließen.

PUBKEYAUTHENTICATION
Spezifiziert ob Authentifizierung mittels kryptographische Schlüssel erlaubt ist. Standard ist ja, funktioniert nur in Protokoll 2

GATEWAYPORTS
Spezifiziert ob sich entfernte Rechner an weitergeleitete TCP-Ports des Clients

binden dürfen.

HOSTKEY *Parameter: -h*
Gibt die Datei an, die die privaten Server-Schlüssel enthält. Standard ist /etc/ssh_host_key. Sshd verwendet keine Dateien die Gruppen- oder Welt-Zugriffe erlauben. Mehrere Dateien können angegeben werden.

IGNORERHOSTS
.rhosts- und .shosts-Dateien sollen ignoriert werden, /etc/hosts.equiv und /etc/shosts.equiv werden trotzdem ausgewertet. Standard ist ja.

IGNOREUSERKNOWNHOSTS
Spezifiziert ob Sshd /.ssh/known_hosts des Benutzers während der RhostsRSAAuthentication auswerten soll. Standard ist Nein.

TCPKEEPALIVE
Gibt an ob TCPKeepAlive-Meldungen gesendet werden sollen. Ermöglicht die Überprüfung der Verbindung und tötet die Sitzung wenn die Verbindung abbricht. Verhindert somit Geisterprozeße Standard ist ja, zum abschalten müssen Client und Server auf nein stehen.

KERBEROSAUTHENTICATION
Spezifiziert ob Kerberos-Authentifizierung erlaubt ist. Standard ist ja, der Server benötigt eine Kerberos-Servtab welche die Authentifizierung der KDC-Identität erlaubt. Die Identifizierung erfolgt mittels Kerberos-Ticket oder -Passwort.

KERBEROSGETAFSTOKEN
Wenn AFS verwendet wird und der Benutzer ein Kerberos-5-TGT hat, soll ein AFS-Token bezogen werden bevor in das Heimverzeichnis des Nutzers gesprungen wird. Standard ist nein.

KERBEROSORLOCALPASSWD
Ermöglicht die Authentifizierung des Benutzers durch z. B. /etc/passwd wenn die Kerberosauthentifizierung fehlschlägt. Standard ist ja.

KERBEROSTICKETCLEANUP
Gibt an ob das User-Ticket beim Ausloggen vernichtet werden soll. Standard ist ja.

KEYREGENERATIONINTERVAL *Parameter -k*
Zeit in Sekunden, nach der der private Server-Schlüssel regeneriert wird. Standard ist 3600 Sekunden. Dies soll verhindern, das aufgezeichnete Sitzungen mit einem gestohlenen Serverschlüssel im nachhinein entschlüsselt werden können. Der Schlüssel wird nicht gespeichert.

LISTENADDRESS
Definiert die Adresse an die sich der Sshd binden soll. Standardmäßig bindet sich der Sshd an alle Netzwerkschnittstellen. Mehrere Angaben sind möglich.

LOGINGRACETIME *Parameter: -g*
Der Server beendet ein Login, wenn innerhalb dieser Zeit keine Authentifizierung stattgefunden hat. Standard sind 600 Sekunden, 0 bedeutet keine Begrenzung.

LOGLEVEL
Auskunftsfreudigkeit des Sshd wenn mitgeloggt wird.
Optionen: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2 und DEBUG3.
Standard ist INFO. DEBUG verletzt die Privatsphäre der Nutzer und ist nicht empfohlen.

MACS
Gibt die MACs (Message-Authentication-Code-Algorithmen, Prüsummenverfahren bzw. Hash) an. Dies wird in Protokoll 2 benutzt um die Integrität der Übertragung zu schützen. Mehrere Algorithmen können per kommagetrennter Liste angegeben werden. Standard ist: »hmac-md5, hmac-sha1, hmac-ripemd160, hmac-sha1-96, hmac-md5-96«.

MAXSTARTUPS
Gibt die maximale Anzahl von konkurrierenden und unauthentifizierten Verbindungen zum Sshd an. Zusätzliche Verbindungen werden verworfen. Standard ist 10.

PASSWORDAUTHENTICATION
Gibt an ob die Authentifizierung via Passwort erlaubt ist. Standard ist ja, gilt für beide Protokolle.

PERMITEMPTYPASSWORDS
Erlaubt leere Passwörter. Standard ist nein.

PERMITROOTLOGIN
Gibt an ob sich root per Ssh einloggen kann. Standard ist ja, Mögliche Argumente sind: yes - login erlaubt
without-password - Password-Authentifizierung abgeschaltet für root
forced-commands-only - root-Login mit Public-Key-Authentifizierung ist erlaubt, aber nur wenn der Command-Modus-übergeben wurde, d.h. ein bestimmter Befehl ausgeführt werden soll (nützlich um z. B. Backups zu erlauben).

PIDFILE
Gibt die Datei an, die die Prozeß ID (PID) des Sshd enthält. Standard ist

/var/run/sshd.pid

PORT *Parameter: -p*
Gibt den Port an, an den sich der Sshd binden soll. Standard ist 22, mehrere Ports können übergeben werden.

PRINTMOTD
Gibt an ob sshd beim Login /etc/motd ausgeben soll. Wird auf manchen System bereits von der Shell übernommen Standard ist ja.

PROTOCOL
Gibt die Protokollversionen an, Standard ist 1, 1,2 erlaubt 1 und 2

PUBKEYATHENTICATION
Erlaubt das Login per Schlüssel.

REVERSEMAPPINGCHECK
Sshd überprüft ob der Clienthostname auf die übertragene IP passt. Standard ist nein.

RHOSTSAUTHENTICATION
Erlaubt rhosts-Authentifizierung mittels .rhosts oder /etc/hosts.equiv Standard ist nein.

RHOSTSRSAAUTHENTICATION
Erlaubt rhosts-Authentifizierung mittels .rhosts oder /etc/hosts.equiv und RSA-Host-Authentifizierung Standard ist nein.

RSAAUTHENTICATION
Gibt an das reine RSA-Authentifizierung erlaubt ist. Standard ist ja, betrifft nur Protokoll 1.

SERVERKEYBITS
Gibt die Größe der Serverschlüssels in Bit an. Minimum ist 512, Standard ist 768.

STRICTMODES
Gibt an ob der Sshd Besitzer und Modi der Konfigurationsateien überprüfen soll. Anfänger lassen ihre Dateien manchmal offen für die Welt. Standard ist ja.

SUBSYSTEM
Spezifiziert ein externes Subsystem (z. B. Secure-FTP). Optionen sind ein Subsystem-Name und der Befehl der bei einer Anfrage ausgeführt werden soll.
Beispiel sftp: »Subsystem sftp /usr/libexec/sftp-server«

SYSLOGFACILITY
: Übergibt den Bezeichner des loggenden System an sylogd.
Optionen sind DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7 Standard ist AUTH.

USELOGIN
: Login(1) soll für interaktive Logins benutzt werden. Login wird nicht für entfernte Befehle verwendet. Standard ist Nein

X11DISPLAYOFFSET
: Spezifiziert die erste Displaynummer für X11-Weiterleitung um Probleme mit echten X11-Servern zu vermeiden. Standard ist 10.

X11FORWARDING
: Gibt an ob X11-Weiterleitung erlaubt werden soll. Standard ist nein. Das Verbieten erhöht die Sicherheit nicht da die User eigene Forwarder installieren können.

XAUTHLOCATION
: Gibt den Ort des xauth(1)-Programms am. Standard ist /usr/X11R6/bin/xauth.

11.1.2 *Konfiguration des Clients*

HOST
: Nachfolgend angegebene Regeln gelten nur für den angegebenen Host. Mit mehrerenen Host-Parametern können mehrere Hosts verschieden konfiguriert werden.

AFSTOKENPASSING
: Siehe Kap. 11.1.1, »Konfiguration des Dæmons« auf Seite 147 ff.

BATCHMODE
: Wenn auf ja gesetzt, wird kein Passwort abgefragt. Nützlich in Shellskripten vor allem in Verbindung mit schlüsselbasierter Authentifikation.

CHECKHOSTIP
: Wenn aktiviert wird ssh zusätzlich die IP Adresse mit der in known_hosts vergleichen. Dies ermöglicht es gespoofte Verbindungen anhand des Keyfingerprints zu erkennen.

CIPHER

Zu verwendende Verschlüsselungsalgorithmen in Protokoll 1

CIPHERS
Zu verwendende Verschlüsselungsalgorithmen in Protokoll 2

COMPRESSION
Spezifiert ob Kompression eingeschaltet werden soll.

COMPRESSIONLEVEL
Komprimierungsrate falls Komprimierung eingeschaltet ist. Rate geht von 1 (wenig und schnell) bis 9 (hoch und langsam).

CONNECTIONATTEMPTS
Anzahl der Verbindungsversuche (1 pro Sekunde) bevor auf RSH zurückgeschaltet oder abgebrochen wird. Nützlich vor allem in Scripts.

PUBKEYAUTHENTICATION
Gibt an ob die Authentifizierung per Schlüssel erfolgen soll, betrifft nur Protokoll 2.

ESCAPECHAR
Definiert das Fluchtzeichen. Standard: ' ', möglichst genau ein Zeichen. Anpassungen können teilweise für Binär-Daten notwendig sein.

FALLBACKTORSH
Spezifiziert ob auf RSH (unverschlüsselt!) zurückgegangen werden soll falls die Ssh-Verbindung nicht zustande kommt.

FORWARDAGENT
Spezifiziert ob die Verbindung zum Authentication-Agent zur entfernten Maschine weitergeleitet werden soll.

FORWARDX11
Spezifiziert ob X11-Verbindungen automatisch über die sichere Verbindung getunnelt werden sollen.

GATEWAYPORTS
Spezifiziert ob sich entfernte Maschinen an lokal weitergeleitete Ports hängen dürfen.

GLOBALKNOWNHOSTSFILE
Definiert eine andere Datei statt /etc/ssh_known_hosts.

HOSTKEYALIAS
Definiert ein Alias das anstelle des echten in known_hosts verwendet wird.

Nützlich für das tunneling oder wenn mehrere Server auf einem Host laufen.

HOSTNAME
Definiert einen Hostnamen zum Log in z. B. als Spitznamen oder Abkürzung.

IDENTITYFILE
Definiert die Datei, aus der die RSA-Identität des Benutzer ausgelesen wird. Standard ist /.ssh/identity, zusätzlich können Identitäten des Authentication-Agents angegeben werden. » « kann benutzt werden, ebenso können mehrere Dateien angegeben werden, die der Reihe nach durchsucht werden.

LOCALFORWARD
Definiert einen TCP/IP-Port der von der lokalen zur entfernten Maschine weitergeleitet wird. Mehrere Ports können weitergeleitet werden. Format ist: lokalerPort:entfernteMaschine:entfernterPort , nur root kann privilegierte Ports weiterleiten.

LOGLEVEL
Siehe Kap. 11.1.1, »Konfiguration des Dæmons« auf Seite 147 ff.

MACS
Siehe Kap. 11.1.1, »Konfiguration des Dæmons« auf Seite 147 ff.

NUMBEROFPASSWORDPROMPTS
Anzahl der Passwortabfragen vor Abbruch. Standard ist 3.

PASSWORDAUTHENTICATION
Spezifiziert ob Password-Authentifikation erlaubt ist. Betrifft Protokoll 1 & 2.

PORT
Gibt den Port am entfernten Host an, den Ssh verwenden soll, um den Sshd zu kontaktieren. Standard ist 22, mehrere Ports können übergeben werden.

PROTOCOL
Gibt die Protokollversionen an, Standard ist *2,1*, d.h. es wird erst 2 probiert und bei Bedarf auf 1 zurückgefallen.

PROXYCOMMAND
Definiert einen Befehl der benutzt wird um zum Server zu verbinden. Die gesamte Zeile wird als Befehl an /bin/sh übergeben. %h und %p werden als entfernter Host und Port interpretiert. CheckHostIP ist nicht verfügbar wenn mit ProxyCommand gearbeitet wird.

REMOTEFORWARD
Leitet einen TCP/IP-Port der entfernten Maschine über den verschlüsselten

Kanal zum angegeben Host:Port der lokalen Maschine. Mehrere Ports können angegeben werden, Format ist entfernterPort:Host:lokalerPort.

Forwarded einen TCP/IP Port der entfernten Maschine über den verschlüsselten Kanal zum angegeben Host:Port der lokalen Maschine, mehrere Ports können angegeben werden, Format ist entfernterPort:Host:lokalerPort

RHOSTSAUTHENTICATION
Gibt an ob rhosts-basierte Authentifikation benutzt werden soll. Betrifft nur den Client, wenn der Server kein rhost zuläßt wird dies nicht benutzt.

RHOSTSRSAAUTHENTICATION
Gibt an ob RSA-rhosts-basierte Authentifikation benutzt werden soll. Betrifft nur den Client, wenn der Server kein rhost zuläßt wird dies nicht benutzt.

RSAAUTHENTICATION
Gibt an ob RSA-Authentifizierung benutzt werden soll. Wird nur benutzt, wenn die Identitätsdatei dazu existiert und betrifft nur Protokoll 1.

CHALLENGERESPONSEAUTHENTICATION
Gibt an ob Response-Authentication benutzt werden soll.

STRICTHOSTKEYCHECKING
Wenn aktiviert, fügt ssh keine bekannten Schlüssel zu HOME/.ssh/known_hosts und HOME/.ssh/known_hosts2 hinzu und verweigert die Verbindung zu Hosts deren Schlüssel geändert wurden. Wenn auf »ask« gesetzt muß der Benutzer neue Schlüssel bestätigen, bevor sie hinzugefügt werden.

USEPRIVILEGEDPORT
Gibt an ob ein privilegierter Port für ausgehende Verbindungen genutzt werden soll. Wenn deaktiviert können einige ältere Server mit RhostsAuthentication und RhostsRSAAuthentication nicht mehr kontaktiert werden.

USER
Definiert den Benutzernamen, als der man sich einloggen will. Sehr nützlich wenn man mehrere Maschinen mit verschiedenen Logins hat.

USERKNOWNHOSTSFILE
Definiert die Datei die anstelle von /.ssh/known_hosts verwendet wird

USERSH
Erlaubt rsh/rlogin für diesen Host.

XAUTHLOCATION
Pfad des xauth(1)-Programms. Standard ist /usr/X11R6/bin/xauth.

11.1.3 *OpenSSH benutzen*

11.1.3.1 *Schlüsselbasierte Authentifikation*

Mit OpenSSH kann man neben Passwörtern auch kryptographische Schlüssel zur Authentifikation verwenden. Die Schlüssel-Datei kann mit einem Passwort gesichert werden, muss sie aber nicht. So kann man Loginmöglichkeiten generieren, die bspw. in Skripten verwandt werden. Hierzu muss auf dem Client ein Schlüssel erstellt werden, dessen öffentlicher Teil auf den Server übertragen werden muss. Dabei kann man verschiedene Schlüssel-Typen verwenden, RSA1 für RSA im Protokoll 1 und RSA oder DSA für Protokoll 2. Ich empfehle den Einsatz von DSA-Schlüsseln der Länge 1024. RSA mit 1024 Bit sollte nicht mehr als sicher betrachtet werden, 2048 Bit sind empfehlenswert, so man denn RSA überhaupt noch einsetzen muss.

[!h]

```
stefan@local $ ssh-keygen -b 1024 -t dsa
stefan@local $ scp .ssh/id_dsa.pub stefan@remote:/home/stefan/.ssh
stefan@local $ ssh stefan@remote
stefan@remote $ cd .ssh && cat id_dsa.pub » authorized_keys
stefan@remote $ chmod 600 authorized_keys ; rm id_dsa.pub
```

Listing 11.1: SSH mit Schlüsselauthentifikation

11.1.3.2 *Verschlüsselte Tunnel*

OpenSSH kann dazu verwandt werden, verschlüsselte Netzwerktunnel zu entfernten Rechnern zu erstellen. Dies ist nützlich, wenn ungesicherte Verbindungen (SMTP, POP, NNTP, IMAP oder ähnliches) durch nicht-vertrauenswürdige Netze geleitet werden. Im Beispiel 11.2 werden zwei Beispiele zu SSH-Tunneln gezeigt, das erste leitet den entfernten Port 5432 von 192.168.2.5 auf den lokalen Port 5000 weiter. Das heißt, SSH loggt sich erst auf 192.168.2.1 als stefan ein, und erstellt dann die verschlüsselte Verbindung vom lokalen Rechner zu 192.168.2.5 über die 192.168.2.1. Dabei ist nur die Strecke vom lokalen Rechner zur 192.168.2.1 mit SSH gesichert, der Teil von 192.168.2.1 zur 192.168.2.5 ist nicht verschlüsselt. Das zweite Beispiel loggt sich direkt auf der 192.168.2.5 ein leitet den dortigen 127.0.0.1:5432 direkt auf den lokalen Rechner weiter.

[!h]

```
$ ssh -2 -c aes128-cbc -C stefan@192.168.2.1
                   -L 50000:192.168.2.5:5432

$ ssh -2 -c aes128-cbc -C stefan@192.168.2.5
                     -L 50000:127.0.0.1:5432

$ psql -U stefan -h 127.0.0.1 -p 50000 MyDatabase
```

Listing 11.2: PostgreSQL-Ports mit SSH tunneln

11.1.3.3 *Befehle auf dem entfernten Rechner ausführen*

SSH kann ebenso als Ersatz für RSH verwandt werden und so Befehle auf der entfernten Maschine ausführen. Listing 11.3 zeigt zwei Befehle, die alle auf der entfernten 192.168.1.1 ausgeführt werden.

[!h]

```
$ ssh stefan@192.168.1.1 "uname -mrs"
NetBSD 2.0.2 sparc64
# ssh stefan@192.168.1.1 "dump -0au -f dump0 /home"
```

Listing 11.3: Entfernte Befehle mit SSH ausführen

11.1.4 *OpenSSH härten*

STARKE PASSWÖRTER

Wenn Passwort-Authentifikation notwendig ist, müssen unbedingt starke Passwörter verwandt werden. Es existieren einige Skripte, die Netzwerke nach SSH-Dæmons abscannen und dann einfache Brute-Force-Attacken fahren. Dabei werden standardmäßig einige Benutzernamen durchprobiert wie root, administrator, www, database und auch englische Vornamen wie John, Mike und Co. Daher ist es keine schlechte Idee nicht nur diese einfachen Vornamen zu verwenden, sondern diese bspw. mit einer Jahreszahl oder einem Sonderzeichen zu erweitern.

ROOT-LOGIN VERBIETEN

Standardmäßig ist das Login für den Benutzer root verboten. Dies sollte auf keinen Fall geändert werden.
Option: `PermitRootLogin no`

STANDARDPORT ÄNDERN

Der Standard-Port für SSH ist der Port 22. Jeder automatisierte Angriff wird sich zuerst gegen diesen Port richten. Man sollte ihn daher, sofern möglich, ändern.
Option: `Port 12345`

NETZWERKSCHNITTSTELLE FESTLEGEN

Mit der Option `ListenAddress` kann man die Netzwerkschnittstelle festlegen, an die sich der SSH-Dæmon binden soll. Hat man mehrere Netzwerkkarten, sollte man SSH nur an die benötigten binden.

LÖCHER IN DER FIREWALL VERHINDERN

Standardmäßig kann man mit SSH entfernte Ports und X11-Dienste weiterleiten. Dies setzt die sorgsam gepflegten Firewall-Regeln außer Kraft. Daher sollte man die Weiterleitung von TCP/IP-Ports und X11 abschalten.

```
AllowTcpForwarding no
X11Forwarding no
```

BENUTZER UND GRUPPEN AUSSPERREN

Man kann mit den Optionen »AllowUsers«, »DenyUsers«, »AllowGroups« und »DenyGroups« einstellen welche Benutzer und Gruppen sich (nicht) einloggen dürfen. Davon sollte man Gebrauch machen und wenn möglich unerwünschte Benutzer explizit aussperren. Man kann dabei auch explizit Hostnamen übergeben, so das sich ein Operator bspw. nicht von Windows-Rechner aus einloggen kann.

PROTOKOLL 1 VERBIETEN

Das SSH-1 Protokoll ist bereits seit Jahren unsicher. Durch den verwandten scwachen MAC-Algorithmus ist es möglich bestehende Verbindungen zu übernehmen. Daher muss SSH-1 unbedingt verboten werden. Die Option »Protocol« in der sshd_config spezifiziert die zu verwendeten Protokolle, es genügt hier nur die 2 zu übergeben.

NETZBEREICHE EINGRENZEN

Leider gibt es einige der oben beschriebenen Skripte, die insbesondere oft aus China, Süd-Korea, Taiwan oder den USA abgefeuert werden. In Deutschland ist insbesondere das T-Online-Netz in meinem Logs unangenehm aufgefallen. Kann man daher mit einem Paketfilter wie ipf(8) das Netzwerk eingrenzen, das sich mit dem SSH-Dæmon verbinden darf, sollte man dies unbedingt tun. Es grenzt die Anzahl der Skript-Kiddie-Versuche stark ein und hält die Logs sauber.

SSHD-PRIVILEGIEN EINSCHRÄNKEN

Normalerweise wird SSH an Port 22 gebunden und als root gestartet. Sollte es

nun gelingen SSH zu übernehmen, hat der Einbrecher auch root-Rechte erlangt. Daher ist es ein großer Schritt für die Sicherheit des Systems, wenn der sshd nicht als root gestartet wird. Dazu kann bspw. systrace (Kap. 8.1, »Systemaufrufe mit Systrace steuern« auf Seite 101 ff.) verwandt werden oder man aktiviert mit »UsePrivilegeSeparation« den Privilege-Separation-Mechanismus im sshd. Dies führt dazu das der sshd während der Authentifizierungsphase einen unprivilegierten Kind-Prozeß startet und nach der erfolgreichen Anmeldung ein sshd-Kind mit der Berechtigung des angemeldeten Benutzers. Somit sind keine root-Rechte mehr im Spiel.

MEHRERE SSH-DAEMONS STARTEN

Es ist zwar sehr selten das ein SSHD abstürzt, kann aber dennoch vorkommen. Da man damit in der Regel vom Server ausgesperrt ist, sollte man einen zweiten SSH-Dæmon starten, der auf einem anderen Port lauscht und als Reserve dient.

NICHT-BENÖTIGTE AUTHENTIFIZIERUNGSALGORITHMEN DEAKTIVIEREN

Alle nicht-benötigten Algorithmen sollten explizit deaktiviert werden, also bspw:

```
RhostsRSAAuthentication no
HostbasedAuthentication no
KerberosAuthentication no
RhostsAuthentication no
```

LEERE PASSWÖRTER VERBIETEN

Unbedingt leere Passwörter mit `PermitEmptyPasswords no` verbieten.

DOS MIT ZU VIELEN OFFENEN SSHD VERHINDERN

Mit der Option `MaxStartups` kann man die maximale konkurrierende Anzahl nicht-authentifizierter Verbindungen festlegen. Weitere Verbindungsversuche werden zurückgewiesen bis die Warteschlange nicht mehr voll ist.
Die etwas seltsame Option funktioniert so: `MaxStartups 10:30:60` heißt, das bei 10 offenen Verbindungen, mit einer Wahrscheinlichkeit von $\frac{30}{100}$ (30%) neue Verbindungen abgelehnt werden. Mit jeder weiteren offenen Verbindung werden neue abgelehnt, bis schließlich bei 60 offenen Verbindungen keine neuen mehr angenommen werden.

11.2 verschlüsselte IP-tunnel mit stunnel

Stunnel ist ein SSL-Verschlüsselungstunnel, der entwickelt wurde um Server, die kein SSL unterstützen, in einem SSL-Tunnel anzusprechen.

11.3 portscans mit portsentry verhindern

Portsentry ist ein Programm, das die Ports einer Netzwerkschnittstelle überwacht. Wird dabei ein Portscan erkannt, kann es die Route zum scannenden Rechner ändern und so weitere Verbindungen von dort verhindern. Konfiguriert wird es in der `/usr/pkg/etc/portsentry.conf` und der `/usr/pkg/etc/portsentry.ignore`. In die letztere Datei trägt man alle IP-Adressen oder Subnetze ein, die Portsentry ignorieren soll. In der ersten Datei werden die Konfigurationsoptinen festgelegt. Diese sind:

`TCP_PORTS`, `UDP_PORTS`
: kommaseparierte Liste von Ports, auf denen Portsentry lauschen soll.

`ADVANCED_PORTS_TCP`, `ADVANCED_PORTS_UDP`
: Jeder unbelegt Port unterhalb dieser Grenze soll überwacht werden

`ADVANCED_EXCLUDE_TCP`, `ADVANCED_EXCLUDE_UDP`
: Diese Ports werden nicht überwacht.

`RESOLVE_HOST`
: 1 aktiviert DNS-Namensauflösung, 0 deaktiviert sie

`BLOCK_UDP`, `BLOCK_TCP`
: 0 = blockiere Scan nicht, 1 = blockiere Scan, 2 = externen Befehl ausführen

`KILL_ROUTE`
: Befehl um die Route zu einem Scanner zu verändern, `/sbin/route add \$ TARGET\$ 127.0.0.1` funktioniert

`SCAN_TRIGGER`
: Anzahl von Verbindungsversuchen, nach denen Alarm ausgelöst wird

Mit dem externen Befehl kann man bspw. auch eine Email abschicken oder den Paketfilter anpassen.

Die Konfiguration des Wächters muss zwingend sorgsam geprüft werden, da man sich sonst sehr leicht selbst ausschließt, oder Kunden den Zugang zum Netz ver-

wehrt.

11.4 Rechnerpräsenz in Netzwerken mit Arpwatch überwachen

Arpwatch überwacht Ethernet und IP-Adressen und meldet Änderungen an Syslog und / oder per Email.

Die Optionen von Arpwatch sind dabei:

-d Startet Arpwatch im Debug-Modus und sendet so keine Mails

-f datei Übergibt die Adressdatenbank

-i Setzt die Netzwerkschnittstelle

-n setzt zusätzliche Netzwerke um Meldungen über »Bogonen« zu vermindern

-n Deaktiviert die Meldungen über »Bogonen«

-r Spielt eine mit tcpdump(8) bzw. pcapture gesicherte Datei ab

Arpwatch gibt folgende Statusmeldungen auf der Konsole aus:

new activity Neue Adresse (jünger als sechs Monate) gefunden

new station Neue Adresse gefunden

flip flop Die Ethernet-Adresse ist von der am häufigsten gesehenen zur zweithäufigsten abgestiegen

changed ethernet address Neue Ethernet-Adresse für einen Rechner

Arpwatch gibt folgende Statusmeldungen an Syslogd:

ethernet broadcast
: Die MAC eines Rechners ist eine Broadcast-Adresse

ip broadcast
: Die IP eines Rechners ist eine Broadcast-Adresse

bogon
: Die Quell-IP ist nicht im lokalen Subnetz

ETHERNET BROADCAST
: Ein Broadcast von der Quelle

ETHERNET MISMATCH
: Die MAC-Ethernet-Adresse der Quelle stimmte nicht mit der Adresse des ARP-Paketes überein

REUSED OLD ETHERNET ADDRESS
: Die Ethernet-Adresse ist von der am häufigsten gesehenen Adresse zur drittenhäufigsten (oder weniger) abgestiegen

SUPPRESSED DECNET FLIP FLOP
: Flip-Flop wurde unterdrückt, eine Adresse war im DECnet

[!h]

```
From: arpwatch (Arpwatch)
To: root
Subject: changed ethernet address

            hostname: <unknown>
          ip address: 192.168.2.1
    ethernet address: 0:4:75:ae:50:53
     ethernet vendor: 3 Com Corporation
old ethernet address: 0:a:5e:5:65:eb
 old ethernet vendor: <unknown>
           timestamp: Tuesday, September 6, 2005
                          20:29:46 +0200
  previous timestamp: Tuesday, September 6, 2005
                                20:27:41 +0200
               delta: 2 minutes

From: arpwatch (Arpwatch)
To: root
Subject: flip flop

            hostname: <unknown>
          ip address: 192.168.2.1
    ethernet address: 0:a:5e:5:65:eb
     ethernet vendor: <unknown>
old ethernet address: 0:4:75:ae:50:53
 old ethernet vendor: 3 Com Corporation
           timestamp: Tuesday, September 6, 2005
     20:29:46 +0200
  previous timestamp: Tuesday, September 6, 2005
                                20:29:46 +0200
               delta: 0 seconds
```

Listing 11.4: Auszug aus einem Arpwatch-Protokoll

11.5 Netzwerkverkehr mit tcpdump überwachen und analysieren

Tcpdump ist ein Programm um den gesamten Netzwerkverkehr einer Schnittstelle mitzuschneiden und zu analysieren. Dies ist in vielen Fällen sehr nüzlich, z. B. um Fehler in Netzverbindungen zu finden, TCP/IP besser zu verstehen oder um den Netzwerkdurchsatz zu messen.

Tcpdump versetzt dazu die entsprechende Netzwerkschnittstelle in den sog. Promiscous-Modus, d. h. die Netzwerkkarte liest nun nicht mehr nur für sie bestimmte Pakete, sondern schneidet alle (gewünschten) mit und gibt sie auf dem Bildschirm aus.

Um auf die Netzwerkschnittstelle zuzugreifen benötigt man Root-Rechte, da so alle übertragenen Daten (auch Passwörter im Klartext!) abgefangen und mißbraucht werden können.

Von Tcpdump mitgeschnittene Daten können von Zusatzprogrammen analysiert und graphisch aufbereitet werden.

11.5.1 *Ausgabe*

Tcpdump ermöglicht es, den kompletten Netzwerkverkehr, der an einer Schnittstelle vorbeifließt, mitzuschneiden. Der Mitschnitt lässt sich nach verschiedenen Gesichtspunkten beeinflussen und analysieren. Startet man Tcpdump ohne Argumente bindet es sich an die erste Netzwerkschnittstelle in der Systemliste, die aktiv ist. Eine bestimmte Schnittstelle lässt sich mittels `-i` spezifizieren. Startet man nun `tcpdump -i tlp1` so wird der gesamte Verkehr über tlp1 mitgeschnitten und am Bildschirm ausgegeben:

Tcpdump zeigt hierbei in Bsp. 11.5.1 den Zeitstempel des Rechners, danach die IP-Adressen bzw. die FQDN[1] inklusive Portnummer an. In Tab. 11.5.1 wird der erste Beispieleintrag genauer beschrieben.

Tcpdump bietet aber noch weitere Optionen um die Ausgabe zu beeinflussen, z. B. `-t` oder `-v` um den Zeitstempel bzw. die Ausgabemenge zu beeinflussen. Als weitere Option beeinflußt `-e` die Darstellung des »Link Level Headers«. In Ethernet-Netzen wird die Zieladresse, das Protokoll und die Paketlänge, in FDDI zusätzlich das Frame-Control-Feld ausgegeben. Token Ring erweitert die Ausgabe noch um das Access-Control-Feld. Für SLIP-Verbindungen wird die Verbindungsrichtung angegeben (I für einkommend, O für ausgehend), der Pakettyp (»ip«, »utcp«, »ctcp«) angezeigt und Kompressionsinformationen ausgegeben.

[1] *Full Qualified Domain Name*, Hostname plus Domain

[!h]

```
tcpdump: listening on tlp0
22:29:37.647072 192.168.2.2.64856
                > frnk.dnscache.mediaways.net.domain:
    12045+ A? ftp.netbsd.org. (32)
22:29:37.856241 frnk.dnscache.mediaways.net.domain
                > 192.168.2.2.64856:
    12045 1/0/0 A ftp.netbsd.org (48)
22:29:37.857544 192.168.2.2.63115
                > ftp.netbsd.org.ftp: S 1798388065:1798388065(0)
    win 65535 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF)
22:29:37.857575 192.168.2.2.63123
                > dhcp-153-96-93-2.ifak.fhg.de.www: F 1510028614:
    1510028614(0) ack 3324910551 win 65535
<nop,nop,timestamp 2055 1126038302> (DF)
22:29:37.993659 dhcp-153-96-93-2.ifak.fhg.de.www > 192.168.2.2.63123:
R 1:1(0) ack 1 win 65535 <nop,nop,timestamp 2055 1126038302> (DF)
^C
31 packets received by filter
```

Listing 11.5: Tcpdump-Beispiel

Feld	Datum
Zeitstempel	22:29:37.647072
Quelle > Ziel	192.168.2.2.64856 > frnk.dnscache.mediaways.net.domain
Adresseintrag	A? (qtype=A)
nachgefragte Adresse	ftp.netbsd.org

Tabelle 11.1.: ein Tcpdump-Beispiel näher betrachtet

11.5.2 *Datagrammtypen*

11.5.2.1 *ARP/RARP*

Angezeigt werden hierbei der Anfragetyp und die Argumente, die Ausgabe ist sehr leicht verständlich:

11.5.2.2 *TCP*

Das allgemeine Format für ein TCP Datum ist:
`Quelle > Ziel: Flags Sequenznr. ACK Window Urgent Optionen`

[!h]

```
arp who-has Boromir tell Arwen
arp reply Arwen is-at 0:3:d3:1a:db:21
```

Listing 11.6: Tcpdump-Beispiel einer ARP-Anfrage

`Flags` sind Kombinationen aus »S« (Syn), »F« (Fin), »P« (Push), »R« (Reset) oder ».« für keine Flags.
Quelle, Ziel und Flags werden immer angezeigt, die darauf folgenden Optionen sind vom Paket abhängig.

Die Sequenznr. ist die Anzahl der Sequenzen die von den Daten im Paket benutzt werden. »ACK« ist die erwartete folgende Sequenznummer in Gegenrichtung, »Window« zeigt die Anzahl der Bytes an, die im Pufferspeicher in Gegenrichtung verfügbar sind. »Urgent« zeigt an das dringende Daten im Paket enthalten sind. »Optionen« sind enthaltene TCP-Optionen.

11.5.2.3 *AFS*

Die Pakete des Andrew File Systems werden folgendermassen ausgegeben:

[!h]

```
src.sport > dst.dport: rx packet-type
src.sport > dst.dport: rx packet-type service call
            call-name args
src.sport > dst.dport: rx packet-type service reply
            call-name args
```

Listing 11.7: Tcpdump-Beispiel von AFS-Verkehr

Die Ausgabe kann erweitert werden, `-v` gibt zusätzlich ACK-Pakete und Headerinformationen wie »RX-ID«, Rufnummer, Sequenznummer, Seriennummer und RC-Paketflags aus. `-vv` erweitert noch zusätzlich die Ausgabe um die MTU Informationen, ein weiteres `v` gibt dazu noch die Service-ID und den Security-Index aus.

Da AFS-Calls sehr groß sind, muss die »snaplen« mit `-s 256` heraufgesetzt werden.

[!h]

```
# tcpdump -i ex0 'icmp'
tcpdump: listening on ex0
00:01:57.047524 Aragorn > 192.168.2.1: icmp: echo request seq 0
00:01:57.047654 192.168.2.1 > Aragorn: icmp: echo reply seq 0
```

Listing 11.8: Tcpdump-Beispiel von ICMP-Paketen

11.5.2.4 *ICMP*

Es wird wie üblich Zeitstempel und Quelle/Ziel angezeigt, dazu das IP-Protokoll (»icmp«) und der ICMP-Typ (8/0). Zu ICMP-Pakettypen gibt es eine Übersicht im Anhang A

11.5.3 *Filter*

Neben den vielen Möglichkeiten die Ausgabe zu beeinflussen hat Tcpdump die weitaus interessantere Fähigkeit verschiedene Pakete zu filtern. So ist es möglich mit der Option `-T` einen Pakettyp (bspw. »rtp«, »rpc« oder »snmp«) zu spezifizieren. Zusätzlich dazu kann mittels `-F` eine Datei angegeben werden, aus der Tcpdump Filterregeln ausliest, diese Regeln können aber auch einfach mit den entsprechenden Schlüsselwörtern als Kommandoparameter übergeben werden. Das »$«-Zeichen soll hier als Platzhalter für Netz- / IP-Adressen oder Hostnamen dienen, letztere müssen allerdings von der betreffenden Maschine aufgelöst werden können.

Schlüsselwort	Funktion
`type`	mit `host`, `net` und `port` können die entsprechenden Parameter angegeben werden
`dir`	gibt die Übertragungsrichtung an
`proto`	übergibt das Protokoll: `ether, fddi, tr, ip, ip6, arp, rarp, decnet, tcp, udp`
`and, or, not`	erlaubt es Filterregeln zu verknüpfen
`dst host $`	filtert das `destination field` der Pakete
`src host $`	filtert das `source field` der Pakete
`host $`	filtert das `source` und `destination field` der Pakete
`ether dst $`	filtert die Ethernet destination address
`ether src $`	filtert die Ethernet source address
`ether host $`	filtert die Ethernet source oder destination address
`gateway $`	prüft ob das Paket das spezifizierte Gateway passiert hat, d. h. die Adresse darf weder Source noch Destination sein

`dst net $`	filtert nach dem Zielnetz (siehe networks(4))
`src net &`	filtert nach dem Quellnetz (siehe networks(4))
`net &`	filtert nach Quell- oder Zielnetz
`net $ mask $`	filtert nach Netz und Netzmaske (Nur in IPv4)
`net $/$`	filtert nach Netz und passender Subnetz Bitlänge in CIDR Notation
dst port $	filtert in den Protokollen ip/tcp, ip/udp, ip6/tcp oder ip6/udp nach dem Zielport
src port $	filtert nach dem Quellport
port $	filtert nach Quell- oder Zielport
`less $`	filtert Pakete die kleiner als $ sind
`greater $`	filtert Pakete die größer als $ sind
`ip proto $`	filtert nach IP-Protokollen: `icmp, icmp6, igmp, igrp, pim, ah, esp, vrrp, udp, tcp`, da `icmp, udp, tcp` Schlüsselwörter sind, müssen sie mit \ escaped werden
`ip6 proto $`	filtert nach dem entsprechenden IPv6 Protokoll
`ip6 protochain $`	filtert Pakete die IPv6 sind und einen passenden `protocol` Header haben
`ip protochain &`	filtert Pakete die IPv4 sind und einen passenden `protocol` Header haben
`ether broadcast`	filtert Ethernet Boradcast Pakete
`ip broadcast`	filtert IP Broadcast Pakete mit Nullen oder Einsen Header und prüft die lokale Subnetzadresse
`ether multicast`	filtert Ethernet Multicast Pakete
`ip multicast`	filtert IPv4 Multicast Pakete
`ip6 multicast`	filtert IPv6 Multicast Pakete
`ether proto $`	filtert Ethernet Pakete in folgenden Protokollen: `ip, ip6, arp, rarp, atalk, aarp, dec- net, sca, lat, mopdl, moprc, iso, stp, ipx, netbeui`, diese müssen wieder per \ escaped werden
`decnet src $`	filtert in DECNET nach dem angegebenen Quellhost
`decnet dst $`	filtert in DECNET nach dem angegebenen Zielhost
`decnet host $`	filtert in DECNET nach dem angegebenen Host
`ip, ip6, arp, rarp, atalk, aarp, decnet, iso, stp, ipx, netbeui`	sind Kurzformen für `ether proto $`
`vlan`	filtert IEEE 802.1Q VLAN Pakete
`vlan $`	filtert VLAN Pakete mit entsprechender ID
`tcp, udp, icmp`	Kurzform von `ip proto $ or ip6 proto $`
`iso proto $`	filter OSI Pakete des Typs `clnp, esis, isis`
`clnp, esis, isis`	Kurzform für `iso proto $`

Tcpdump ist weiterhin noch in der Lage den Inhalt der Pakete mittels logischer Ausdrücke zu filtern, dies geschieht mit einer einfachen Anweisung der Form:

Ausdruck Relation Ausdruck

mit Relation aus {>, <, >=, <=, =, ! =} und Ausdruck ist ein arithmetischer Ausdruck bestehend aus Integerzahlen, Binäroperatoren aus {+, −, ∗, /, &, |} und spezielle Ausdrücke um Paketdaten zu filtern. Daten innerhalb der Pakete werden mit *$Protokoll [$Ausdruck : $Grösse]* gefiltert, dabei ist $Protokoll aus {»ether«, »fddi«, »tr«, »ppp«, »slip«, »link«, »ip«, »arp«, »rarp«, »tcp«, »udp«, »icmp«, »ip6«,} und spezifiziert das gewünschte Protokoll. *$Ausdruck* übergibt das Byte-Offset relativ zur Protokollschicht. *$Grösse* übergibt optional die Anzahl (1,2 oder 4) der Bytes ab dem übergebenen Byte-Offset, zusätzlich kann noch mit *len $* die Paketlänge angegeben werden. Tcpdump ist auch in der Lage einige Feldnamen bzw. Offsets, TCP-Flags und ICMP-Typen über Namen anzusprechen, diese sind in Tab. 11.5.3

`Feldnamen`

- `icmptype`
- `icmpcode`
- `tcpflags`

`TCP Flags`

- `tcp-fin`
- `tcp-syn`
- `tcp-rst`
- `tcp-push`
- `tcp-push`
- `tcp-ack`
- `tcp-urg`

`ICMP Typen`

- `icmp-echoreply`
- `icmp-unreach`
- `icmp-sourcequench`
- `icmp-redirect`
- `icmp-echo`
- `icmp-routeradvert`
- `icmp-routersolicit`
- `icmp-timxceed`
- `icmp-paramprob`
- `icmp-tstamp`
- `icmp-tstampreply`
- `icmp-ireq,`
- `icmp-ireqreply`
- `icmp-maskreq`
- `icmp-maskreply`

Tabelle 11.4.: Pakettypen für Tcpdump spezifizieren

Geklammerte Ausdrücke können wieder mit logischen Operatoren (`!= = NOT, && = AND und || = OR`) verknüpft werden.

[!h]

```
## zeigt allen Verkehr auf einem Gateway an,
## der nicht in das eigene Subnetz geht
# tcpdump ip and not net localnet

## zeigt FTP Traffic der über 'Mimung' gekommen ist
# tcpdump 'gateway Mimung and (port ftp or ftp-data)'

## zeigt ICMP Pakete an die nicht Ping und Pong sind
# tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'

## zeigt allen Verkehr der 'Ratatoskr' aber nicht 'Fenris' betrifft
# tcpdump ip host Ratatoskr and not Fenris
```

Listing 11.9: Tcpdump-Filterbeispiele

11.5.4 *Tcpdump-Daten mit Tcptrace aufbereiten*

Tcptrace analysiert Tcpdump-Dateien und erzeugt dafür Zusammenfassungen bzw. xplot-Graphen.

11.5.4.1 *Ausgabe*

`-b` bzw. `-l`	kurze bzw. lange Zusammenfassung
`-r`	Rundlaufzeit-Statistik (*RTT*)
`-D`	dezimale Ausgabe
`-X`	hexadezimale Ausgabe
`-n`	Namen nicht auflösen
`-s`	Hostnamen statt FQDN
`-G`	erzeugt alle Graphen
`-T`	Durchsatzdiagramme
`-R`	Rundlaufzeit-Graphen (*RTT*)
`-S`	Zeitsequenzgraphen
`-N`	Congestion-Window-Graphen
`-F`	Segmentgröße
`-C` bzw. `-M`	Farbe / Monochromgraphik

Tabelle 11.6.: Tcptrace-Optionen

11.5.4.2 *Filter*

Für gewöhnlich snifft Tcpdump mehrere Verbindungen mit, die auch von Tcptrace erkannt werden. Diese Verbindungen werden mit »a2b_...« bzw. »b2a_...« usw. bezeichnet. Tcptrace ist selbstverständlich auch wieder in der Lage dies Verbindungen zu filtern. Mittels `-i $` wird die Verbindung $ ausgeblendet, mit `-o $` wird nur die Verbindung $ betrachtet, `-c` blendet Verbindungen aus die nicht abgeschlossen wurden (keine SYN und FIN Pakete vorhanden). Es lassen sich auch Segmentnummern übergeben die jeweils obere (`-E $`) bzw. untere (`-B $`)Grenze darstellen. `-u` gibt zusätzlich noch minimale UDP-Informationen aus, `-e` extrahiert den Inhalt jedes TCP-Streams in eine Datei, `-p` schreibt den Inhalt aller Pakete in eine Datei.

11.5.4.3 *Plotter und andere Freunde*

Geradezu unverzichtbar im Umgang mit Tcpdump und Tcptrace ist `xplot` [PP], das die erzeugten Diagramme unter X11 graphisch darstellt. Desweiteren gibt es eine Reimplementierung von `xplot` für Java namens `jplot` [PP]. Tracelook [PP] erzeugt ebenfalls xgraph-Plots. Da das xplot-Format etwas schwierig zu handhaben ist gibt es von der Uni Ohio ein AWK-Skript [PP] das xplot nach gnuplot konvertiert, und so z. B. für LaTeX aufbereitbar macht. Abb. 11.1 zeigt einen mit Gnuplot geplotteten Zeitsequenzgraphen

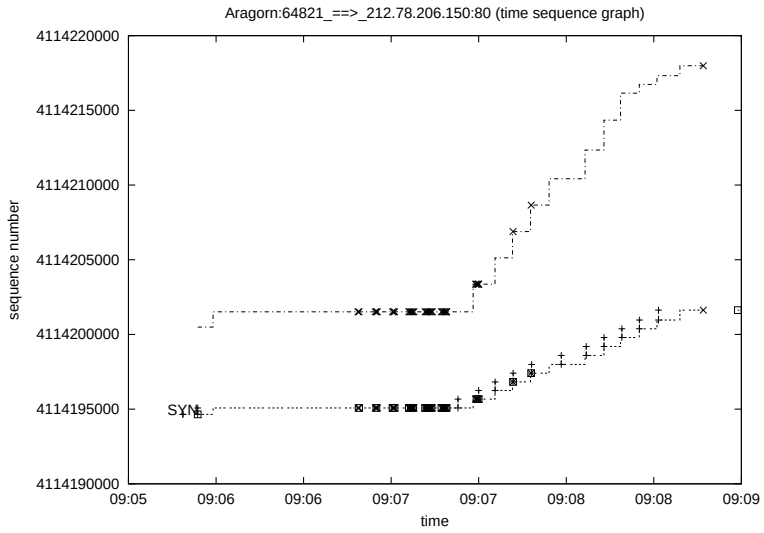


Abbildung 11.1.: mit Gnuplot geplotteter Tcpdump-Zeitsequenzgraphen

Neben Tcptrace gibt es von Sony Japan noch ein Tool namens tcpillust [PP],das aus Tcptrace-Dateien Graphiken erzeugt, die denen aus W. R. Stevens »TCP/IP Illustrated« Reihe gleichen. Tcpdstat erzeugt ebenfalls Statistiken aus Tcptrace Dateien, unter anderem mit Zeit, Größe und Paketen je Protokoll. Flstats erzeugt ebenfalls Flussstatistiken.

Da Tcptrace in der Regel große Dateien erzeugt können diese mit `tcpslice` [PP] in kleinere zerlegt werden. `tcpdmerge` arbeitet genau entgegengesetzt, es führt 2 dumps wieder zusammen. `tcpslice` extrahiert Teile eines Mitschnitts,

Äußerst bedenklich ist allerdings die extrem hohe Verletzung der Privatsphäre durch

mitgesniffte Daten, daher hat Ethan Blanton mit `tcpurify` [PP] einen Sniffer entwickelt der auch Wert auf die Privatsphäre legt. Ebenso gibt es noch `tcpdpriv` [PP], das dazu dient Tcptrace Dateien zu anonymisieren.

11.6 EMAILS ÜBER SSL GETUNNELT MIT FETCHMAIL ABHOLEN

Fetchmail [PP11] ist ein beliebtes Programm um Emails abzuholen. Es unterstützt neben POP3 und IMAP auch POP3 und IMAP über SSL (pops, imaps). Wenn der Mailserver die SSL-Varianten unterstützt, muss man sie unbedingt verwenden, da so Abhöraktionen des Netzwerkverkehr erschwert werden und die Passwörter nicht mehr im Klartext durchs Netz gehen.

Konfiguriert wird Fetchmail entweder per Kommandozeilen-Optionen oder in der Konfigurationsdatei (`~/.fetchmailrc`), für die das Konfigurationswerkzeug [PP12] existiert. In der folgenden Liste stehen alle SSL-relevanten Optionen.

`-SSL` / `SSL`
: aktiviert SSL, der Server muss es unterstützen

`-SSLCERT <NAME>` / `SSLCERT`
: Name des SSL-Zertifikats des Clients, einige Server benötigen diese Angabe

`-SSLKEY <NAME>` / `SSLKEY`
: Name des privaten SSL-Schlüssel des Clients, einige Server benötigen diese Angabe

`-SSLPROT <NAME>` / `SSLPROTO`
: Setzt das gewünschte SSL-Protokoll, möglich sind:`ssl2`, `ssl3` and `tls1`

`-SSLCERTCK` / `SSLCERTCK`
: verlangt strikte Prüfung des SSL-Zertifikats, das Server-Zertifikat muß mit einem gültigen lokalen Zertifikat signiert sein

`-SSLCERTPATH <DIRECTORY>` / `SSLCERTPATH`
: Verzeichnis des lokalen SSL-Zertifikats (→ `-sslcertck`)

`-SSLFINGERPRINT` / `SSLFINGERPRINT`
: SSL-Fingerprint des Servers, wie von OpenSSL definiert

Ein Beispiel einer `~/.fetchmailrc` gibt es in 11.10:

[!h]

```
poll pop3.server.local with proto POP3 user 'MailNutzer'
     there with password 'XXX' is 'lokalerNutzer' here
 ssl sslproto ssl3 fetchall
```

Listing 11.10: Beispiel einer .fetchmailrc

SIEHE AUCH:
http://www.catb.org/~esr/fetchmail/

12 ANGEWANDTE KRYPTOGRAPHIE

Beware of bugs in the above code; I have only proved it correct, not tried it.

(Donald E. Knuth)

12.1 DAS KRYPTOGRAPHISCHE PSEUDOGERÄT CGD

12.1.1 *Was ist CGD?*

CGD ist der »cryptographic disk driver«, der ab NetBSD 2.0 verschlüsselte Partitionen auf Blockebene ermöglicht. Hierbei wird der Zugriff auf die echte Partition über einen Pseudogerätetreiber abgewickelt, der die Daten entschlüsselt an das lesende Programm schickt und verschlüsselt auf die Festplatte schreibt.

CGD verwendet dazu symmetrische Kryptoverfahren mit einem einfachen Passwort, das zum Einbinden der echten Partition in den Pseudogerätetreiber übergeben werden muss. Zusätzlich kann CGD auch beim Starten einen Zufallsschlüssel für eine Partition generieren, der nach dem Herunterfahren verloren geht. Dieses Verfahren bietet sich für die Swappartition an, da deren Daten nach dem Herunterfahren des System nicht mehr von Bedeutung sind.

12.1.2 *Installation und Einrichtung*

CGD wird als Pseudogerät im Kernel konfiguriert, dazu muss die Kernelkonfiguration wie in Listing 12.1 angepasst werden. Benötigt man mehr als vier verschlüsselte Partitionen, muss man die 4 als letzte Option erhöhen.

[!h]

```
pseudo-device  cgd  4  # cryptographic disk driver
```

Listing 12.1: CGD im Kernel aktivieren

Nachdem der Kernel für CGD konfiguriert wurde, muss man die passenden Pseudogeräte (/dev/[r]cgd[0-3][a-p]) in /dev/ erstellen, dies geschieht mit `cd /dev/ && MAKEDEV cgd`.

CGD unterstützt zur Zeit drei verschiedene Algorithmen, TripleDES, AES und Blowfish, allesamt im CBC-Modus. TripleDES ist definitiv nicht empfehlenswert, da es unsicher und extrem langsam ist, Blowfish und AES sind beide recht sicher[1] und im Allgemeinen recht schnell, wobei Blowfish schneller als AES ist. Ich persönlich setze AES für mein Homeverzeichnis und Blowfish für /tmp und swap ein.

12.1.3 *Eine Datenpartition verschlüsseln*

Da CGD auf Partitionsebene arbeitet, benötigt man zuerst eine Partition, die verschlüsselt werden soll. Meist ist dies die Homepartition. Wird die Partition bereits genutzt, müssen die Daten gesichert werden, siehe dazu Kap. 14 und anschließend sicher gelöscht werden, siehe dazu Kap. 9.2.0.3.

Ist die Partition nun leer und kann für CGD verwendet werden, erstellt man zuerst die CGD-Konfiguration mit cgdconfig(8) wie in Listing 12.2 beschrieben. Dazu kann man verschiedene Optionen übergeben, unter anderem die Verifikationsmethode, den Dateinamen für die Konfig und den gewünschten Algorithmus sowie die Schlüssellänge. Mit der Verifikationsmethode wird überprüft, ob die CGD-Partitition korrekt eingebunden wurde, d.h. hier wird geprüft, ob ein FFS-Dateisystem auf der Partition existiert, nachdem das Passwort angegeben wurde. Nachdem die Konfiguration erstellt wurde, wird mit dem zweiten Befehl das Passwort festgelegt. Anschließend wird ein Dateisystem angelegt und die Pseudopartition nach /home eingemountet, wo sie ganz normal verwendet werden kann. Solle die Partition deaktiviert werden, muss man sie unmounten und mit `cgdconfig -u /dev/cgd0a` deaktivieren. Man kann nun nicht mehr auf die Daten zugreifen.

Um die Homepartition bei jedem Startvorgang automatisch einzubinden, muss man in der /etc/rc.conf `cgd=yes` setzen und `/dev/cgd0a /home ffs rw 1 1` in die /etc/fstab eintragen.

[1] Blowfish war AES-Finalist

[!h]

```
# cgdconfig -g -V ffs  -o /etc/cgd/wd0h aes-cbc 256
# cgdconfig -V re-enter cgd0 /dev/wd0e
# newfs /dev/cgd0a
# mount /dev/cgd0a /home
```

Listing 12.2: CGD-Konfiguration erstellen

Man wird nun automatisch nach der Passphrase gefragt und die Partition wird danach ggf. mit fsck überprüft und eingebunden.

12.1.4 *CGD für Swap und /tmp*

Die Swappartition wird verwendet, um Daten aus dem Arbeitsspeicher auszulagern, bspw. wenn dieser zu voll ist. Die kann die Sicherheit des Systems gefährden, wenn es sich dabei um eigentlich verschlüsselte Daten handelt, die unverschlüsselt im RAM liegen und ausgelagert werden.

Damit dies nicht passiert, kann man auch die Swappartition mit CGD verschlüsseln. Im Prinzip geht man hier genauso vor wie bei einer normalen Datenpartition, lediglich die Schlüsselgenerierung wird abgeändert. CGD unterstützt »urandomkey« als Schlüsselmethode. Hierbei werden einfach aus /dev/urandom, einem Pseudozufallsgenerator, Zeichen ausgelesen und als Schlüssel verwendet. Der Schlüssel wird dabei nicht gespeichert, so das beim Absturz des Systems oder nach dem Herunterfahren die Partition nicht mehr entschlüsselt werden kann. Dies ist aber bei einer Swappartition nicht notwendig, da sie bei jedem Systemstart neu initialisiert wird.

In Listing 12.3 wird eine solche Konfiguration erstellt. Hierzu wird die Swappartition zuerst deaktiviert und anschließend eine Konfiguration mit der entsprechenden Schlüsselmethode erzeugt.

Da bei jedem Start des Systems die Swappartition neu angelegt wird, muss auch das Disklabel zurückgespielt werden, da es standardmäßig auf »4.2BSD« statt »swap« gesetzt wird. Dies geschieht am einfachsten indem man die Swappartition einmal händisch konfiguriert, das Label sichert und später automatisch bei jedem Start einbindet.

Listing 12.4 demonstriert das Vorgehen. Zuerst wird das CGD-Gerät konfiguriert (1), das Disklabel erzeugt (2) und in eine Datei gesichert (3). Die Befehle (4) und (5) spielen das Disklabel zurück und konfigurieren die Swappartition wieder als Swap.

[!h]

```
# swapctl -d /dev/wd0b
# cgdconfig -g -k urandomkey -o /etc/cgd/wd0b \
                                blowfish-cbc 192
# cat /etc/cgd/wd0b
algorithm blowfish-cbc;
iv-method encblkno;
keylength 192;
verify_method none;
keygen randomkey;
```

Listing 12.3: CGD für die Swappartition konfigurieren

[!h]

```
# cgdconfig cgd1 /dev/wd0b
# disklabel -i /dev/cgd1a
 partition> a
 Filesystem type [?] [swap]: swap
 Start offset ('x' to start after partition 'x')
                                    [0c, 0s, 0M]:
 Partition size ('$' for all remaining)
                  [755.508c, 1547280s, 755.508M]:
 partition> W
 Label disk [n]? y
 Label written
 partition> Q
3# disklabel -r /dev/cgd1a > /etc/cgd/swap.disklabel
4# disklabel -R -r cgd1 /etc/cgd/swap.disklabel
5# swapctl -a /dev/cgd1a
6# swapctl -l
Device     512-blocks Used  Avail Capacity Priority
/dev/cgd1a   1547280    0  1547280     0%  0
```

Listing 12.4: Disklabel für die Swappartition sichern

Automatisieren kann man diesen Vorgang für jeden Systemstart in /etc/rc.local wie in Listing 12.5 gezeigt.

[!h]

```
/sbin/cgdconfig cgd1 /dev/wd0b
/sbin/disklabel -R -r cgd1 /etc/cgd/swap.disklabel
/sbin/swapctl -a /dev/cgd1a
```

Listing 12.5: Verschlüsselte Swappartition beim Start einbinden

12.1.5 *CGD für die /tmp-Partition*

Die /tmp-Partition sollte genauso wie die Swappartition geschützt werden, dazu muss sie auch genauso konfiguriert werden. Es muss hierbei lediglich das Dateisystem nach der Konfiguration neu angelegt werden, da die /tmp-Partition ja als normale Datenpartition eingebunden wird.

Listing 12.6 zeigt die Konfiguration an einem Beispiel.

[!h]

```
# cgdconfig -g -k randomkey -o /etc/cgd/wd0h \
                             blowfish-cbc 192
# cat /etc/cgd/wd0h
algorithm blowfish-cbc;
iv-method encblkno;
keylength 192;
verify_method none;
keygen randomkey;
#
```

Listing 12.6: CGD für /tmp konfigurieren

Um die verschlüsselte Partition beim Systemstart einzubinden, kann man die Zeilen aus Listing 12.7 in /etc/rc.local einfügen.

Zuerst wird die Partition für CGD konfiguriert und mit newfs ein neues Dateisystem angelegt. Da die /tmp-Partition in der Regel recht klein ist, dauert dies auch nicht lange. Anschließend wird sie eingemountet und mit chmod die Zugriffsrechte angepasst.

[!h]

```
/sbin/cgdconfig cgd2 /dev/wd0h && /sbin/newfs /dev/cgd2a
/sbin/mount /dev/cgd2a /tmp
/bin/chmod 777 /tmp/
```

Listing 12.7: verschlüsselte /tmp-Partition einbinden

12.1.6 *CGD für Container und/oder CDs/DVDs*

NetBSD kann mittels vnd(4) virtuelle Datenträger bereitstellen. Dies heißt im Klartext, das man eine Datei erzeugt, in dieser Datei ein Dateisystem anlegt und dieses über ein Loopbackgerät (/dev/vndo) wie eine normale Platte einbindet.

Da sich CGD nicht dafür interessiert, ob es auf einer »echten« Platte oder einem VND-Gerät angelegt wird, kann man so verschlüsselte Container erzeugen.

Dazu befolgt man Listing 12.8, welches zuerst eine 695MB große Datei mit dd(1) erzeugt, diese mit vnconfig(8) als virtuelle Platte einbindet und anschließend mit cgdconfig(8) verschlüsselt.

Den entstandenen Container kann man auf Wechseldatenträger kopieren, per NFS importieren oder auf eine CD brennen.

[!h]

```
$ dd if=/dev/zero of=image bs=1m count=695
# vnconfig -c vnd0 image
# cgdconfig -g -o image.cgd blowfish-cbc 128
# cgdconfig -V re-enter cgd3 /dev/vnd0a image.cgd
# newfs /dev/cgd3a
# mount /dev/cgd3a /mnt/
   ...
# umount /dev/cgd3a
# cgdconfig -u cgd3
# vnconfig -u vnd0
```

Listing 12.8: CGD-Container

SIEHE AUCH:
cgd(4), cgdconfig(8), vnd(4), vnconfig(8)
(2006, Chapter 14. The Cryptographic Device Driver (CGD))
(2006, Part V. Building the System)
(2003)
(2003)

12.2.1 *Was ist CFS?*

CFS ist das »Cryptographic Filesystem« von Matt Blaze, welches als erstes Dateisystem Verschlüsselung implementierte und im Prinzip wie NFS arbeitet. Dies bedeutet, das komplette Verzeichnisse und deren Inhalt verschlüsselt im Dateisystem abgelegt werden. Beim Zugriff werden sie über den cfsd(8), der auf 127.0.0.1 lauscht, unverschlüsselt an das anfragende Programm übergeben. Soll die Datei zurückgeschrieben werden geschieht dies wieder über den cfsd(8), der die Datei entsprechend verschlüsselt in das Dateisystem schreibt. Für den Benutzer ist CFS nach der Eingabe der Passphrase transparent, d.h. er bemerkt nicht daß CFS überhaupt installiert ist. CFS verschlüsselt neben dem Inhalt der Dateien auch die Namen, so daß keinerlei Rückschlüsse auf den Inhalt möglich sind. Listing 12.9 zeigt beispielhaft ein verschlüsseltes und entschlüsseltes CFS-Verzeichnis.

Da CFS auf Dateiebene verschlüsselt, lassen sich die Dateien normal auf Wechseldatenträger kopieren und so zwischen verschiedenen Systemen austauschen. Ebenso ist es möglich, die verschlüsselten Verzeichnisse per NFS von einem Server zu importieren und lokal zu entschlüsseln.

Da NetBSDs mount_nfs(8) keinen anderen Port als 2049 ansprechen kann, kann man den CFS cfsd(8) nicht gleichzeitig mit einem NFS-Dæmon betreiben.

[!h]

```
$> ls -la loop/entschluesselttest/
total 18104
drwxr-xr-x 2 stefan stefan     512 Apr 21 21:52 ./
drwx------ 6 stefan stefan    1024 Apr 21 21:51 ../
-rw-r--r-- 1 stefan stefan  311296 Apr 21 21:52 brief.tex
-rw-r--r-- 1 stefan stefan    8094 Apr 21 21:51 dmesg
-rwxr-xr-x 1 stefan stefan 8898275 Apr 21 21:51 netbsd*
$> ls -la crypto/cd98d68fd9f031bc/
total 18104
drwxr-xr-x 2 stefan stefan     512 Apr 21 21:52 ./
drwxr-xr-x 6 stefan stefan    1024 Apr 21 21:51 ../
lrwxrwxrwx 1 stefan stefan       8 Apr 21 21:51 .pvect_11f8d60dcf7ef480@ -> fe4fdf15
lrwxrwxrwx 1 stefan stefan       8 Apr 21 21:51 .pvect_2ff8f5c49324f41a@ -> 69837ab5
lrwxrwxrwx 1 stefan stefan       8 Apr 21 21:52 .pvect_de90918d9ef9382867086595913ba1c6@ \
                                                -> 475d934f
-rwxr-xr-x 1 stefan stefan 8898283 Apr 21 21:51 11f8d60dcf7ef480*
-rw-r--r-- 1 stefan stefan    8102 Apr 21 21:51 2ff8f5c49324f41a
-rw-r--r-- 1 stefan stefan  311296 Apr 21 21:52 de90918d9ef9382867086595913ba1c6
```

Listing 12.9: Verschlüsseltes und entschlüsseltes CFS-Verzeichnis

12.2.2 *Installation und Konfiguration*

CFS wird aus pkgsrc/security/cfs installiert und kann anschließend konfiguriert werden. Dazu muss ein Loopbackverzeichnis erstellt werden, durch das die Daten durchgeschleift werden, außerdem das eigentliche verschlüsselte Verzeichnis und ein Mountpunkt, an dem die entschlüsselnden Verzeichnisse eingemountet werden. In Listing 12.10 wird /home/stefan/crypt als Daten- und /null/ als Loopbackverzeichnis angelegt und zum NFS-Export vorbereitet, mountd(8) und cfsd(8) werden (ggf. neu) gestartet. Anschließend muss bei jedem Start mountd(8) via /etc/rc.conf gestartet werden. Um cfsd(8) zu starten und /null zu mounten, können die letzten beiden Zeilen in /etc/rc.local eingefügt werden.

[!h]

```
$ mkdir /home/stefan/crypt
# mkdir /null
# chmod 0 /null
# echo /null localhost » /etc/exports
# /etc/rc.d/mountd [re]start && /usr/pkg/sbin/cfsd
# echo "mount -o intr,-2,-w=4096,-r=4096 127.0.0.1:/null \
        /home/stefan/crypt" » /etc/rc.local
# echo ‘which cfsd‘ » /etc/rc.local
```

Listing 12.10: CFS-Verzeichnisse einrichten und Dienste starten

Nachdem die ersten beiden Verzeichnisse eingerichtet und die Dienste gestartet sind, kann man das verschlüsselte Verzeichnis einrichten. Dazu existiert mit cfs_mkdir(1) ein eigener Befehl. Mit diesem Befehl legt man auch den Verschlüsselungsmodus fest.

CFS unterstützt verschiedene Algorithmen, siehe Listing 12.11. DES ist sehr langsam und definitiv nicht mehr sicher, MacGuffin und SAFER-SK128 sind nicht sehr verbreitet, Blowfish hingegen ist sehr weit verbreitet, sehr schnell und wird bisher als recht sicher betrachtet.

Listing 12.12 generiert ein mit Blowfish verschlüsseltes Verzeichnis. Dabei verlangt es eine mindestens 16-stellige Passphrase, die entsprechend sicher gewählt werden muss. Um das Verzeichnis nutzen zu können, muss es mit cfs_attach(1) eingebunden werden. Man übergibt als Option zuerst den Pfad des verschlüsselten Verzeichnisses und als zweite Option einen frei gewählten Namen, unter dem das Verzeichnis im Loopbackverzeichnis eingemountet wird. Im Beispiel ist /home/stefan/encrypted dann als /home/stefan/crypt/usable_cfsdir ansprechbar und kann entsprechend genutzt werden. Alle Dateien die nach /home/stefan/crypt/usable_cfsdir kopiert werden, werden verschlüsselt unter /home/stefan/encrypted abgelegt. Soll das entschlüsselnde Verzeichnis wieder entfernt werden, gibt man cfs_detach und den Verzeichnisnamen ein.

[!h]

```
1 einfaches DES im Hybridmodus mit zwei Schlüsseln
3 3DES mit drei Schlüsseln
B Blowfish mit 128-Bit-Schlüssel
M MacGuffin
S SAFER-SK128
```

Listing 12.11: verfügbare CFS-Algorithmen

[!h]

```
$ cfs_mkdir -b /home/stefan/encrypted
$ cfs_attach /home/stefan/encrypted usable_cfsdir
$ cfs_detach usable_cfsdir
```

Listing 12.12: CFS Verzeichnis anlegen, einbinden und deaktivieren

12.2.3 *CFS auf Wechselmedien*

CFS lässt sich hervorragend auf Wechselmedien einsetzen, solange diese ein Unix-Dateisystem unterstützen. USB-Sticks oder ZIP-Disketten müssen also mit FFS, LFS, ext2, ext3 oder Ähnlichem formatiert werden, anschließend kann man ein verschlüsseltes CFS-Verzeichnis komplett auf das Medium kopieren oder auf dem Medium erstellen. Das Medium kann dann auf verschiedenen Rechnern verwendet werden, solange diese (a) das Dateisystem lesen können und (b) CFS installiert haben.

Ich habe die Portabilität getestet indem ich einen USB-Stick mit ext2 formatiert und ein CFS-Verzeichnis angelegt habe. Dieses ließ sich von NetBSD, FreeBSD und Gentoo Linux anstandslos einbinden. Andere Unixe sollten damit auch keine Probleme haben.

Neben wiederbeschreibbaren Wechselmedien kann man auch CDs bzw. DVDs mit CFS schützen. Dazu muss man lediglich darauf achten das die Dateien alle auf die CD passen und die verschlüsselten Verzeichnisse mit mkisofs(8) in ein ISO-Image überführen, das gebrannt werden kann.

Ist auf dem Zielrechner der cfsd(8) gestartet und entsprechend konfiguriert, kann man die CD mounten und das verschlüsselte Verzeichnis normal mit cfs_attach(1) einbinden.

12.2.4 *CFS und Datensicherung*

Da CFS auch die Pfadnamen verschlüsselt, ist eine händische Datensicherung recht problematisch. Entweder man sichert immer das komplette verschlüsselte Verzeichnis oder verwendet dump(8) mit Dumplevels oder andere Backupprogramme mit Levels/Datenoptionen.

Man kann auch die Daten entschlüsseln und entschlüsselt sichern, dann ist es aber zwingend erforderlich, das Sicherungsarchiv zu schützen, sprich zu verschlüsseln.

12.2.5 *CGD oder CFS?*

Welches System ist zu bevorzugen? CGD oder CFS?

Diese Frage lässt sich nicht einfach mit »Entweder - Oder« beantworten, da beide Systeme andere Anforderungen und Ziele haben.

CGD ist ein Jahrzent jünger, wesentlich transparenter und im Allgemeinen auch schneller als CFS. Es arbeitet aber auf Blockebene, CFS hingegen auf Dateiebene. Und das ist auch ein wichtiger Unterschied. Mit CGD verschlüsselte Dateien lassen sich nur entschlüsselt sichern - CFS-verschlüsselte Dateien hingegen kann man auch verschlüsselt sichern.

CGD ist ursprpünglich dazu gedacht, Laptops etc. bei Verlust zu schützen, daher schützt es nur eine Partition, CFS hingegen kann beliebig viele *verschiedene* verschlüsselte Verzeichnisse in einer Partition anlegen und ist somit wesentlich flexibler und erlaubt eine bessere Granularität als CGD. Möchte man an einem Mehrbenutzerrechner jedem Benutzer zugestehen, sein Homeverzeichnis mit CGD zu schützen, muss zwangsläufig jeder Benutzer auch eine eigene Partition bekommen.

CGD ist also auf einem Mehrbenutzerrechner nicht wirklich einsetzbar. Hat man hingegen ein Einbenutzerrechner oder einen Rechner für einige wenige Benutzer, kann man CGD problemlos einsetzen.

Viel wichtiger ist aber, daß man CGD und CFS problemlos kombinieren kann - in einer Partition. Auf meinem Laptop und meinen Arbeitsrechnern sind die Homepartitionen (auch /temp und Swap) jeweils mit CGD verschlüsselt. Zusätzlich sind auch noch ~/mail, ~/.gnupg und einige andere Verzeichnisse mit sensiblen Daten mit CFS verschlüsselt. Dies ist bspw. recht praktisch, wenn ich einen Vortrag halte und mein Rechner in einem unsicheren Netz ist. Ich kann meine Homepartition mit CGD problemlos einmounten, die CFS-geschützten Dateien sind aber immer noch

verschlüsselt und somit geschützt.

12.3 symmetrische verschlüsselung mit mcrypt(1)

Mcrypt ist ein Programm zur symmetrischen Verschlüsselung von Dateien. Es übernimmt verschiedene Optionen, unter anderem:

-g generiert OpenPGP-Paket

-d entschlüsseln

-s *size* Schlüsselgröße

-z (de)komprimieren mit GZIP

-p (de)komprimieren mit BZIP2

-b »baremode«, Metadaten wie Modus oder Algorithmus werden nicht gespeichert, verhindert eine Identifikation des Verschlüsselungsmodus mit file(1)

–flush leitet die Daten sofort weiter, nützlich in Pipes

–echo zeigt Sternchen (»*«) bei der Passworteingabe an

–list zeigt eingebaute Algorithmen und Modi

-m *modus* setze Modus

-a *algo* setze Algorithmus

–list-keymodes zeigt eingebaute Schlüsselmodi

-o *keymode* setze Schlüsselmodus

–list-hash zeigt eingebaute Hash-Algorithmen

-h *hash* setze Hash

-v höhere Auskunftsfreudigkeit

[!h]

```
$ ls
brief.txt
$ mcrypt -h ripemd320 -o s2k-salted-ripemd -m cbc \
                 -a rijndael-256 --echo brief.txt
Enter the passphrase (maximum of 512 characters)
Please use a combination of upper and lower case letters and numbers.
Enter passphrase: ********************
Enter passphrase: ********************

File brief.txt was encrypted.
$ ls -l
total 32
-rw-r--r--  1 stefan  wheel  7641 Aug 25 22:26 brief.txt
-rw-------  1 stefan  wheel  7816 Aug 25 22:26 brief.txt.nc
$ file brief.txt.nc
brief.txt.nc: mcrypt 2.5 encrypted data, algorithm: rijndael-256, \
                                    keysize: 32 bytes, mode: cbc,
$ rm -P brief.txt
$ mdecrypt --echo brief.txt.nc
Enter passphrase: **********************
File brief.txt.nc was decrypted.
$ ls -l
total 32
-rw-------  1 stefan  wheel  7641 Aug 25 22:26 brief.txt
-rw-------  1 stefan  wheel  7816 Aug 25 22:26 brief.txt.nc
```

Listing 12.13: Eine Datei mit mcrypt(1) ver- und entschlüsseln

NetBSD kommt im Basissystem mit OpenSSL, einer kryptographischen Werkzeugsammlung. Ursprünglich und Hauptsächlich ist es zur Verwaltung von SSL-Zertifikaten für Webserver gedacht, kommt aber mit weiteren nützlichen Programmen zur Verschlüsselung und für Prüfsummen.

Neben dem OpenSSL in der Basisversion (`/usr/bin/openssl`) gibt es OpenSSL noch als Paket in Pkgsrc, welches dann als `/usr/pkg/bin/openssl` installiert wird.

OpenSSL startet mit dem Befehl openssl(1) eine Shell, in der verschiedene Befehle zur Verfügung stehen. Man kann diese Befehle auch direkt als Option übergeben und ausführen lassen. Mit `version -a` werden Informationen zum installierten OpenSSL angezeigt, siehe Bsp. 12.14.

[!h]

```
$ openssl version -a
OpenSSL 0.9.8b 04 May 2006
built on: NetBSD 4.99.1
platform: NetBSD-i386
options:  bn(32,32) md2(int) rc4(ptr,int) des(ptr,risc2,16,long) blowfish(ptr2)
compiler: gcc version 4.1.2 20060628 (prerelease) (NetBSD nb2 20060711)
OPENSSLDIR: "/etc/openssl"
$
```

Listing 12.14: OpenSSL-Version anzeigen

Mit dem Schlüsselwort `dgst` bekommen Sie Zugriff auf die Prüfsummenverfahren. Neben diversen Ausgabeformaten (Binär/Hexadezimal) und Möglichkeiten Signaturen zu prüfen, können sie mit einem Schlüsselwort aus Abb. 12.1 wie in Bsp. 12.15 die passende Prüfsumme erstellen.

[!h]

```
$ openssl dgst -sha512 /netbsd
SHA512(/netbsd)= 8aee42db7987d47e75381f835ded8988a5c2b6c4fd \
           f9235dcac0ccba6a6f1ef1db7cfb403f3b15c782b6a52c74 \
                     3c9849938631789191512695215216798 1ca41
$ openssl dgst -md4 /netbsd
MD4(/netbsd)= e8eef667d884b5cdb52af682331e5231
```

Listing 12.15: Prüfsummen mit OpenSSL erstellen

Genau wie das `dgst`-Stichwort Prüfsummen erzeugt, können Sie mit dem Stichwort `enc` Daten ver- und entschlüsseln. Dazu übergeben Sie nach dem Stichwort `enc` die Option `-e` für verschlüsseln oder `-d` für entschlüsseln, den passenden Algorithmus

- md5
- md4
- md2
- sha1
- sha
- sha256
- sha512
- mdc2
- ripemd160

Tabelle 12.1.: in OpenSSL verfügbare Prüfsummenverfahren

wie in Tab. 12.2 aufgeführt und ggf. weitere Optionen für den Schlüssel-Hash oder ein Passwort bzw. einen Schlüssel. Bsp. 12.16 zeigt eine Ver- und Entschlüsselung einer Datei mit OpenSSL.

[!h]

```
$ openssl enc -aes-256-ofb -in netbsd -out netbsd.enc -e -salt
enter aes-256-ofb encryption password:
Verifying - enter aes-256-ofb encryption password:
$ openssl enc -aes-256-ofb -in netbsd.enc -out netbsd.out -d
enter aes-256-ofb decryption password:
$  ls -l netbsd*
-rwxr-xr-x  1 stefan  stefan  6845211 Sep 17 21:38 netbsd
-rw-r--r--  1 stefan  stefan  6845227 Sep 17 21:48 netbsd.enc
-rw-r--r--  1 stefan  stefan  6845211 Sep 17 21:49 netbsd.out
$
```

Listing 12.16: Dateien mit OpenSSL verschlüsseln

Im Kapitel Kap. **??**, »**??**« auf Seite **??** ff. wird die Einrichtung von SSL für Apache beschrieben.

Algorithmen			
-aes-128-cbc	-aes-128-cfb	-aes-128-cfb1	-aes-128-cfb8
-aes-128-ecb	-aes-128-ofb	-aes-192-cbc	-aes-192-cfb
-aes-192-cfb1	-aes-192-cfb8	-aes-192-ecb	-aes-192-ofb
-aes-256-cbc	-aes-256-cfb	-aes-256-cfb1	-aes-256-cfb8
-aes-256-ecb	-aes-256-ofb	-aes128	-aes192
-aes256	-bf	-bf-cbc	-bf-cfb
-bf-ecb	-bf-ofb	-blowfish	-cast
-cast-cbc	-cast5-cbc	-cast5-cfb	-cast5-ecb
-cast5-ofb	-des	-des-cbc	-des-cfb
-des-cfb1	-des-cfb8	-des-ecb	-des-ede
-des-ede-cbc	-des-ede-cfb	-des-ede-ofb	-des-ede3
-des-ede3-cbc	-des-ede3-cfb	-des-ede3-ofb	-des-ofb
-des3	-desx	-desx-cbc	-idea
-idea-cbc	-idea-cfb	-idea-ecb	-idea-ofb
-rc2	-rc2-40-cbc	-rc2-64-cbc	-rc2-cbc
-rc2-cfb	-rc2-ecb	-rc2-ofb	-rc4
-rc4-40	-rc5	-rc5-cbc	-rc5-cfb
-rc5-ecb	-rc5-ofb		

Tabelle 12.2.: in OpenSSL verfügbare Kryptoalgorithmen

Teil III

ANWENDUNGEN

13

ANWENDUNGEN

Si vis pacem, para bellum.

(Vegetius)

13.1 INSTALLIERTE PAKETE AUF SICHERHEITSLÜCKEN ÜBERPRÜFEN

Anwendungen lassen sich auf NetBSD am einfachsten mit Pkgsrc verwalten. Damit können Programme problemlos installiert, deinstalliert oder aktualisiert werden. Vereinfacht dargestellt ist Pkgsrc nichts anderes als ein paar Verzeichnisse mit Makefiles, die Abhängigkeiten und andere Optionen auflösen. Um die installierten Pakete zu verwalten wird eine Paketdatenbank angelegt, dazu werden unter /var/db/pkg Verzeichnisse mit Informationen zu den Paketen angelegt. Auf diese Informationen kann man mit pkg_info(1) zugreifen, so zeigt z. B. »pkg_info -L Paketname« alle vom Paket installierten Dateien an. Ein einfaches »pkg_info | sort« zeigt alle installierten Pakete mit ihren Beschreibungen alphabetisch sortiert an.

Um installierte Pakete auf Sicherheitslücken hin zu überprüfen, kann man das Paket Audit-Packages ([PP17]) verwenden. Mit den beiden Befehlen download-vulnerability-list(8) und audit-packages(8) wird zuerst eine menschen- und maschinenlesbare Liste mit Informationen zu unsicheren Paketen heruntergeladen und in pkgsrc/distfiles/ abgelegt. Anschließend kann mit audit-packages(8) eine Liste verwundbarer Pakete angezeigt werden, die auf dem System installiert sind. Diese Arbeit lässt sich via Cron und Email oder den täglichen Wartungsskripten automatisieren.

Um allgemein auf veraltete Pakete zu prüfen, können Sie pkg_chk(1) verwenden. Es wird als [PP17] installiert und zeigt mit der Option `-i` alle Pakete an, dies installiert sind, zu denen aber in Pkgsrc bereits eine neuere Version existiert. Die so gefundenen Pakete sind nicht zwangsläufig unsicher, sollten aber trotzdem schnellstmöglich aktualisiert werden.

[!h]

```
$ cd /var/db/pkg/teTeX-bin-3.0nb6/
$ ls
total 152
-rw-r--r--    1 root  wheel   4251 Feb 11  2006 +BUILD_INFO
-rw-r--r--    1 root  wheel   1863 Feb 11  2006 +BUILD_VERSION
-rw-r--r--    1 root  wheel     56 Feb 11  2006 +COMMENT
-rw-r--r--    1 root  wheel  26157 Feb 14  2006 +CONTENTS
-rwxr-xr-x    1 root  wheel   5150 Feb 11  2006 +DEINSTALL*
-rw-r--r--    1 root  wheel    216 Feb 11  2006 +DESC
-rwxr-xr-x    1 root  wheel   6289 Feb 14  2006 +DIRS*
-rw-r--r--    1 root  wheel    439 Feb 11  2006 +DISPLAY
-rwxr-xr-x    1 root  wheel  12098 Feb 11  2006 +INSTALL*
-rw-r--r--    1 root  wheel     10 Feb 11  2006 +SIZE_ALL
-rw-r--r--    1 root  wheel      9 Feb 11  2006 +SIZE_PKG
drwxr-xr-x    2 root  wheel    512 Feb 14  2006 ./
drwxr-xr-x  423 root  wheel  11776 Sep  4 18:30 ../
$ cat +DESC
This package contains the executables for the teTeX installation
as well as other files that were created when generating the binaries:
info files, format files and manual pages.

Homepage:
http://www.tug.org/tetex/
$ cat +SIZE_ALL
246875795
```

Listing 13.1: Aufbau der Paketdatenbank in /var/db/pkg

[!h]

```
# download-vulnerability-list && audit-packages
[...]
    Data traffic for this session was 201061 bytes in 1 file.
    Total traffic for this session was 205431 bytes in 1 transfer.
221 Thank you for using the FTP service on ftp.NetBSD.org.
Package vulnerabilities file has been updated
[...]
Package gd-2.0.33nb1 has a denial-of-service vulnerability (vulnid:unknown),
            see http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-2906
[...]
```

Listing 13.2: Installierte Pakete auf Sicherheitslücken prüfen

SIEHE AUCH:
http://www.pkgsrc.org

Teil IV

DATENSICHERUNG

14

STRATEGIEN ZUR DATENSICHERUNG

Niemand will Backup. Alle wollen Restore.

(Kristian Köhntopp)

14.1 DEFINITIONEN

Ein Backup ist die Sicherung relevanter Daten um diese vor Verlust oder Beschädigung zu schützen. Es ist ein Sicherungssystem für den Fall des Verlusts der Daten. Im allgemeinen Falle ist eine Datensicherung nicht mit der Archivierung der Daten gleichzusetzen, da in der Regel nur eine Momentaufnahme des Datenbestandes gesichert wird. Ist eine Archivierung des Datenbestandes notwendig[1], kann dies über spezielle Archivierungsprogramme erfolgen, oder in dem die Datensicherungen selbst archiviert werden. Eine weitere Möglichkeit Daten vor Verlust zu schützen ist die Spiegelung dieser auf andere Systeme.

14.2 VORBEREITUNG DER STRATEGIE

Da die Organisation einer Sicherungsstrategie äußerst komplex ist, ist es notwendig eine umfassende Anforderungsanalyse vorzunehmen und die gewonnenen Erkenntnisse umzusetzen. Hierzu gilt es einen Notfallplan[2] zu erstellen, der das Fundament für die Sicherungsstrategie bildet.

Die einzelnen Schritte umfassen:

[1] bspw. §239 Abs. 4 Satz 2 HGB *»Die Handelsbücher und die sonst erforderlichen Aufzeichnungen können auch in der geordneten Ablage von Belegen bestehen oder auf Datenträgern geführt werden, soweit diese Formen der Buchführung einschließlich des dabei angewandten Verfahrens den Grundsätzen ordnungsmäßiger Buchführung entsprechen. Bei der Führung der Handelsbücher und der sonst erforderlichen Aufzeichnungen auf Datenträgern muß insbesondere sichergestellt sein, daß die Daten während der Dauer der Aufbewahrungsfrist verfügbar sind und jederzeit innerhalb angemessener Frist lesbar gemacht werden können. Absätze 1 bis 3 gelten sinngemäß.«*

[2] Neudeutsch auch gerne als *»Disaster Recovery Plan«* bezeichnet.

1. Inventur
2. Bedrohungsszenarien analysieren
3. Wichtige Daten finden
4. Sicherungssysteme sichern
5. Dokumentation
6. Testen

Am einfachsten ist es, den Plan so aufzustellen, wie es im Idealfall (also ohne technische Einschränkungen) möglich wäre. Anschließend passt man den Idealplan an die reale Situation an, in dem man bspw. die Sicherungszyklen an die verfügbaren Zeitfenster anpasst. Steht der Plan und seine Implementierung, gilt es ihn zu testen, zu testen, zu testen und zu warten. Denn die zugrundeliegende Umgebung verändert sich und ebenso wie Sicherheitsstrategien, muss der Plan stetig an die neue Situation angepasst werden. Daher empfiehlt es sich, in der Regel alle sechs Monate, den Plan zu überprüfen und gegebenenfalls anzupassen.

14.3 Inventur

Ziel der Inventur ist es, einen Überblick über die eingesetzten Systeme zu bekommen. Dabei ist es notwendig zu katalogisieren, welche Hardware, Betriebssysteme, Anwendungsprogramme und Dienste auf welchen Maschinen laufen. Anhand dieser Inventur können Anforderungen an die Sicherungssoftware festegelegt werden.

Existiert schon ein Inventar, idealerweise in einer Datenbank oder einem speziellen Inventurprogramm, kann dieses um die Informationen bezgl. der Sicherung erweitert werden. Falls noch kein Inventar besteht, sollte beim Erstellen eines Neuen auf die Erfassung der notwendigen Daten zur Organisation der Sicherung geachtet werden.

14.4 Bedrohungsszenarien analysieren

»Wodurch werden meine Daten bedroht?« und »Wie kann ich mich vor diesen Bedrohungen schützen?« lauten die Fragestellungen zu den Bedrohungen.

In der Regel gibt es bestimmte Fehlertypen / Bedrohungen, die jeweils eine eigene Sicherungsstragie erfordern:

1. Benutzerfehler
   Der Benutzer löscht oder verfälscht Dateien. Hier muss die letzte vorhandene Version zurückgespielt werden. Dieser häufige Fehler erfordert eine Archivierung der Daten oder Sicherungsbänder.

2. Administratorenfehler
   Ist in der Regel eher selten, wenn er aber vorkommt, dann richtig. Schließlich kann sich `root` nicht nur einfach ins Knie schießen, sondern beide Beine wegblasen. Hier hilft meist nur die Rücksicherung des kompletten Systems, daher sollte man das *gesamte* System bspw. mit Snapshots sichern.

3. Systemfehler

   a) Festplatte defekt
      Leider auch recht häufig, reißt dies meist alle Daten (und oft das gesamte System) in den Tod. Hier helfen Sicherungen auf Wechselmedien, andere Rechner oder RAID-Systeme.

   b) Dateisystemkorruption
      Hier helfen nur *archivierte* Sicherungen, da sich ein Fehler auf die Spiegel bzw. Sicherungen fortpflanzt.

4. elektronischer Einbruch / Vandalismus / Diebstahl
   Einbruch auf elektronischem Wege und daraus resultierende Vernichtung der Daten[3] erfordert eine Rücksicherung der Daten. Im Falle schleichender Korruption[4] sind auch hier wieder Archive notwendig.

5. herkömmlicher Einbruch / Vandalismus / Diebstahl
   Was ist wenn jemand einbricht und alle Rechner klaut? Hier hilft die letzte Datensicherung weiter, aber wurde die auch gestohlen oder vernichtet?

6. Naturkatastrophen / Höhere Gewalt
   Glücklicherweise wird Deutschland nicht von Wirbelstürmen oder Erdbeben heimgesucht, aber Fluten sind an gewissen Standorten schon eine Gefahr. Weiterhin können auch andere Gefahren lauern, wie Flugzeugabstürze / -einschläge, Erdrutsche, Großbrände oder Unglücke in benachbarten Unternehmungen (Chemiewerk, E-Werk, Raffinierie).
   Was passiert also wenn unser Gebäude/Stadtteil/Stadt ausradiert wird und unser Unternehmen die Daten noch benötigt? Existieren Sicherungen an anderen Orten? Hier sollte entsprechend dem Gefahrenpotential Sicherungsbänder ausgelagert werden, bspw. in andere Filialen. Ebenso möglich ist eine Spiegelung/Replikation der Datenbestände auf entfernte Systeme.

[3] Die Daten müssen nicht unbedingt gelöscht worden sein, aber sind sie noch integer?
[4] Wurden Daten manipuliert? Wenn ja, wann?

14.5 WICHTIGE DATEN FINDEN UND ORGANISIEREN

Als erster Schritt ist es notwendig, alle zu sichernden Daten zu finden. Dies sind in der Regel alle Daten, deren Verlust nicht akzeptabel ist.

Dies hat drei Gründe:

- nicht wiederherstellbare Daten
  Die Daten sind bei Verlust nicht wiederherstellbar, da sie bspw. an einen bestimmten Zeitpunkt gebunden sind[5] oder die beteiligten Programme / Bearbeiter nicht mehr verfügbar sind

- wirtschaftlicher Totalschaden
  Die Daten sind zwar rekonstruierbar, der Aufwand dafür, oder die Nichtverfügbarkeit, übersteigt monetär den Aufwand der Datensicherung.

- Systemdaten
  Auf einem System liegen nicht nur einfach Daten vor, sondern es existiert mindestens ein Betriebssystem und in der Regel auch noch weitere Programme bzw. Dienste. Es empfiehlt sich auch diese Daten zu sichern. Moderne Sicherungsmedien stellen genügend Platz für Betriebssystemdateien zur Verfügung. Kann, oder möchte, man Betriebssystem und Anwendungen nicht sichern, sollte man zumindest deren Konfigurationsdateien sichern.

Eine finanzielle Bewertung der Daten hilft auch festzustellen, ob der Aufwand für die Datensicherung den Wert der Daten übersteigt.

Im Allgemeinen bietet es sich an Datenbestände aus Anwendungsprogrammen zu sichern, bspw. Datenbanken oder CAD-Systeme. Desweiteren sollten die Homeverzeichnisse der Benutzer regelmäßig gesichert werden. Konfigurationsdateien oder modifizierte Programme sollten einmalig bzw. nach Veränderungen gesichert werden.

Um die Datensicherung zu erleichtern, sollten die erzeugten bzw. verwendeten Dateien systematisch abgelegt werden und in Mehrbenutzerumgebungen / Netzwerken die Benutzer in die Struktur der Verzeichnisse (Was wird gesichert?) eingewiesen werden.

Bei der Organisation der Daten sollte man auch deren Sicherung im Blick haben. Hat man Daten, die nicht oft verändert werden, sollte man diese so in Verzeichnissen ablegen, daß die Sicherung vereinfacht wird. Meist umfasst dies Bilder, Fotos, Audiodateien oder Filme, die lediglich ein- oder zweimal auf CD oder DVD ge-

[5] z. B. Patientendaten, Logdateien, Versuchsprotokolle

brannt werden müssen und nicht im regelmäßigen Sicherungszyklus erfasst werden müssen.

Einige Programme legen Daten in Systempfaden und/oder im Homeverzeichnis des Benutzers ab. Werden nur die Homeverzeichnisse gesichert, sollte man Änderungen oder neue Dateien nur im Homeverzeichnis ablegen. Ein Beispiel dafür ist LaTeX, das standardmäßig die Pakete unter /usr/pkg/share/texmf ablegt. Zusätzlich kann man auch für jeden Benutzer ein eigenes Verzeichnis unter ~/texmf anlegen. Werden die Homeverzeichnisse regelmäßig gesichert, bietet es sich an die eigenen TeX-Pakete im Homeverzeichnis abzulegen.

Weiterhin ist es nicht notwendig Dateien zu sichern, die sich aus anderen Dateien erzeugen lassen. Dies umfasst beispielsweise PDF-Dateien, die mit LaTeX erzeugt werden. Es genügt die TeX-Dateien zu sichern. Oder Binaries, deren Quellcode gesichert wird, Fotos die für eine Webseite verkleinert und aufbereitet wurden etc. pp.

14.6 Sicherung der Sicherungssysteme

Einige Sicherungssysteme, insbesondere solche für Netzwerksysteme wie Amanda oder Bacula, verwenden einen Index. In diesem Index[6] werden alle Metadaten zur Sicherung abgelegt. Also Daten zu »wann wurde welche Datei von welchem Client in welcher Version auf welches Band gesichert«. Dieser Index ist also wichtig wenn man eine Rücksicherung einer bestimmten Version durchführen muss. Ohne den Index kann man zwar in der Regel noch die Bänder von Hand zurückspielen, dies ist aber bei einer größen Anzahl von Bändern recht mühselig. Daher sollte zwingend auch das Sicherungssystem selbst gegen Ausfälle und Verlust gesichert werden. Welche Daten hierbei zu sichern sind, hängt vom verwendeten Softwarepaket ab. Verwendet man eigene Skripte, die bspw. einen Katalog der gesicherten Dateien erstellen, sollte dieser ebenfalls gesichert werden.

14.7 Organisation der Sicherung

Hat man ein Netzwerk aus heterogenen Rechnern zu sichern, sollte man bereits bei der Einrichtung bzw. der Administration einige Punkte beachten.

[6] Logdateien bei Amanda, eine Datenbank bei Bacula

14.7.1 *Einweisung der Benutzer*

Insbesondere in Netzwerken, in denen nur Teile der Clients gesichert werden, ist es wichtig die Benutzer der Systeme über die Strategie zu unterrichten. Dabei ist darauf zu achten daß Anwenderdaten von den Benutzern in Verzeichnissen abgespeichert werden, die auch gesichert werden. Gegebenenfalls muss man hier mit sozialen Maßnahmen (Richtlinien, Abmahnungen, LARTs etc.) nachhelfen.

Dürfen die Benutzer auf den Clients selbständig Programme installieren, muss bereits im Vorfeld dafür gesorgt werden, daß deren Anwendungsdaten ebenfalls mitgesichert werden. Ist dies nicht möglich, muss ein Prozedere installiert werden, daß in solchem Falle die Sicherungsmechanismen anpasst.

14.7.2 *Cronjob vs. Batchjob*

Wenn man einzelne Systeme sichern möchte, kann man dies bspw. nachts per Cron erledigen. Dabei sollte man aber beachten, daß die entsprechenden Skripte robust sind und Toleranzen einplanen. Kommt es nämlich dazu, das sich die Laufzeit der Skripte verschiebt, kann unter Umständen der gesamte Sicherungsalgorithmus fehlschlagen.

Ein Beispiel aus der Praxis:
In einem Unternehmen sollten vier kleinere Arbeitsgruppenserver mit verschiedenen Unixvarianten gesichert werden. Dazu stand ein älterer Rechner mit Spoolingplatte und Streamer zur Verfügung. Nächtlich sollten die vier Server ihre Daten per dump und SSH-Tunnel auf den Spoolserver schieben, der diese danach auf ein Streamerband schreibt. Da alle Rechner im selben Raum standen, wurden sie über einen einfachen Switch extra für die Sicherung mit TBase100 verkabelt. Ein Cronjob auf den vier Servern startete mit einer Stunde Abstand den Dump-Lauf und auf dem Spoolingserver die endgültige Sicherung auf Band. Da das Backupnetzwerk eben nur für das Backup genutzt wurde, bemerkte niemand in der Firma daß ein Port am Switch massive Hardwareprobleme entwickelte und daher die Sicherung über das Netzwerk extrem langsam wurde. Zu dem Zeitpunkt als der Spoolingserver per Cron begann die Daten auf Band zu schreiben, lieferte der dritte Server seine Sicherung immer noch auf die Spoolingplatte ein. Der Spoolingserver verwendete zwar dump um das Band zu schreiben, so daß die anderen Sicherungen fehlerfrei waren, das Archiv vom dritten Server war aber defekt. Das ganze fiel erst auf, als ein Admin früher als gewöhnlich in der Firma auftauchte und bemerkte daß der Sicherungsprozeß noch läuft.

Nachdem der Netzwerkfehler behoben war, wurde das Skript abgeändert. Und

zwar so, daß nächtlich per Cron auf dem Spoolingserver ein Shellskript anlief, das nacheinander die Sicherungsläufe auf den Servern anwarf und erst nach erfolgreichem Abschluss dieser Sicherungen die Archive auf Band schrieb. Abschließend wurden die Logdateien aller Dump-Läufe per Mail an die Administratoren weitergeleitet – und von diesen auch gelesen.

14.7.3 *Homogene Systeme*

Meist ist es zwar nicht möglich alle Systeme homogen aufzusetzen, man sollte sich aber bemühen uniforme Systeme einzurichten und zu benutzen. Dazu gehört auch, daß man bspw. die Systemdienste gleich einrichtet, Benutzerdaten möglichst zentral verwaltet und gleiche Hardware einsetzt.

Dies vereinfacht die Administration des Netzes und damit auch die Sicherung bzw. Rücksicherung im Falle eines Falles.

14.7.4 *Dokumentation der Adminstration*

Die normalen administrativen Aufgaben sollten ebenso dokumentiert werden wie das Sicherungssystem selbst. Es ist nützlich den Arbeitsablauf zu dokumentieren, da man so auch später noch nachvollziehen kann, was wann warum wie gemacht wurde. Aber besonders nach einem Systemausfall und der Systemrestaurierung ist es nützlich, Aufzeichnungen über das System zu haben, da man bestimmt das ein oder andere Detail vergessen und die Daten dazu nicht gesichert hat. Dies ist insbesondere dann wichtig, wenn mehrere Administratoren in der EDV arbeiten. Man kann beispielsweise Blog-Systeme oder Wikis verwenden, um die Arbeit in Gruppen aufzuzeichnen. Dann darf man aber nicht vergessen, den Datenserver zu sichern :-)

14.7.5 *Shellskripte statt Handarbeit*

Meist kommt es vor, daß man einzelne Handgriffe wirklich manuell erledigt – sei es bspw. das Anlegen neuer Benutzer oder das Warten des Datenbankservers. Meist kann man die Befehle dazu auswändig und gibt sie von Hand ein. Praktischer ist es aber, diese Befehle in ein Shellskript zu gießen und ggf. per Cron ausführen zu lassen. Dies hat den Vorteil, daß die Skripte mitgesichert werden, also nach einem Systemausfall wiederhergestellt werden können. Außerdem kann man ein Skript leicht kommentieren und somit dokumentieren. Im Zweifelsfalle wissen also auch die Kollegen was auf den Maschinen abläuft.

14.7.6 *Logdateien erstellen und auswerten*

Für gewöhnlich können die meisten Sicherungsprogramme Logdateien erstellen. Dies sollte man auch aktivieren – und die erzeugten Protokolle unbedingt lesen. Denn nur so ist sichergestellt, daß alle Sicherungsläufe auch wie erwartet funktionieren.

14.7.7 *Verifikation der Daten*

Wann immer Daten übertragen oder bearbeitet werden, kann dabei etwas schief gehen. Daher sollte jedes Programm, das Daten verifizieren kann, dies auch tun.

Die Kompressionsprogramme Bzip2 und Gzip unterstützen mit der Option `-t` die Verifikation der komprimierten Daten.

Ansonsten kann man Daten mit Prüfsummen wie MD5 oder SHA1 (siehe Kap. 10.4) nach der Übertragung testen. Komplette Dateisysteme lassen sich mit mtree(8) (Siehe Kap. 10.1) überprüfen und vergleichen.

[!h]

```
$ gzip netbsd.gz
$ root@balmung {48} gzip -t netbsd.gz ; echo $?
0
$ root@balmung {49} date » netbsd.gz
$ root@balmung {50} gzip -t netbsd.gz ; echo $?
gzip: input not gziped (MAGIC0)
gzip: netbsd.gz: uncompress failed
1
$
$ md5 dump.0 > MD5
$ scp dump.0  stefan@backup:/home/backup/
$ ssh stefan@backup md5 /home/backup/dump.0 > MD5.1
$ diff MD5 MD5.1
```

Listing 14.1: Daten verifizieren

14.7.8 *Den GAU erwarten*

Im Endeffekt zielt die gesamte Organisation der Datensicherung nur auf einen Tag ab – den des Super-GAUs. Es wird der Tag kommen, an dem Mond und Sonne verschlungen werden, der Weltenbrand beginnt und das Schicksal der Götter besiegelt ist. Alle Rechner werden in Flammen stehen und ihre Daten verlieren – und nur ein

kleiner, unbedeutender Systemadminstrator mit einem Schrank voll Sicherungsbänder wird das Schicksal wenden können.

Und wehe Ihm er hat nicht alle benötigten Werkzeuge bereitliegen.

Wichtig ist es die Konfiguration aller Rechner und deren Software zu kennen. Am besten nicht (nur) als Datei, sondern auch als Ausdruck.

Dann benötigt man alle Installationsmedien für das Betriebssystem und eine Live-CD oder eine Festplatte mit einem eingerichteten NetBSD.

Eine Live-CD kann man mit pkgsrc/sysutils/mklivecd selbst erstellen, man sollte dazu einen Kernel verwenden, der alle benötigten Geräte (Laufwerke, Netzwerkkarten, CGD, RAID, LVM, etc. pp.) unterstützt und alle benötigten Pakete (z.B. mcrypt, Amanda) bereitstellen. Alternativ kann man auch eine kleinere Festplatte mit NetBSD einrichten und alle benötigten Programme einbinden.

14.7.9 *Alles sichern*

Das Sicherungssystem muss nach dem Prinzip »Erstmal alles sichern und den Rest ausschließen« angelegt sein. In der Praxis heißt dies bspw. daß alle Platten, die nach /home/ gemountet werden, gesichert werden sollen. Dies ist besonders praktisch, wenn bspw. ein Kollege eine weitere Platte für eine Arbeitsgruppe als /home/ag/ einbindet, dann aber vergisst diese Platte im Backupskript nachzutragen. Sichert das Skript allerdings alle Partitionen die in der /etc/fstab nach /home/ gemountet werden, wird diese neue Partition automatisch mitgesichert.

14.7.10 *Testsysteme und -läufe*

Es ist allgemein bekannt, das man nicht am Produktivsystem herumschrauben sollte, wenn sich dies vermeiden lässt. Ähnlich wie bei wichtigen Servern sollte auch für die Sicherungsanlage ein Testsystem bereitgestellt werden, an dem Testläufe und Versuche durchgeführt werden können.

Außerdem sollte regelmäßig eine Inventur der Medienbestände durchgeführt werden, in der neben deren Vollständigkeit auch die Funktionsfähigkeit[7] der Laufwerke und Medien getestet wird.

[7] Bänder sollten bspw. mindestens einmal pro Jahr einmal durchgespult werden.

14.7.11 *Verschiedene Sicherungskreise*

Aus Datenschutz- oder rechtlichen Gründen oder technischen Gegebenheiten kann es notwenig werden das Netzwerk in verschiedene Sicherungskreise zu unterteilen, die entsprechend behandelt werden.

14.7.12 *Maximale Archivierungsdauer*

So wie der Gesetzgeber für einige Daten eine Mindestarchivierung vorschreibt, legt er auch bestimmte Obergrenzen fest. So sind beispielsweise Eintragungen in der Personalakte über Disziplinarmaßnahmen nach Eintritt des Verwertungsverbots[8] von Amts wegen zu entfernen und zu vernichten. Das heißt auch, das in dieser Frist angelegte Sicherungskopien der Akte zu vernichten sind. Daher kann es sinnvoll sein bspw. die Personalabteilung in einen eigenen Sicherungskreis zu gliedern und auch entsprechend zu archivieren, da man sonst erfolgreich juristisch belangt werden kann.

14.7.13 *Wochenende und Feiertage beachten*

In Unternehmungen, in denen nicht rund um die Uhr, oder zumindest jeden Tag, gearbeitet wird, wird oftmals nur an den regulären Werktagen gesichert. Dies kann problematisch werden, wenn doch einmal an einem Feiertag oder Wochenende gearbeitet wird. Denn wie es Murphy's Law will, wird es dann garantiert zu einem Datenverlust kommen.

Sind die entsprechenden Rechner auch außerhalb der Werkszeiten an, können sie über das Netzwerk gesichert werden. Falls nicht, sollte in den Netzwerkrichtlinien und Benutzerordnungen auf die fehlende Sicherung hingewiesen werden, und Maßnahmen ergriffen werden, die den Benutzern eine Sicherung der Daten ermöglicht. Dies kann bspw. über CD-Brenner, ZIP-Disketten oder USB-Sticks erfolgen. Falls Wechseldatenträger verboten sind, kann man bspw. einen Fileserver bereitstellen, auf den die Mitarbeiter Kopien der zu sichernden Dateien ablegen können.

[8] Meist zwei Jahre.

14.8 DOKUMENTATION

Es muss sichergestellt werden, daß das gesamte Prozedere auch von anderen Administratoren als dem Implementierenden ausgeführt werden kann. Daher ist es lebenswichtig, eine funktionierende Dokumentation des gesamten Systems zu erstellen. Idealerweise sollte hier der gesamte Entwurf und die gesamte Implementierung des Systems formal dargelegt werden – zusätzlich aber auch eine einfache und leicht verständliche Schritt-für-Schritt-Anleitung zur Rücksicherung von Daten.

Anhand der Dokumentation sollte ein Administrator, der bisher noch nicht mit dem Sicherungssystem gearbeitet hat, in der Lage sein eine Rücksicherung durchzuführen. Es ist unerlässlich mehr als eine Person im Sicherungssystem zu unterweisen, denn schließlich kann diese nicht rund um die Uhr verfügbar[9] sein.

Nützlich ist es, die Einweisung in das Sicherungssystem mit der Dokumenation als Testlauf anzusetzen. Hierbei sollten weitere Administratoren anhand der Dokumentation eine Rücksicherung auf ein Testsystem durchführen, wobei der Entwickler des Systems lediglich beobachtet und nicht eingreift. So kann bequem die Qualität der Dokumentation (Versteht man was dort beschrieben wurde? Fehlen Teile?) und die Funtionstüchtigkeit des Sicherungssystems (Gelingt die Rücksicherung?) überprüfen. Nach Abschluss der Rücksicherung wird eine Manöverkritik durchgeführt, in der gemeinsam Schwächen und Fehler des Systems und der Dokumentation durchgesprochen und abgestellt werden.

Wie hoffentlich schon ersichtlich wurde, ist es wichtig nicht nur einen Administrator in das Sicherungssystem einzuweisen, sondern auch hier mehrere Personen auszubilden. Insbesondere in kleinen bis mittleren Unternehmungen, die keine eigentliche EDV-Abteilung sondern nur einen »Teilzeit-Admin« haben, ist dies oftmals ein Problem. Lösungen lassen sich dann evtl. über externe Berater finden, die im System eingewiesen und vertraglich entsprechend in den Ablauf eingebunden werden.

14.9 TESTEN, TESTEN, TESTEN

Ein Sicherungssystem, das nicht unter realen Bedingungen getestet wurde, kann und darf nicht als funktionierend betrachtet werden. Dies hat mehrere Gründe:

- Der Administrator muss in der Lage sein die Rücksicherung durchzuführen. In der Regel erzeugt ein Systemausfall mit Datenverlust Stress. Wenn dann zusätzlich der Chef das Problem am besten gestern gelöst sehen will, wird der Administrator zusätzlich unter Druck gesetzt. In solch einer Situation muss man aber einen kühlen Kopf bewahren um die Situation nicht weiter eskalieren zu lassen. Daher ist es unerlässlich, derartige Situationen und Verfahren in Ruhe durchzuspielen, um Sicherheit im Ablauf zu bekommen. Ist man an den Ablauf der Prozedur gewöhnt, verringert dies den auftretenden Stress und somit Fehlerquellen

[9] Freigang auf dem Hof, Fütterungszeiten u. Ä.

- Es ist nicht sichergestellt, daß die Sicherungsstrategie auch wirklich funktioniert. Man kann zwar versuchen im Entwurfsprozeß der Strategie Fehlerquellen zu minimieren, kann dies aber, wie in jedem Entwurfsprozeß überhaupt, nicht ausschließen. Nur durch Testläufe lässt sich sicherstellen, das die Prozedur funktioniert und auch wirklich die gewünschten Daten gesichert werden. Daher sollte man verschiedene Testläufe durchführen, in denen mögliche Fehler/Situationen betrachtet werden:
  - Rücksicherung von n einzelnen Dateien
  - Rücksicherung von n Versionen verschiedener Dateien
  - Rücksicherung eines kompletten Dateisystems (eines Clients) und Vergleich der Rücksicherung mit dem echten System
  - ein komplettes System (Betriebssystem mit Konfiguration plus Anwendungssoftware und Daten) wieder aufsetzen
  - Rücksicherung einer Archivversion zu einem bestimmten Zeitpunkt in der Vergangenheit (*Point-in-Time-Recovery*)
  - Rücksicherung eines Systems obwohl der Backup-Server ausgefallen ist

14.10 Verminderung von Ausfällen und Ausfallzeiten

Neben der eigentlichen Datensicherung ist es erstrebenswert Ausfälle erst gar nicht auftreten zu lassen. Hierzu existieren ebenfalls verschiedene Strategien, wie bspw. »hochverfügbare Systeme« (*HV* oder engl. *HA*) oder Replikation. Wünscht man hochverfügbare Systeme, muss man in entsprechende Technik investieren. Dies setzt unter anderem redundante Bauteile (Netzteile, Laufwerke), zuverlässige Hardware, RAID-Systeme, entsprechende Stromversorgung und im Betrieb austauschbare Hardware voraus. Auch wenn man so unter Einsatz entsprechender finanzieller Mittel ein hochverfügbares System installiert, ist immer noch eine Datensicherung erforderlich.

Selbst wenn eine Verfügbarkeit von 99,99999% (das heißt maximal 5,3 Minuten Ausfallzeit pro Jahr) erreicht werden kann, ist das System nicht vor Benutzer- und Bedienfehlern oder höherer Gewalt sicher.

Bei einer Spiegelung werden Daten idealerweise sofort (also direkt beim Schreiben) auf ein anderes System übertragen. Dies ist bspw. bei RAID-5 Systemen der Fall, wo Daten auf mindestens drei Festplatten verteilt werden und so bei Ausfall einer Platte die Daten integer bleiben.

Abzuraten ist hierbei dringendst von RAID-0, dem sogenannten »Striping«. Hierbei werden zur Geschwindigkeitssteigerung mindestens zwei Festplatten zu einer Logischen zusammengefasst. Bei diesem System multipliziert sich auch die Verfügbarkeit[10] beider Fesplatten. Da die Verfügbarkeit immer kleiner Null ist, wird die Gesamtverfügbarkeit logischerweise mit jeder weiteren Platte kleiner.

[10] $\mathrm{Verfügbarkeit} = \frac{\mathrm{meantime\ between\ failure}}{\mathrm{meantime\ between\ failure} * \mathrm{meantime\ to\ repair}}$

Eine ideale Form der Spiegelung ist die Replikation eines laufenden Systems auf ein identisches, bereitstehendes Ersatzsystem (»hot standby«) das bei Ausfall des Originals sofort einspringen kann. Diese Variante kostet mehr als das Doppelte eines Einzelsystems, außerdem ist sie nur sinnvoll wenn der Nutzen den Aufwand rechtfertigt, man also auf entsprechende Hochverfügbarkeit angewiesen ist. Wichtig ist hierbei vor allem auch eine räumliche Trennung (mindestens eine Brandschutzschleuse) der Systeme. Da beide Systeme in der Regel rund um die Uhr laufen sind sie den üblichen elektrischen Gefährdungen (Stromausfall, Blitzschlag, Kurzschluss) ausgesetzt und müssen dagegen gesichert werden. Diese Variante hat den Vorteil, das bei ausreichendem Schutz des Ersatzsystems dieses das ausgefallene System mit minimalem Zeitaufwand ersetzen kann. Nachteilig sind hierbei die Kosten.

Besonders im Datenbankenbereich sind schon seit Jahren erfolgreich Replikationsmechanismen etabliert. Für PostgreSQL existieren mehrere Programme, die synchrone oder asynchrone Replikation ermöglichen. Mit einem synchronen Replikationssystem, wie beispielsweise Pgpool (siehe Kap. **??**, Seite **??**), werden alle SQL-Anforderungen auf mehrere verteilte Systeme repliziert, so daß der Datenbestand auf mehreren unabhängigen Rechnern vorliegt. Da diese Rechner räumlich getrennt aufgestellt werden können, lassen sich die Daten vor einem Totalverlust schützen.

14.11 Datenrettung

Ist es notwendig Daten nach einem Headcrash oder einem Brand wiederherzustellen, muss man die Daten von den beschädigten Festplatten retten. Hierzu sollte man unbedingt, insbesondere im professionellen Bereich, spezialisierte Datenrettungsunternehmen beauftragen. Diese verfügen über die notwendige Erfahrung und Technologien um beschädigte Medien wiederherzustellen.

Sind Daten versehentlich gelöscht wurden (rm, Partitionierung/Formatierung) können diese u.U. mit Datenrettungsprogrammen (bspw. pkgsrc/sysutils/gpart) wiederhergestellt werden. Dabei sollte man grundsätzlich nicht auf dem betroffenen Medium arbeiten, insbesondere Schreibzugriffe sind zu verhindern. Alle Datenrettungsversuche dürfen nur auf einer 1:1-Kopie (siehe Kap. 16.3, S. 228) des Mediums erfolgen.

Sollen die Daten (trotz der recht hohen Kosten) von einem Datenrettungsunternehmen wiederhergestellt werden, sollte man jeglichen Zugriff auf die Medien unterlassen und die Profis arbeiten lassen.

15 SICHERUNGSVARIANTEN

Kenntnisse kann jeder haben, aber die Kunst zu denken ist das seltenste Geschenk der Natur.

(Friedrich II von Preußen)

15.1 KOMPLETTBACKUP

Ein Komplettbackup sichert alle relevanten Daten. Dies ist in der Regel sehr einfach zu bewerkstelligen, außerdem ist das Zurückspielen der Daten ebenfalls recht einfach. Nachteile sind dabei vor allem die Dauer der Sicherung und der benötigte Platz, da auch Daten gesichert werden, die seit dem letzten Backup nicht verändert wurden.

15.2 DIFFERENTIELLES BACKUP

In Anlehnung an das mathematische δ auch Delta-Methode genannt. Um Platz und Zeit zu sparen werden hierbei nur Daten gesichert, die seit dem letzten Komplettbackup geändert wurden. Bei der Rücksicherung muss allerdings die Reihenfolge der Bänder beachtet werden, im allgemeinen benötigt man hierzu die letzte Komplettsicherung[1] und das letzte Tagesband.

15.3 INKREMENTELLES BACKUP

Hier werden nur die Dateien gesichert, die sich seit dem letzten Inkrementalbackup geändert haben. Bei der Rücksicherung muss ebenfalls die Reihenfolge beachtet werden, dafür ist bei der Sicherung noch weniger Platz und Zeit als beim Differentialbackup erforderlich.

15.4 DUMP-LEVEL

Für gewöhnlich unterstützen ordentliche™ Sicherungsprogramme den Einsatz sogenannter Level, die mittels einer Zahl die zu sichernden Dateien angeben. Auf dem

[1] meist montags oder freitags erstellt

Level n werden alle Dateien gesichert, die seit der Sicherung mit $n-1$ verändert wurden.

Vorteil ist hierbei die Zeit- und Platzersparnis, Nachteil ist die aufwändigere Rücksicherung.
Man kann hierbei aber auch die Methoden mischen, so ist es beispielsweise praktikabel montags eine Komplettsicherung mit Level 0 durchzuführen und werktags, also dienstags – freitags, mit Level 1 die Arbeitsdaten der Woche zu sichern. Da es auch am Wochenende zu Arbeitseinsätzen kommmen kann[2], können am Sonnabend und Sonntag mit Level 2 die Tagesdaten gesichert werden.

15.5 Beispielstrategie

Eine einfache Beispielstrategie für einen einzelnen Rechner mit Leveln:

- Am Montag existiert 1 GB an Daten
- es werden jeden Tag 100 MB Daten erzeugt
- am Sonntag liegen 1,6 GB Daten vor
- Sicherungen erfolgen in der Nacht, vor Arbeitsbeginn
- Sonntag Mittag sei eine Rücksicherung vom Band notwendig

Tabelle 15.1 zeigt die verwendeten Dumplevel, die Tabellen 15.2, 15.3 und 15.4 zeigen die anfallende und gesicherte Datenmenge.

	Mo	Di	Mi	Do	Fr	Sa	So
Komplett	0	0	0	0	0	0	0
Differentiell	0	1	1	1	1	1	1
Inkrementell	0	1	2	3	4	5	6

Tabelle 15.1.: Wochensicherung mit Leveln

	Mo	Di	Mi	Do	Fr	Sa	So
Komplett	0	0	0	0	0	0	0
Tagesration	1 GB	1,1 GB	1,2 GB	1,3 GB	1,4 GB	1,5 GB	1,6 GB
Gesamt	1 GB	2,1 GB	3,3 GB	4,6 GB	6,0 GB	7,5 GB	**9,1 GB**

Tabelle 15.2.: Datenmenge einer Komplettsicherung

[2] Vorausgesetzt das Wochenende ist kein normaler Arbeitstag.

	Mo	Di	Mi	Do	Fr	Sa	So
Inkrementell	0	1	1	1	1	1	1
Tagesration	1 GB	0,1 GB	0,2 GB	0,3 GB	0,4 GB	0,5 GB	0,6 GB
Gesamt	1 GB	1,1 GB	1,3 GB	1,6 GB	2,0 GB	2,5 GB	**3,1 GB**

Tabelle 15.3.: Datenmenge einer Inkrementellsicherung

	Mo	Di	Mi	Do	Fr	Sa	So
Differentiell	0	1	2	3	4	5	6
Tagesration	1 GB	0,1 GB	0,1 GB	0,1 GB	0,1 GB	0,1 GB	0,1 GB
Gesamt	1 GB	1,1 GB	1,2 GB	1,3 GB	1,4 GB	1,5 GB	**1,6 GB**

Tabelle 15.4.: Datenmenge einer Differentiellsicherung

Die Komplettsicherung sichert jeden Tag alle Daten. Die Rücksicherung erfolgt vom letzten Band, also dem Sonntagsband.

Differentiell sichert man jeden Montag alle Daten und dienstags bis freitags auf Level 1. Die Rücksicherung erfolgt erst mit dem letzten Montagsband und anschließend dem letzten Tagesband.

In der inkrementellen Sicherung sichert man jeden Montag alle Daten und wochentags inkrementell, also nur Veränderungen zum Vortag. Die Rücksicherung erfolgt hier in der Reihenfolge aller Bänder von Montag bis Sonntag. Nachteil der Strategie ist, daß Dateien in der Regel nur auf einem einzigen Band gesichert sind, was bei Verlust des Bandes dem Verlust aller Daten des Tages und der darauffolgenden Tage gleichkommt.

Daher gibt es noch eine andere, platzsparende Strategie, die an die »Türme von Hanoi« angelehnt ist.

Diese Strategie (siehe Tab. 15.5) verteilt einzelne Dateien auf mehrere Bänder, wobei bei diesem Algorithmus aber am wenigsten Platz verschwendet wird. Für einen gesamten Monat, lässt sich der Algorithmus mit Level-1 Sicherungen an den folgenden Montagen erweitern. Vorteil der Strategie ist hierbei die Sicherung der Dateien auf mehr als ein Medium, ohne jedesmal alles sichern zu müssen.

Die Verteilung der Dateien auf die Bänder wird in Tabelle 15.6 illustriert.

Mo	Di	Mi	Do	Fr	Sa	So
0	3	2	5	4	7	6
1	3	2	5	4	7	6
1	3	2	5	4	7	6
1	3	2	5	4	7	6

Tabelle 15.5.: Türme-von-Hanoi-Algorithmus

Wochentag	Daten auf dem Tagesband
Montag	Montag
Dienstag	Montag & Dienstag
Mittwoch	Montag & Dienstag & Mittwoch
Donnerstag	Mittwoch & Donnerstag
Freitag	Mittwoch & Donnerstag & Freitag
Sonnabend	Freitag & Sonnabend
Sonntag	Freitag & Sonnabend & Sonntag
Montag	1. Montag - 2. Montag
Dienstag	2. Montag & 2. Dienstag
...	

Tabelle 15.6.: Datenverteilung im Hanoi-Algorithmus

15.6 Komprimierung und Verschlüsselung

Oftmals ist es nützlich ein Archiv zu komprimieren um Platz zu sparen. Jedoch sei hierzu angemerkt, daß bei Archivfehlern ein Dekomprimieren der gepackten Archive oftmals nicht mehr möglich ist. Daher sollte man die Kosten und Nutzen abwägen und die gesicherten Archive jedesmal verifizieren lassen. Ebenso ist es sinnvoll nur gängige Komprimierungsprogramme zu verwenden, um später wieder an die Daten heranzukommen.

Aktuelle Bandlaufwerke unterstützen in der Regel Hardwarekomprimierung. Das heißt daß das Laufwerk vor dem eigentlichen Schreiben der Daten diese in einem eigenen Chip komprimiert. Dieses Verfahren hat den Vorteil die sichernde Maschine nicht weiter zu belasten.

Ob man Komprimierungsverfahren einsetzen sollte, hängt von der entstehenden Last bzw. Einsparung (Netzwerk ./. CPU ./. Bandbedarf) und von der Zusammensetzung der Daten ab. Sind die Daten in der Mehrzahl (natürlichsprachige) Texte (Logdateien, Textverarbeitung, Datenbanken mit Texten) ist eine recht große Kompressionsrate gegeben. Bereits komprimierte (z. B. JPEG, MP3, AVI) oder verschlüsselte Datentypen hingegen lassen sich nicht weiter komprimieren.

Es ist sinnlos ein verschlüsselndes Dateisystem einzusetzen und unverschlüsselte

Backups im Schrank liegen zu haben, daher sollten auch Backups verschlüsselt werden. Dies gilt insbesondere dann, wenn die Sicherungsbänder ausgelagert werden sollen.

Zur Komprimierung eignen sich die Standardprogramme gzip und bzip2[3], da diese relativ weit verbreitet sind und bisher zuverlässig arbeiten.

Verschlüsseln kann man die Archive sicher mit OpenSSL, GnuPG oder dem symmetrischen Verschlüsselungprogramm mcrypt, welches verschiedene Algorithmen wie Rijndael, 3DES oder Panama unterstützt.

Werden die Archive nicht mittels einer Pipe direkt verschlüsselt auf Datenträger geschrieben, müssen die unverschlüsselten Dateien sicher gelöscht (»gewipet«) werden. Dazu gibt es die Option `-P` für rm, die Dateien vor dem Löschen dreimal überschreibt. Diese Option genügt aber nicht den allgemein anerkannten Regeln der Technik. Daher sollte hierfür dringend ein Programm verwendet werden, das zumindest den US DoD 5220.22-M ECE oder Peter Gutmanns Standard implementiert. In pkgsrc befindet sich dazu pkgsrc/security/destroy oder pkgsrc/sysutils/wipe. Empfehlenswert ist es allerdings, die unverschlüsselten Archive in eine mit CGD verschlüsselte Partition zu schreiben und danach ebenfalls zu wipen. Hierzu kann man auf einzelnen Rechnern eine hinreichend große /tmp-Partition erstellen, die mit CGD verschlüsselt ist.

Verwendet man GnuPG zum Verschlüsseln der Archive, sollte man auf asymmetrische Verschlüsselung verzichten und Symmetrische einsetzen. Denn für die Entschlüsselung benötigt man sonst wieder den GnuPG-Schlüssel – was das Archiv unbrauchbar macht, wenn der nicht mehr existiert.

[!h]

```
# dump -0a -f - /etc/ | bzip2 > dump.bz2
# gpg -a -c dump.bz2 && destroy -f -s 7 t dump.bz2
# dump -0a -f - /etc/ | bzip2 | openssl aes-256-ecb -out dump.bz2.enc -e -salt
# openssl aes-256-ecb -in dump.bz2.enc -d -salt | bunzip2 - | restore -r -f -
```

Listing 15.1: Archive komprimieren und verschlüsseln

Listing 15.1 erzeugt einen Dump, der sofort per bzip2 komprimiert wird. Anschließend wird der dump mit GnuPG symmetrisch verschlüsselt und mit destroy sicher gelöscht. Zeile drei erzeugt einen Dump der sofort per Pipe an bzip2 zum komprimieren und OpenSSL zum Verschlüsseln geschickt wird. Die vierte Zeile ist des Gegenstück zur Dritten, denn hier wird der verschlüsselte und komprimierte Dump entschlüsselt, dekomprimiert und an restore geschickt.

[3] bzip2 erzeugt zwar im Allgemeinen eine bessere Komprimierungsrate, erzeugt dabei aber auch höhere Systemlast und Laufzeiten

15.7 MEDIEN

Als Medium eignet sich prinzipiell alles was am Markt erhältlich ist, jedoch sollte man einige Vorüberlegungen treffen.

1. Verlässlichkeit
   Die Medien müssen zuverlässig sein und auch längere Archivierungsperioden problemlos überstehen. Außerdem müssen Lesegeräte noch verfügbar sein, um die Sicherungsmedien einlesen zu können. Dies ist besonders wichtig, wenn Medien über mehrere Jahre archiviert werden müssen. Hier sollte man regelmäßig (bspw. alle sechs Monate) zusammen mit der Inventur prüfen ob die ältesten Medien noch lesbar sind und ggf. die Medien auswechseln. Im allgemeinen muss man bei der Archivierung spätestens alle Acht bis Zehn Jahre eine komplette Migration der Medien durchführen, da die Medien und Lesegeräte veraltet sind.
2. Geschwindigkeit

   Das Sicherungssystem muss so schnell wie möglich sein. Dabei sollte man aber auch die Kapazitäten des Netzwerks und des Sicherungsservers beachten, denn es ist übertrieben, einen 50-MBit-Streamer einzusetzen, wenn die Sicherung in einem 10-MBit-Netz erfolgt.
   Ein weiterer Faktor, der die Geschwindigkeit des Sicherungslaufwerks beeinflusst, ist ein Bandwechsel. Wenn ein Sicherungslauf auf mehrere Bänder verteilt werden muss, senkt die Wartezeit auf das neue Band den Gesamtdurchsatz des Systems.
3. Dauer der Rücksicherung (sog. *Time to Data*)
   »Wie lange benötige ich um bei einer Rücksicherung an bestimmte Daten zu kommen?«
   Hierbei spielen verschiedene Faktoren eine Rolle, wie Zugriffszeit der Medien aber auch die Organisation der Archivierung selbst.
   Mehr als 90% aller Rücksicherungen betreffen nur einzelne Dateien, die in älteren Versionen restauriert werden müssen. Der gesamte Zeitaufwand der Operation besteht daraus, die richtigen Medien mit der gewünschten Version zu finden, einzulegen und die Dateien zurückzuspielen. Verwendet man hierzu ein automatisiertes System (Index, Bandwechsler) geht dies wesentlich schneller als wenn man die Daten von Hand zurückspielen muss.
   Ist man auf besonders schnelle Zugriffszeiten angewiesen, sollte man zur Sicherung hierarchische Speichersysteme[4] verwenden.

[4] Hierbei werden verschiedene Medientypen mit unterschiedlichen Zugriffzeiten eingesetzt. Häufig benötigte Dateien werden auf schnellen Medien gesichert (Festplatte, Magneto-Optische Medien), während ältere bzw. seltener benötigte Daten auf entsprechend langsameren aber größeren Medien (Magnetbänder) gesichert werden. Diese Methode war früher, als Festplatten noch in Megabyte vermessen wurden, sehr beliebt und wird heute noch in einigen sehr datenintensiven Umgebungen (Grafik-/Videobearbeitung) eingesetzt.

4. Kapazität
   Die anfallenden Datenmengen müssen auf möglichst wenige Medien verteilt werden. Verwendet man einen einfachen Streamer, ist es besonders wichtig daß alle Daten auf ein Band passen, denn niemand möchte nachts um drei neben dem Streamer Wache schieben.
   Beachten sollte man auch das eine steigende Kapazität die Zeit zur Rücksicherung anhebt, denn ein 10GB Band lässt sich schneller durchspulen als ein 100GB Band.
   Aus betriebswirtschaftlicher Sicht ist eine Betrachtung der Kosten unerlässlich, denn es ist unnötig einen 100GB-Streamer mit entsprechend großen und teuren Medien anzuschaffen, wenn täglich nur 5GB Daten anfallen. Andererseits sollte man auch nicht zu kurz planen, da die Hardware wenigstens die vier Jahre bis zur Abschreibung im Einsatz bleiben sollte.
5. Kosten
   Dabei darf das System üblicherweise nichts Kosten. Denn wozu braucht man schon ein Backupsystem, wenn doch alles prima läuft?[5]

Disketten, ZIP-Medien, CD/DVD und Festplatten sind weit verbreitet, billig und von daher gut zur Rücksicherung geeignet. Nachteilig wirkt sich hierbei allerdings die relativ geringe Integrität und die kurze Lebensdauer der Datenträger aus. Relativ kostspielig und langsam, aber dennoch hervorragend geeignet aufgrund hoher Kapazitäten und exzellenter Integrität und Lebensdauer sind Magnetbänder.

Selbstverständlich bietet es sich auch hier wieder an verschiedene Medien zu nutzen und zu mischen, z. B. ZIP-Disketten oder CDRW/DVDRW für tägliche Sicherungen.

Festplatten können aufgrund der hohen Kapazität und Geschwindigkeit auch zur Datensicherung eingesetzt werden, sollten dann aber als Wechselplatte ausgeführt sein und außerhalb der Sicherungsläufe offline gelagert werden.

Medien	Verfügbarkeit	Geschwindigkeit	Haltbarkeit	€/GB
DVD+RW	++	+	-	0,3€
USB-Stick	++	+	+	70,00€
ZIP250	○	-	○	32,00€
Platten	++	++	-	0,7€
Bänder	+	○	++	1-5,00€

Tabelle 15.7.: Vergleich von Sicherungsmedien

15.7.1 *Organisation und Lagerung der Medien*

Bei großen Datenmengen und niedrigen Migrationszyklen, sowie der Archivierung der Medien, ist eine ausgefeilte Organisation der Lagerung notwendig.

[5] man Zynismus

Zuerst ist die Lagerung der Medien ihren physikalischen Bedürfnissen anzupassen. Für gewöhnlich befinden sich in deren Spezifikation Angaben zur besten Luftfeuchtigkeit, Umgebungstemperatur, Luftdruck und so weiter und sofort. Insbesondere magnetische Datenträger sollten vor starken Temperaturschwankungen geschützt werden. Bänder müssen aufrecht stehend gelagert werden, da sich sonst das Magnetband lockern kann – und niemand will Bandsalat bei einer Rücksicherung erleben.

Physikalische Sicherungen (gegen Einbruch) und Brandschutzeinrichtungen sind ebenso obligatorisch wie eine restriktive Zugangspolitik. Hierzu kann man gesonderte Räume einrichten, die durch entsprechende Türen und Schlösser gesichert werden. Hat man diese Möglichkeiten nicht, bietet sich ein Tresor[6] für die Medien an.

Da die Anzahl der Medien mit der Zeit wächst und die Archivierung komplexer wird, bietet sich ein datenbankengestütztes Inventarisierungssystem an. Hierzu sollten mindestens folgende Attribute aufgenommen werden:

- Primärschlüssel
- Name
- Zweck
- Medientyp
- gesicherte Dateien inkl. Datum / Version, Eigentümer, Attribute
- Lagerort

Der Primärschlüssel dient der Identifikation des Mediums und kann bspw. eine laufende Nummer sein, sinnvoller ist es aber in ihm Informationen zu codieren (bspw. Datum, gesicherte Daten, IP-Adresse, Level oder ähnliches). Der Primärschlüssel muss mindestens auf jedem Medium und der entsprechenden Hülle angebracht werden. »Name« kann ggf. verwendet werden, wenn man die Medien nicht über den Primärschlüssel ansprechen möchte. Zusätzlich muss ebenfalls der Zweck, Medientyp und der Lagerort der Medien katalogisiert werden.

In der Datenbank werden auch die Metadaten der gesicherten Dateien erfasst. Dies ist notwendig, um im Falle einer Rücksicherung das passende Sicherungsband zu finden.

Für große Datenmengen existieren auch automatische Bandwechseleinheiten, die die Medien und Hüllen mit einem Strichcode etikettieren können. In Verbindung mit einem einfachen Handscanner ist dies ideal um größere Medienbestände zu inventarisieren und zu verwalten.

15.8 Prüfliste zur Strategie

[6] Beispielsweise ein echter *Franz Jäger, Berlin*.

* **Wer** ist verantwortlich?
* **Wie** wird gesichert?
* **Wann** wird gesichert?
* **Was** wird gesichert?
* **Welche** Medien und Geräte werden verwendet?
* **Wo** werden die Medien gelagert?
* **Wie** lange werden die Medien gelagert? .
* **Wie/Wann** wird die Integrität der Sicherung geprüft?
* **Welche** Backupstrategie? (Wann welche Level?)
* **Wo** ist die Dokumentation zur Backupstrategie?

Tabelle 15.8.: Prüfliste zur Strategie

16 PROGRAMME ZUR DATENSICHERUNG

Nicht kleckern, klotzen!

(Generaloberst Heinz Guderian)

16.1 DUMP(8), DUMP_LFS(8) UND RESTORE(8)

Dump und Restore sind historisch gewachsen die bedeutendsten Backupprogramme unter Unix. Sie arbeiten hierbei auf Blockebene, also unterhalb der Dateischicht. Dies heisst das Dump einen Verzeichnisbaum oder ein Dateisystem[1] komplett mit allen Eigenschaften sichert und nicht auf Dateiebene Dateien kopiert. So ist es quasi möglich eine Partition zu spiegeln und zurückzusichern oder auch auf andere Rechner zu übertragen.
Da Dump speziell an das Dateisystem angepasst ist, kann dump(8) nur FFS sichern, für LFS gibt es dump_lfs(8). Die beiden Programme unterscheiden sich aber in der Bedienung nicht. Restore entpackt einen dump aber dateisystemunabhängig, kann also auch bspw. auf einem NFS-Verzeichnis arbeiten.

Dump unterstützt inkrementelle Backups durch die Verwendung sogenannter Dumplevels, dabei werden bei jedem Backup in der Datei /etc/dumpdates die entprechenden Daten eingetragen und können so später ausgelesen und aufbereitet werden. Ein Dump mit Dumplevel 0 erzeugt ein Komplettbackup, während ein Level größer 0 die Änderungen seit dem letzten Backup inkrementell sichert. Level 2 sichert also alle Daten, die seit dem letzten Dump mit Level 1 erzeugt oder geändert wurden. Die Daten dazu werden in einem menschenlesbaren Format in /etc/dumpdates abgelegt, wenn man die Option `-u` übergibt und Dump auf eine Partition ansetzt, nicht aber auf einzelne Dateien oder Verzeichnisse.

Da Dump für gewöhnlich dazu eingesetzt wird, komplette Partitionen zu sichern, verfügt es über keine Ein-/Ausschlusslisten, wie dies einige andere Programm benutzen. Stattdessen prüft Dump, ob das Fileflag NODUMP gesetzt ist. Mit diesem Flag kann man einzelne Dateien oder Verzeichnisse vom Dump ausschließen. Allerdings muss man hierbei beachten, daß Dump mit Level 0 generell Fileflags ignoriert. Möchte man trotzdem markierte Dateien nicht sichern, muss man `-h0` übergeben.

Dump lässt sich auf Dateisysteme und Partitionen ansetzen, dies geschieht am einfachsten mit Listing 16.2, die Option `-a` veranlasst dabei die Grösse des Archivs selbst festzulegen, anstatt nur Dateien voreingestellter Größe zu schreiben.

[1] allerdings bleibt dump dabei in der angegebenen Partition, folgt also Mountpoints nicht.

[!h]

```
$ chflags nodump .fetchmailrc .signature mail/ .gnupg/
$ ls -lo .signature
-rw-r--r--  1 stefan  stefan  nodump 369 Feb 18 21:09 .signature
$ chflags dump mail/
```

Listing 16.1: NODUMP setzen

[!h]

```
# dump -0u -f dump-02-31-12 -a /home
# dump -0u -f dump-02-31-12 -a /dev/wd0e
```

Listing 16.2: Sicherung mit dump

Dump kann mit der Option `-W` auch eine Übersicht der bisher erstellten Dumps aus /etc/dumpdates erstellen und mit `-w` die noch zu sichernden Partitionen angeben. Möchte man Archive bestimmter Größe erzeugen (bspw. für Streamer oder um sie auf CD/DVD zu brennen), verfügt dump über einige, teils historische, Optionen. Von Interesse dürfte heute aber nur noch `-B` sein, das die Größe der zu sichernden Archive in Kilobyte erwartet. Dazu muss aber zwingend der Name eines jeden zu erzeugenden Archivs übergeben werden, da dump sonst automatisch nacheinander in die zuletzt[2] angegebene Datei schreibt.

ṣelber hat keine Möglichkeit Archive zu komprimieren, dies kann aber mittels Pipe an Bzip2 oder Gzip bzw. einem kleinen Shellskript gelöst werden. Möglich ist ebenfalls der Einsatz im Netzwerk mittels rdump(8) und rrestore(8), dies ist aber aufgrund mangelnder Sicherheit bzw. fehlender Verschlüsselung nur in gesicherten Netzwerken möglich, so das ein SSH-Tunnel mit der Umgebungsvariable `RCMD_CMD` aufgerufen werden sollte. Besser ist hier eine Pipe an ssh, der die Ausgabe auf einen Rechner im Netzwerk schreibt, wie in Listing 16.4 illustriert.

16.1.1 *restore*

Restore kann ein komplettes Archiv mit der Option `-r` zurückspielen (siehe Listing 16.5). Dabei erzeugt es eine Datei namens »restoresymtable«, in der die Metainformationen zur Rücksicherung (Inodes, welche Bänder, etc.) abgelegt werden. Diese Datei kann nach der Rücksicherung *aller* Bänder entfernt werden.

Möchte man nur bestimmte Pfade oder Dateien zurückspielen, kann man Teile des Verzeichnisses mit der Option `-x` übergeben. Leider unterstützt restore keine RegExes, aber man kann sich hierbei mit ein paar Backticks behelfen. Um zu prüfen ob ein

[2] werden drei Dateinamen angegeben, aber fünf Dateien erzeugt, werden die dritte, vierte und fünfte in die dritte Datei geschrieben, man hat dann also drei Dateien mit der ersten, zweiten und fünften Sicherung vorliegen

[!h]

```
#!/bin/sh

DATUM=‘date +%y%m%d‘
QUELLE=/home/

# Mediengröße in kB, 102000=>99MB,
# 715000=>698MB, 2097152000=>2000MB
MEDIUM=102000

#Größe des Verzeichnis in kB
MEDIEN=‘du -sk $QUELLE | awk ’{print $1}’‘

# Medienzahl kalkulieren, Nachkomma wird abgeschnitten!
MEDIEN=$(($MEDIEN/$MEDIUM))

# Medienzahl aufrunden
MEDIEN=$(($MEDIEN+1))

# Dateinamen iterieren
DAT=${DATUM}_1
i=1
while [ $i -lt $MEDIEN ]
do
i=‘expr $i + 1‘
DATEIEN=${DATUM}_$i
DAT=${DAT},${DATEIEN}
done

echo "dump -0 -B $MEDIUM -f $DAT $QUELLE"
```

Listing 16.3: dump-Befehl für mehrere Volumes

[!h]

```
# dump -0a -f - home | ssh operator@192.168.1.1 dd of=/usr/backup/dump.0
```

Listing 16.4: Dump im Netzwerk

[!h]

```
# restore -r -f 02-11-01 && restore -r -f 02-11-08
# rm restoresymtable
```

Listing 16.5: Zwei Dumps mit restore zurückspielen

Archiv erfolgreich geschrieben wurde, reicht es zu Testzwecken die allerletzte Datei zu restaurieren. Da dump alle vorigen Einträge zumindest traversieren muss, werden Fehler im Archiv erkannt und gemeldet. In der Dateiliste (`-t`) werden die Inodes und die korrespondierende Dateiname angegeben, man kann also den letzten Eintrag zurücksichern.

[!h]

```
$ restore -x stefan/ www/
$ restore -t > dateiliste
$ restore -x ‘grep -i ’bz2$’ dateiliste | awk ’{print $2}’‘
$ restore -x ‘tail -n 1 dateiliste | awk ’{print $2}’‘
```

Listing 16.6: einzelne Dateien mit restore zurückspielen

Sucht man eine bestimmte Datei im gesicherten Verzeichnisbaum, kann man restore -i verwenden. Hier wird eine Shell aufgerufen, in der man mit ls und cd den Verzeichnisbaum traversieren kann. Hat mann die gewünschten Dateien gefunden, kann man sie mit add zur Liste der zurückzusichernden Dateien hinzufügen. Mit extract werden die in der Liste spezifierten Dateien im aktuellen Verzeichnis entpackt.

Um einen unterbrochenen Lauf fortzusetzen, kann man die Option `-R` verwenden.

16.1.2 *Dump im Detail*

Da das wichtigste Sicherungsprogramm ist, werde ich es näher untersuchen:

Das exemplarisch abgedruckte Log eines Sicherungslaufes gibt wertvolle Informationen wieder:

- das zu sichernde Verzeichnis /home ist als eigene Slice /dev/rwd1a in /etc/fstab aufgeführt
- Dieser Sicherungslauf (Level 1) beginnt 12:17 am 24.12.2005
- Der letzte Sicherungslauf im Level 0 fand am 18.12. statt
- Der dump wird durchgeführt und erfolgreich beendet, dazu wird eine Statistik (Dauer, Platz, Übertragungsrate) ausgegeben.

Interessant sind hierbei insbesondere die einzelnen Durchläufe (*Pass I - IV*):

1. aus dem aktuellen Datum, dem angegebenen Level (hier: 1) und den letzten relevanten Durchläufen aufgezeichnet in /etc/dumpdates berechnet dump die Variable `DUMP_SINCE`. Alle Inodes im Dateisystem werden traversiert und deren *modification time (mtime)* wird mit `DUMP_SINCE` bezgl. Größe verglichen. Ist die *mtime* einer Datei größer oder gleich `DUMP_SINCE`, wird der Inode in die Liste zu sichernder Dateien übernommen. Nichtzugeordnete Inodes werden ignoriert, so das dump am Ende des ersten Durchlaufes drei Listen erzeugt hat:

[!h]

```
root@# dump -1au -h0  -f - /home |gzip > home.1
  DUMP: Found /dev/rwd1a on /home in /etc/fstab
  DUMP: Date of this level 1 dump: Sat Dec 24 12:17:56 2005
  DUMP: Date of last level 0 dump: Sun Dec 18 23:20:19 2005
  DUMP: Dumping /dev/rwd1a (/home) to standard output
  DUMP: Label: none
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 41438 tape blocks.
  DUMP: Volume 1 started at: Sat Dec 24 12:18:18 2005
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: 40541 tape blocks
  DUMP: Volume 1 completed at: Sat Dec 24 12:18:40 2005
  DUMP: Volume 1 took 0:00:22
  DUMP: Volume 1 transfer rate: 1842 KB/s
  DUMP: Date of this level 1 dump: Sat Dec 24 12:17:56 2005
  DUMP: Date this dump completed:  Sat Dec 24 12:18:40 2005
  DUMP: Average transfer rate: 1842 KB/s
  DUMP: level 1 dump on Sat Dec 24 12:17:56 2005
  DUMP: DUMP IS DONE
root@#
```

Listing 16.7: Protokoll einer Dump-Sicherung

   a) Inodes der Dateien, die gesichert werden sollen
   b) Inodes aller Verzeichnisse
   c) Zugeordnete Inodes

2. Durchlauf II läuft in mehreren Stufen ab:
   a) Die in Durchlauf 1 erstellte Liste der Verzeichnisse wird erneut traversiert und geprüft ob zu sichernde Dateien im Verzeichnis existieren. Wenn nicht, wird der entsprechende Inode aus der Liste zu sichernder Verzeichnisse entfernt.
   b) Die Verzeichnisse werden erneut überprüft, in dem der Verzeichnisbaum von den Blättern zur Wurzel traversiert wird. Dabei sollen evtl. im letzten Durchlauf freigewordene Verzeichnisse gefunden werden.
   Es wurden in diesem Lauf keine weiteren Verzeichnisse freigemacht, da sonst ein weiterer *Pass II* stattgefunden hätte.
3. Vorbereitung von Pass III:
   Bevor dump mit dem Schreiben der Daten beginnt, werden deren Metainformationen abgelegt. Hierzu wird ein *Header* geschrieben, der die nachfolgenden Daten beschreibt. Zusätzlich werden zwei Inode-Tabellen geschrieben, eine im nativen Format und eine, aus Kompatibilitätsgründen, im normalisierten alten BSD-Format.

4. Ein Header, der den Inode des Verzeichnisses enthält, und die Datenblöcke für jedes Verzeichnis in der zu-sichernde-Verzeichnisse-Liste werden geschrieben.
5. Ein Header, der den Inode der Datei enthält, und die Datenblöcke für jede Datei in der zu-sichernde-Dateien-Liste werden geschrieben.
6. Nachbereitung von Pass IV :
   Der abschließende Header wird aufs Band geschrieben.

Inkonsistenzen in der Sicherung können nur auftreten, wenn das Quelllaufwerk während der Sicherung beschrieben wird. Dabei ist es möglich, das einzelne Dateien nicht, oder nur falsch, gesichert werden. Eine Korrumpierung der *gesamten* Sicherung ist dabei aber äußerst unwahrscheinlich.

16.2 tar, star, cpio, afio und pax

Tar(1) ist das wohl bekannteste Sicherungsprogramm und vielseitig einsetzbar. Allerdings gibt es verschiedene Versionen und Implementierungen von tar, so das man darauf achten sollte nur entsprechend kompatible Versionen einzusetzen bzw. bzw kompatible Archive zu erzeugen.
Tar steht für Tape Archiver, es wurde also dazu entwickelt Bandarchive zu beschreiben. Es erzeugt dazu eine Datei, die alle zu sichernden Dateien enthält und schreibt diese auf ein Band – oder in eine einfache Datei.

[!h]

```
$ tar cpfz operator@backup:/home/backup/backup.tgz /home/
$ tar xpfz backup.tgz
```

Listing 16.8: Dateien mit tar archivieren und entpacken

Die Option `-f` erzeugt eine Datei, standardmäßig schreibt das Archiv sonst auf das erste Bandlaufwerk /dev/nrsto.
Zurückspielen kann man ein Backup mit `x`, die Option `p` sorgt dafür, daß die Dateirechte beibehalten werden.

Tar schreibt relativ zum jetzigen Pfad, d. h. das gesicherte /home/-Verzeichnis wird an aktueller Stelle entpackt. Tar kann mit der Option `-z` gzip- bzw. mit `-j` bzip2-komprimierte Archive erzeugen. Entpackt werden Archive mit `-x`.

Es sei hierbei nicht verschwiegen das Tar grosse Probleme mit defekten Archiven hat, da bei einem Archiv ein Header erzeugt wird der Infomationen über das gesamte Archiv beinhaltet. Taucht jetzt an einer Stelle ein Fehler auf, entpackt Tar das Archiv nur bis zu dieser Fehlerstelle. Star ist eine von Jörg Schilling seit 1982 entwickelte Tar-Implementierung. Es ist die weltweit schnellste Tar-Variante und weist einige sehr nützliche Erweiterungen gegenüber Tar auf. Es unterstützt unter anderem Reguläre Ausdrücke, Levels a la Dump, keine Begrenzung der Pfadlänge, automatische

Erkennung der Endianess oder die Möglichkeit ein Dateisystem mit einem Tarball zu vergleichen. In der Standardvariante kann Star Tar-Archive lesen und umgekehrt. Star lässt sich aus pkgsrc/archivers/star installieren und sollte im Allgemeinen dem normalen Tar vorgezogen werden. Star unterstützt verschiedene Tar-Formate, die mit der Option -H= bestimmt werden können. Am leistungsfähigsten ist EXUSTAR. Cpio (copy in/out) ist von der Syntax her etwas eigen, da es als erste Option die Übergabe einer zu sichernden Datei erwartet und diese auf STDOUT schreibt. Dieses »Problem« erweist sich allerdings mithilfe der Pipe als Vorteil, da so mit find(1) oder ls(1) die Dateiliste übergeben und die Ausgabe mit > bestimmt werden kann.

[!h]

```
$ ls *.jpg | cpio -0 > /home/Bilder.cpio
$ find /home -mtime 7 | cpio -o > /dev/nrst0
$ cpio -i < /dev/nrst0
```

Listing 16.9: Dateien mit cpio sichern

Listing 16.9 zeigt wie man alle JPEGs im aktuellen Verzeichnis in die Datei ./Bilder.cpio sichert. Die Ausgabe kann stattdessen auch auf das Bandlaufwerk /dev/nrsto umgeleitet werden. Um Unterverzeichnisse zu sichern oder inkrementelle Backups zu erzeugen nutzt man find(1). Der zweite Befehl sichert alle Dateien die in den letzten sieben Tagen geändert wurden auf das erste Bandlaufwerk. Zurückspielen kann man die Sicherung mit dem dritten Befehl.

Leider unterstützt Cpio keine Überprüfung der Archive, so daß man dies auf Wunsch manuell machen muss, indem man das Archiv wieder entpackt und die Dateien mit diff(1) oder mtree(8) vergleicht. Afio ist eine Neuauflage von Cpio und behebt einige Probleme, z. B. Probleme mit leeren Dateien oder auf 0 gesetzte Rechte. afio verwendet dabei die selbe Syntax wie Cpio afio ist via pkgsrc/archivers/afio installierbar und sollte Cpio vorgezogen werden. Pax ist ein Standardisierungsversuch der IETF, da es zu viele Implementierungen von Tar und Cpio gibt. Pax ist in der Lage verschiedene Archivformate wie Cpio und Tar zu schreiben und zu lesen und kann so gerade in heterogenen Umgebungen nützlich sein. pax wird unter anderem auch auf den Installationsmedien von NetBSD verwendet.

[!h]

```
# pax -w -f /dev/nrst0 /home
# pax -v -f /dev/nrst0
$ pax -r -s ',^/*usr/*»' -f a.pax
```

Listing 16.10: Sicherung und Rücksicherung mit pax

Befehl Nummer eins in Listing 16.10 schreibt das /home/-Verzeichnis auf Band und zweitens gibt ein Inhaltsverzeichnis der Sicherung aus. Der letzte Befehl extrahiert

alle Dateien in /usr/ im Archiv ./a.pax

16.3 dd(1)

Dd(1) ist kein eigentliches Sicherungsprogramm, sondern es dient dazu Dateien zu konvertieren. Dd kann jedoch auch auf Raw Devices schreiben und von ihnen lesen, dies ermöglicht es, echte 1:1-Kopien von Geräten zu erstellen und zu schreiben, so kann man z. B. mit den Befehlen in Listing 16.11 ein Image einer Floppydiskette erstellen, die Partition /dev/wd1a 1:1 auf /dev/wd2a kopieren oder den MBR der ersten Platte sichern. Somit eignet es sich hervorragend um an Rohdaten, bspw. von Magnetbändern, heranzukommen oder komplette Systemplatten zu sichern. Da dd die gesamten Blöcke einer Partition einliest, wird das erzeugte Image genau so groß wie die Partition werden. Möchte man das Image komprimieren, sollte man vorher im laufenden System den leeren Festplattenplatz mit Nullen überschreiben.

Dd ist ebenfalls in der Lage Dateien zu konvertieren, so kann man mit der Option `conv=swab` die Byte-Order vertauschen lassen.

[!h]

```
# dd if=/dev/zero of=/nullen && rm /nullen
# dd if=/dev/fd0a of=floppy.iso
# dd if=/dev/wd1a of=/dev/wd2a bs=2m | split -b 695M
# dd if=/dev/wd0d of=mbr.dd count=1
```

Listing 16.11: Daten mit dd schreiben

16.4 ghost for unix (g4u)

Ghost for Unix[3] (g4u) ist eine NetBSD-Diskettenvariante, die von Hubert Feyrer entwickelt wurde um Festplatten zu spiegeln und im Netz via FTP zu verteilen. Es verwendet dd (siehe Kap. 16.3) um auf Platten zuzugreifen und kann daher jedes Datei-/Betriebssystem spiegeln. Es verwendet zum Betrieb einige Shellskripte, so das es auch ein ungeübter NetBSD-Anwender einfach anwenden kann. Es eignet sich insbesondere hervorragend um mehrere gleich aufgebaute Systeme zu verteilen, so dass man bspw. in Laboren, Klassenräumen oder Clustern (siehe Regensburger Marathon-Cluster[4]) ein System nur einmal aufsetzt und dieses dann auf mehrere Rechner verteilt.

Es wird lediglich das Disketten- oder CD-Image benötigt um ein bootbares Medium zu erstellen. Davon wird g4u gebootet und anschließend können Festplatten und/oder Partitionen gesichert und restauriert werden.

[3] http://www.feyrer.de/g4u/
[4] http://www.feyrer.de/marathon-cluster/

[!h]

```
# uploaddisk benutzer@ftp.server.local Imagename.gz wd1
# slurpdisk benutzer@ftp.server.local Imagename.gz wd0
# copydisk sd0 sd1
```

Listing 16.12: Festplatten-Images mit g4u erstellen

In Listing 16.12 wird die zweite IDE-Platte (wd1) als ./Imagename.gz auf ftp.server.local gesichert. Der zweite Befehl installiert das Image auf der ersten IDE-Platte, der dritte Befehl kopiert lokal die erste SCSI auf die zweite SCSI-Platte.

G4u ist ein ausgezeichnetes System um komplette Platten zu sichern und zu verteilen. Für Datenbackups eignet es sich nur bedingt, da immer komplette Platten oder Partitionen mit dd gesichert werden.

Auf der Homepage des Systems findet sich eine ausführliche Anleitung.

16.5 AMANDA

Amanda steht für »Advanced Maryland Network Disk Archiver« und eignet sich, wie der Name schon sagt, hervorragend zur Sicherung in Netzen. Es verwendet dazu ein Client/Server-Modell um Unix-Clients und SMB-Shares[5] auf Bandlaufwerk zu sichern. Inzwischen verfügt es auch über die Fähigkeit die Sicherung auf Platten abzulegen oder Platten als Spoollaufwerk einzusetzen. Zur Sicherung eines Unixclients kann es GNU Tar oder das native dump des Clients verwenden. Um Daten von MS-Windows Rechnern zu sichern, müssen diese als Freigabe ins Netzwerk exportiert werden, um dann vom Server aus mit smbtar kopiert werden zu können. Für Unixclients kann Amanda auf den Clients selbst die Daten vor der Übertragung komprimieren oder diese nach der Übertragung auf dem Server komprimieren. Je nachdem ob die Auslastung der Clients oder die des Netzes wichtiger ist, kann man eine passende Strategie wählen. Da Amanda ein eigenes Netzprotokoll zur Übertragung der Daten verwendet, kann es an verschiedenen Punkten Daten abgreifen oder verändern (bspw. komprimieren, verschlüsseln oder auf Viren prüfen).

Die Zyklen der Sicherungsstrategie (also wann welches Level eingesetzt wird) berechnet Amanda selbständig aus den Statistiken der vorangegangenen Läufe, so das ein Optimum an Platzersparnis und Sicherungsaufwand errechnet wird. Bei manuellen Sicherungen lautete die Devise im allgemeinen »Freitags ein Komplettbackup, das archiviert wird.«. Mit Amandas automatischer Strategieberechnung lautet diese Devise: »Gib mir ein Komplettbackup in 7 Tagen.«. Durch diese Definition des *dumpcycle* verteilt Amanda die Sicherungen selbständig auf die Bänder – was dazu führt das den Logdateien erhebliche Bedeutung zukommt. Die Berechnung der Sicherungsstrategie erfolgt aus mehreren Daten, zum einen dem Dumpcycle, also wieviel Tage für einen

[5] Windows-Freigaben

Zyklus verwendet werden sollen. Ist ein Dumpcycle sieben Tage lang, werden pro Tag ein Siebentel der Daten gesichert. Zusätzlich sichert Amanda aber noch jene Daten inkrementell, die seit der ersten Sicherung verändert wurden.

Wie schon bereits im Kapitel 15.7 erwähnt, sind Festplatten heutzutage in der Regel hinreichend groß und auch sehr billig, so das der Einsatz einer Spoolingplatte als Zwischenspeicher wärmstens empfohlen werden kann. Sichert Amanda direkt auf Band und die Daten kommen nicht schnell genug heran, fängt ein Streamer an »Schuhe zu putzen«, also immer wieder vor- und zurückzuspulen. Das tut der Mechanik und dem Band nicht wirklich gut und vermindert den weiteren Durchsatz dramatisch. Amanda unterstützt eine Vielzahl an Bandlaufwerken und Wechseleinheiten, die über Skripte angesteuert werden können.

Seit geraumer Zeit verfügt Amanda über RAIT - dem Redundant Array of Inexpensives Tapes. Ähnlich dem RAID für Festplatten werden hier einfach mehrere Bänder eingesetzt und die Kapazität und Ausfallsicherheit zu erhöhen, indem man bsp. einfach fünf Bänder verwendet – vier für Daten und eines für eine XOR-Prüfsumme. So kann man Partitionen sichern, die viermal so groß sind wie ein einzelnes Band und zusätzlich noch die Integrität der Sicherung erhöhen. Da ein RAIT nicht einfach nacheinander die Sicherungen auf Bänder schreibt, sondern parallel, benötigt es in diesem Falle auch fünf Bandlaufwerke. Wenn das Sicherungssystem entsprechende Durchsatzraten erzeugt, erhöht dieses Verfahren auch den Datendurchsatz.

Allerdings kann Amanda einen dump-Lauf nicht auf mehrere Bänder verteilen. Ist der dump einer Partition größer als das Band, kann dump zum sichern dieser Partition nicht verwendet werden. Stattdessen kann man GNU Tar verwenden und mittels Include/Exclude-Liste[6] die Partition »stücken« und so auf mehrere Bänder verteilen.

16.5.1 *Einrichtung des Servers*

Amanda wird komplett per Konsole bzw. per dotconf-Dateien konfiguriert. Nach der Installation aus pkgsrc/misc/amanda für den Server werden in /usr/pkg/etc/amanda die Standard-Konfigurationsdateien angelegt. Amanda erwartet nun für jeden Konfigurationssatz ein eigenes Verzeichnis der Art /usr/pkg/etc/amanda/*konfigname*.

in jeder Konfiguration erwartet Amanda einige Dateien:

.AMANDAHOSTS Konfiguriert den Netzwerkzugriff ähnlich /etc/hosts

AMANDA.CONF Allgemeine Konfiguration, Geräte, Zyklus etc.

DISKLIST zu sichernde Platten/Verzeichnisse (DLE = Disk List Entry)

TAPELIST Liste der Bänder, von Amanda selbst verwaltet

/ETC/AMANDATES Amandas Äquivalent zu /etc/dumpdates, sollte mit touch(1) angelegt werden

[6] http://www.amanda.org/docs/topten.html#id2552399

/ETC/INETD.CONF Amandas Dienste müssen über den inetd gestartet werden

Die Datei amanda.conf enthält folgende Parameter:

ORG Betreffzeile für Berichtmails

MAILTO Empfänger der Berichtmails

DUMPUSER Benutzer der die Dumps durchführt. Standard ist *[*backup]

INPARALLEL Maximale Anzahl parallell laufender Dumps. (0-63)

DUMPORDER Priorisierungskriterium der Dumpreihenfolge. Möglich sind Größe, Bandbreite und Zeit, jeweils auf- oder absteigend.

NETUSAGE Maximal belegbare Bandbreite.

DUMPCYCLE Anzahl der Tage/Wochen für einen Sicherungszyklus.

RUNSPERCYCLE Durchläufe pro Zyklus, zum Beispiel (Wochen des dumpcycle) mal Werktage

TAPECYCLE Anzahl der Bänder Amanda benötigt, entspricht runspercycle + Sicherheitszuschlag

RUNTAPES Bänder je einzelnem Amandalauf

TPCHANGER Skript für den Bandwechsler

TAPEDEV Bandlaufwerk oder Verzeichnis, wenn auf Platte gesichert werden soll (z.B: file:/mnt/amanda)

TAPETYPE Bandtyp, z. B. HP-DAT, DLT oder harddisk. Beispiele sind bereits in der amanda.conf enthalten.

RESERVE Reserviert n% der Spoolplatte für inkrementelle Backups

LABELSTR Ein RegEx der das Label für die Bänder festlegt.

Wie bereits erläutert, sollte Amanda unbedingt mit einer Spoolplatte betrieben werden. Diese kann ebenfalls in der amanda.conf definiert werden. Ist eine Spoolplatte konfiguriert, benutzt Amanda diese um erst Clientdaten zu sammeln und dann von der Platte auf Band zu schreiben. Definiert man mehrere Spoolplatten, benutzt Amanda diese nach eigenem Gutdünken. Ist ein dump zu groß für die Spoolplatte, wird er direkt aufs Band geschrieben. Wenn ein Band nicht beschrieben werden kann, weil es bspw. defekt oder schreibgeschützt ist, wird ein sogenannter *degraded mode* erzeugt, die Sicherung also nur auf die Spoolplatte geschrieben und nicht auf das Band. Hat man den Bandfehler behoben, kann man mit amflush die Spooldatei auf ein Band schreiben. Eine Spoolplatte wird beispielhaft in Listing 16.13 definiert.

Die Konfiguration der Sicherungsparameter findet in den sogenannten *dumptypes* (Listing 16.14) statt. Hier wird bspw. festegelegt ob beim Sicherungslauf komprimiert werden soll

Die eigentlichen zu sichernden Pfade legt man nun in der ./disklist (Listing 16.15) an:

Zu guter Letzt muss man noch die Amandadienste via /etc/inetd.conf (Listing 16.16) starten

[!h]

```
holdingdisk hd1 {  # Name der Platte
 comment "erste SCSI als Spoolplatte"
 directory "/mnt/sd0a/" # Mountpoint der Platte
 use 5500 MB    # Maximal verfügbarer Platz
 #use -100 MB   # Nutze gesamte Platte bis auf 100MB
 chunksize 1Gb  # Größe einzelner Dateien, wenn der dump
                # auf mehrere Dateien verteilt werden soll
```

Listing 16.13: Spoolplatten für Amanda definieren

[!h]

```
define dumptype always-full { # Name der Konf.
    global  # dumptype global einbinden
    comment "Full dump of this filesystem always"
    compress none
    priority high
    dumpcycle 0
}

define dumptype stantar {
    global # dumptype "global" einbinden
    program "GNUTAR" #gtar verwenden
    comment "standardtar"
    compress client fast # auf Client schnell komprimieren
holdingdisk yes # Spoolplatte verwenden
index  # Inhaltsverzeichnis erstellen
priority # high
dumpcycle 7 # eine Woche Zyklus
    exclude list "/mnt/amanda.exclude"
}

define dumptype root-tar {
    global
    program "GNUTAR"
    comment "root partitions dumped with tar"
    compress none
    index
    exclude list "/usr/amanda/exclude.gtar"
    priority low
}
```

Listing 16.14: dumptypes in amanda.conf definieren

[!h]

```
balmung.net-tex.de /mnt/source stantar
balmung.net-tex.de /home always-full
kusanagi.net-tex.de /home always-full
```

Listing 16.15: Zu sichernden Pfade in der disklist festlegen

[!h]

```
amanda    dgram  udp wait   amanda /usr/pkg/libexec/amandad    amandad
amandaidx stream tcp nowait amanda /usr/pkg/libexec/amindexd   amindexd
amidxtape stream tcp nowait amanda /usr/pkg/libexec/amidxtaped amidxtaped
```

Listing 16.16: Amanda-Server-Dienste starten

16.5.2 *Einrichtung der Clients*

Auf den Clients muss das Amanda-client Paket (pkgsrc/sysutils/amanda-client) installiert werden. Anschließend wird in der /etc/inetd.conf (Listing 16.17) der Amanda-Server gestartet und in der /usr/pkg/etc/amanda/.amandahosts (Listing 16.18) die Zugriffsrechte ähnlich einer .rhosts vergeben.

[!h]

```
amanda dgram udp wait amanda /usr/pkg/libexec/amandad amandad
```

Listing 16.17: Amanda-Client-Dienste starten

16.5.3 *Programme im Amanda-Paket*

Amanda verfügt über mehrere Programme, die mit dem Paket installiert werden:

AMADMIN Verwaltungsprogramm für die Amandadaten. Kann bspw. Bänder freigeben, löschen oder suchen und die Datenbank im-/exportieren.

AMCHECK Überprüft die Konfigurationsdateien auf Fehler. Sollte zu den Bürozeiten durchgeführt werden.

AMCHECKDB Überprüft die Banddatenbank auf Konsistenz, also ob alle aufgeführten Bänder in der Bandliste sind.

AMCLEANUP Bereinigt den Server nach einem Absturz. Normalerweise automatisch ausgeführt wird es nur nach einem Systemfehler benötigt.

AMDD Amandas dd.

[!h]

```
localhost backup
localhost root
```

Listing 16.18: Zugriffsrechte auf dem Client konfigurieren

AMDUMP Startet den Sicherungslauf mit allen DLEs einer Konfiguration. Das Programm für den nächtlichen Cronlauf.

AMFLUSH Schreibt dumps von der Spoolplatte auf Band.

AMGETCONF Gibt Parameter der Konfigurationsdateien wieder.

AMLABEL Labelt ein Band. Amanda überschreibt kein gelabeltes Band und benutzt keines ohne Label.

AMMT Amandas mt.

AMOVERVIEW Listet eine Tabelle auf in der angezeigt wird wann welche Partition mit welchem Level gesichert wurde. (Ähnlich dump -W)

AMRECOVER Durchsucht die Amanda Indizes um wiederherzustellende Dateien auszuwählen. Dateien können bspw. nach Datum gesucht werden.

AMRESTORE Spielt ein Backup vom Band zurück, so das es native zurückgesichert werden kann.

AMRMTAPE Entfernt ein existierendes Band aus der Datenbank. Nur notwendig wenn das Band zerstört wurde.

AMSTATUS Zustand der laufenden Sicherung.

AMTAPE Zugriff auf Bandwechsler, bspw. Band auswerfen oder das Laufwerk reinigen.

AMTOC Perlskript das ein Inhaltsverzeichnis aus einer Logdatei erzeugt. Kann idealerweise mit amdump in einem Cronjob kombiniert werden.

AMVERIFY Überprüft ein Band auf Fehler.

AMVERIFYRUN Überprüft alle Bänder des letzten Laufs auf Fehler.

16.5.4 *Praktischer Einsatz*

Zuerst sollte man mit amcheck(8) die Konfiguration auf Fehler überprüfen. Man kann dies auch automatisiert per Cron zu den üblichen Bürozeiten ausführen, so das rechtzeitig auftretende Probleme erkannt werden können. Anschließend sollten alle Bänder, die verwendet werden soll, gelabelt werden. Dies kann man mit amlabel(8) tun, indem man es ggf. in ein Shellskript einbindet (nützlich für Bandwechsler oder wenn man auf Festplatte sichert).

Mit amdump(8) kann man vorerst von Hand den Sicherungslauf anstoßen und die Benachrichtigungsemail studieren. Nach einem erfolgreichen Sicherungslauf sollte das Band mit amverify(8) oder amverifyrun(8) überprüft werden. Mit amtoc(8) kann man nun aus dem Inhaltsverzeichnis des Sicherungsbandes eine menschenlesbare ASCII-Datei machen, was recht nützlich ist, falls man nicht auf den Katalog zugreifen kann.

Hat das System von Hand funktioniert, sollte man es in ein Shellskript gießen (dokumentieren und sichern nicht vergessen) und von Cron aufrufen lassen.

16.5.4.1 *Rücksicherung*

Einzelne Dateien, Verzeichnisstrukturen oder ganze Platten lassen sich mit amrestore zurückspielen. Als Optionen erwartet amrestore optional mehrere Hostnamen, Plattennamen und Zeitstempel. Wird eine Option nicht gesetzt, werden alle Dateien getroffen und zurückgespielt, übergibt man also gar keine der Optionen, werden alle verfügbaren Sicherungen zurückgespielt.

Hat man den Amanda-Index nicht mehr zur Verfügung, kann man Bänder auch mit amrecover auslesen. Normalerweise spielt amrecover die Dateien der Sicherungen nicht zurück an ihren alten Platz, sondern spielt nur die Archivdateien von Band auf Festplatte. Möchte man die Archivdateien gleich entpacken und die Sicherungen so zurückspielen, übrgibt die Option -p (wie Pipe) und in einer Pipe das entsprechende Restoreprogramm, also GNU Tar oder restore für dump. Neben -p kann man auch -h angeben, um die Header des Bandes auslesen zu können, oder -r um die Archive im Rohformat zu Debuggingzwecken vom Band lesen zu können.

Hat man nicht einmal mehr amrecover zur Verfügung, ist dies wohl der GAU der Datensicherung. Solange die Bänder aber unbeschädigt sind, kann man die Images auch mit dd(1) vom Band herunterziehen. Dabei hilft Amandas Headerdatei (Listing 16.19) am Anfang des Bandes, denn dort wird im Klartext angezeigt, wie man mit dd(1) und GNU Tar oder restore(8) die Daten rücksichert.

[!h]

```
# mt rewind
# dd if=/dev/nrst0 bs=32k count=1
```

Listing 16.19: Amanda Header vom Band restaurieren

SIEHE AUCH:

Teil v

EINBRUCHSERKENNUNG IN NETZWERKE

17

EINBRUCHSERKENNUNG IN NETZWERKE

Das Wissen muss ein Können werden.

(Generalmajor Carl von Clausewitz)

17.1 INTRUSION DETECTION SYSTEME

17.1.1 *Wozu Einbruchserkennung?*

Das Gebiet der Einbruchserkennung ergänzt als Teil des Sicherheitskonzeptes andere technische Sicherheitsstrategien wie Zugangsschutz, Identifikation, Autorisierung, Datensicherung und soziale Maßnahmen (Vorschriften, Richtlinien, Verträge ...). Die Einbruchserkennung ist notwendig bzw. sinnvoll, da in der Regel die o.g. Maßnahmen die Sicherheit als Prozeß nicht komplett gewährleisten können bzw. nicht alle Angriffsszenarien abdecken oder gar selbst angegriffen und ausgehebelt werden können.

17.1.2 *Was ist/kann IDS nicht?*

Auch ein IDS bietet keine 100%-ige Sicherheit, da Sicherheit nun mal kein Zustand ist, der durch Installation eines Programms erreicht wird, sondern ein Prozeß mit vielen veränderlichen Variablen.

Außerdem kann selbst ein installiertes IDS Angriffziel werden und den Zugang zu weiteren Systemen öffnen.

Idealerweise ist ein IDS in der Lage auch unbekannte neuartige Einbruchsszenarien zu erkennen und abzuwehren, aber es dürfte jedem klar sein das dies nur ein Wunschtraum bleiben wird und ein IDS ständige Pflege und Überwachung benötigt. Außerdem ist ein tiefes Verständniss der zugrundeliegenden Protokolle und Anwendungen notwendig. Durch die steigende Komplexität der IT-Infrastrukturen und Sicherheitsmaßnahmen ist in größeren Netzen eine koordinierte Arbeit mehrerer Administratoren notwendig.

17.2 Prinzipien der Einbruchserkennung

Ein System zur Einbruchserkennung[1] besteht im Prinzip aus drei Komponenten, die die jeweilige Funktionalität zur Verfügung stellen:

Datensammlung
: Das System muss die anfallenden Daten sammeln und zur Analyse aufbereiten

Datenanalyse
: Die angesammelten Daten müssen bezgl. Angriffe untersucht werden

Datenpräsentation
: Bei entsprechenden Treffern ist eine Information der Administratoren notwendig.

Zusätzlich existieren noch sogenannte Intrusion Prevention Systeme[2], die anstatt der, oder zusätzlich zur, Präsentation Reaktionen starten um den Angriff abzuwehren. Diese System gleichen den IDS in den wichtigsten Teilen und sind lediglich um eine recht »simple«[3] Reaktionslogik erweitert, daher werde ich nicht gesondert darauf eingehen.

Weiterhin lassen sich IDS in weitere Klassen unterteilen, bspw. bezgl. der zeitlichen Reaktion in reaktive, also solche die nach einem Einbruch reagieren und proaktive, die während eines Einbruchs reagieren und versuchen diesen zu unterbinden. Außerdem existieren noch Postmortem Werkzeuge, die eine Analyse »toter« Systeme ermöglicht.

Außerdem kann man die Verteilung bzw. den Aktionsort der IDS einteilen in hostbasierte IDS (HIDS), die nur auf einem System arbeiten (bspw. Virenscanner) und netzwerkbasierte (NIDS) die auf verteilten Systemen in Netzwerken arbeiten oder in Hybridsystem, die Eigenschaften der NIDS und HIDS vereinen.

17.2.1 *Datensammlung*

Die Datensammlung ist ein technisches und ein soziales Problem, da ein IDS idealerweise alle anfallenden Daten sammeln und analysieren sollte, dabei aber an juristische und moralische Grenzen bezgl. des Datenschutzes stößt.

Nur weil ich alle eMails der Nutzer sammeln *kann*, heißt dies nicht auch das ich das tun[4] *muss*. Ebenso sind potentielle Angriffe die ins Leere laufen, bspw. gegen nicht installierte CGIs auf Webservern oder bestimmte Netzdienste, die nicht angeboten werden uninteressant und können getrost ignoriert werden.

[1] Intrusion Detection System, IDS
[2] IPS
[3] wer schon mal 150 heterogene Netzwerkrechner verarzten musste, weiß wie simpel »simpel« ist ;-)
[4] vom technischen Sinn mal ganz abgesehen, Schwanzverlängerungsspam ist zwar nervig, aber i.d.R. nicht wirklich IDS-relevant

Laut (2000) steigt zwar die Zahl der Angriffe gegen IT-Systeme kontinuierlich an, dabei ist allerdings die schöpferische Höhe der Angriffe in den meisten Fällen eher gering. Insbesondere Angriffe mittels Skripte sind relativ leicht aufklärbar und können daher auch sehr leicht bekämpft werden. Prinzipiell ist es aber auch hier einfacher und sinnvoller die (vorhandenen) Angriffsvektoren auszumerzen.

Prinzipiell unterscheidet sich die Datensammlung bei HIDS und NIDS erheblich voneinander, da ein HIDS nur die relevanten Daten auf einem Host analysieren muss. Um beim Virenscanner als Beispiel zu bleiben, muss dieser auf seinem Host nur »gefährliche« Dateien scannen, also solche die schädlichen Code enthalten können. Durch die Anpassung an lediglich ein bekanntes System ermöglicht ein HIDS eine bessere Analyse der Vorgänge, ist aber bei Ausfall des Hosts ebenfalls inaktiv.

Das NIDS besteht aber in der Regel aus mehreren verteilten Monitoren im Netz und einer Managementkonsole, da Teile des Netzwerkverkehrs gesammelt werden müssen. Die Monitore arbeiten dabei bspw. auf verschiedenen Rechnern oder überwachen das Netz an strategischen Punkten (Gateway, DMZ, Switch) und werden von der Managementkonsole aus gesteuert bzw. wird von dieser aus die Datenanalyse und Datenpräsentation ausgeführt. Ein NIDS ermöglicht die Überwachung eines kompletten Netzes und erfordert dazu lediglich einige Monitore. Außerdem ist ein NIDS nicht von Angriffen/Ausfällen eines Zielsystems betroffen.

Problematisch ist hierbei eine geswitchte Umgebung, wenn die Switches bzw. die Infrastruktur nicht auf Überwachung (Monitoringports o.ä.) ausgelegt sind oder die Datenströme mittels Kryptographie gesichert werden. Zwar wäre es im zweiten Fall möglich mittels Key Escrow[5] den Verkehr vom IDS entschlüsseln und verarbeiten zu lassen, dies korrumpiert aber das gesamte Kryptosystem, da dessen Sicherheit nicht mehr gewährleistet werden kann. Möglich praktikabel ist es aber noch Schnittstellen beim Benutzer einzurichten, die bei dessem Zugriff auf die Chiffrate den Datenstrom analysieren. Hierbei sollte man genau abwägen ob der Einsatz des IDS die Kompromottierung des Kryptosystems rechtfertigt.

17.2.2 *Angriffserkennung*

Ein weites, und weitaus schwierigeres Feld als die Datensammlung ist die Datenanalyse. Auch hierbei existieren wieder verschiedene Vorgehensweisen:

INTEGRITÄTSPRÜFUNG
: Integrität der Daten wird geprüft

SIGNATURERKENNUNG
: bekannte Angriffssignaturen werden erkannt

ANOMALIEERKENNUNG
: vom Normalbetrieb abweichende Operationen werden erkannt

[5] Schlüsselhinterlegung

17.2.2.1 *Integritätsprüfung*

Vom technischen Standpunkt aus die einfachste Variante, Daten (i.d.R. Dateien) werden mit kryptographischen Signaturen oder Prüfsummen (PGP, SHA1) signiert und bei Bedarf gegen eine initiale Datenbank verglichen. Dies kann Postmortem oder reaktiv erfolgen, es wird also aufgeklärt welche Dateien verändert wurden, aber auch ein reaktives System ist möglich, bspw. `verifiedexec(4)` unter NetBSD, das bei jedem `exec()` im Kernel eine aktuelle Prüfsumme der Binärdatei mit einer alten Datenbank vergleicht und bei Fehlern die Ausführung der Datei verweigert.

Angriffe sind in der Regel nur gegen das kryptographische System (Passwort/Schlüssel) möglich.

17.2.2.2 *Signaturerkennung*

Ein System wie bspw. Snort analysiert den gesamten, oder zumindest relevante Teile des, Netzwerkverkehrs und überprüft den Verkehr auf Signaturen verschiedener Angriffe. Die Signatur enthält dabei bestimmte Auffälligkeiten oder Regelmäßigkeiten und Muster die einen Angriff kennzeichnen, z. B. bestimmte Pakettypen an bestimmte Ports oder markante Textstellen in Viren.

Angewendet werden in der Regel Pattern Matching Verfahren die den Verkehr analysieren, zusätzlich sind meist noch andere Verfahren nötig um die Daten vor der eigentlichen Analyse aufzubereiten, bspw. das Zusammensetzen fragmentierten Verkehrs.

Angriffe gegen diese Verfahren sind über Insertion und Evasion möglich.

17.2.2.3 *Anomalieerkennung*

Ein System der Anomalieerkennung überwacht das Verhalten der Benutzer, welches in einem Profil analysiert wird. Entsprechende Abweichungen vom Profil können als Angriff gewertet werden, verwendet also bspw. eine Sekretärin die sonst nur Webbrowser, Emailprogramm und Textverarbeitung von 08:00 bis 16:00 Uhr benutzt auf einmal um 23:00 Uhr `nmap` und `gcc`, deutet dies auf eine Kompromittierung des Benutzerkontos hin, höchstwahrscheinlich wurde also in das System eingebrochen.

Ebenfalls möglich ist die manchmal als Bottleneck bezeichnete Methode der Alarmierung bei privilegierten Operationen, die normalerweise nicht von Benutzer ausgeführt werden dürfen. Gelangt ein Benutzer also an Rechte die ihm nicht zustehen (bspw. `wheel`) und kann so Programme aufrufen die er sonst nicht benutzen durfte (bspw. `su(1)`) alarmiert das System den Administrator.

Problematisch ist hierbei vor allem die faschistoide Überwachung der Benutzer, man sollte also nur Server oder wirklich wichtige Spezialsysteme damit sichern. Außerdem ist es fraglich ob sich ein relativ eindeutiges Profil für jeden Benutzer erstellen lässt. Weiterhin ist eine Implementierung des IDS nicht trivial, da es auch gegen Angriffe durch `root` geschützt sein muss.

17.3 ANGRIFFE GEGEN IDS

Generell kann jedes IDS angegriffen werden, in dem seine Module attackiert werden und die »üblichen Verdächtigen« (Buffer Overflow, Format String, Privilege Escalation ...) ausgenutzt werden. Diese technischen Attacken gelten aber prinzipiell für jedes Programm und können mit bekannten Methoden (Privilege Separation, Kontrolle der Quellen) bekämpft werden.

Trotzdem gibt es noch Angriffe die sich gezielt gegen Methoden der IDS richten.

17.3.1 *Integritätsprüfung*

Die Sicherheit der Integritätsprüfung liegt im allgemeinen in der Sicherheit des verwendeten Kryptosystems, d.h. die verwendeten mathematischen Verfahren und deren Implementierung gelten als sicher.

Weiterhin ist ein ggf. verwendetes Passwort und/oder ein privater Schlüssel explizit zu schützen, d.h. man sollte den privaten Schlüssel nicht auf dem Produktivssystem belassen.

Bei Systemen die keine kryptographische Signatur sondern einfache Prüfsummen einsetzen liegt die Integrität des Systems in der Integrität der initialen Datenbank. Diese sollte hinreichend gesichert werden, in dem man sie bspw. ausdruckt, auf ein WORM Medium bringt oder mit einer kryptographischen Signatur explizit sichert.

17.3.2 *Signaturerkennung*

Aufgrund ihrer Natur und dem Einsatz in verteilten Systemen kann sich ein NIDS schenll zu einem Engpass in Systemen entwickeln und so bspw. auch über DDoS-Attacken ausgeschaltet oder umgangen werden.

Insertion und Evasion sind direkte Angriffe gegen die Signaturerkennungsmechanismen des IDS und versuchen hierbei die logische Reihenfolge des IDS auszuhebeln, in dem manipulierte Pakete in den Verkehr eingeschleust werden. Im Prinzip geht es also nur darum eine Angriffsignatur zu verschleiern, um zu verhindern daß das IDS sie erkennt.

17.3.2.1 *Insertion*

In der Insertion werden Pakete propagiert, die vom IDS, nicht aber vom Zielsystem akzeptiert werden. Das IDS akzeptiert also das manipulierte Paket, welches die Angriffssignatur tarnt und erkennt den Angriff somit nicht. Das Zielsystem allerdings dropt das manipulierte Paket und wird vom Angriff getötet.

Erläuterung der Insertionattacke an einem Beispiel:
Annahme:

- jedes Kästchen sei ein Paket
- das IDS akzeptiert Pakete mit einem *, das Zielsystem nicht
- die Pakete werden in der indizierten Reihenfolge vom IDS untersucht
- erhält das Zielsystem die Pakete H, A, L, T in genau dieser Reihenfolge stürzt es ab
- das IDS kann diesen Angriff verhindern

1.) Der Angreifer schleust folgende Pakete ins Netz ein:
$\boxed{H_1}\ \boxed{X_2^*}\ \boxed{A_3}\ \boxed{L_4}\ \boxed{T_5}$
2.) Das IDS untersucht und akzeptiert alle Pakete, liest also `H X A L T`
3.) es existiert keine Angriffssignatur zu `H X A L T`
4.) alle Pakete werden an das Zielsystem weitergeleitet
5.) das Zielsystem erhält alle Pakete, dropt aber X_2^*
6.) das Zielsystem erhält $\boxed{H_1}\ \boxed{A_3}\ \boxed{L_4}\ \boxed{T_5}$ und stirbt qualvoll

17.3.2.2 *Evasion*

Das umgekehrte Prinzip zur Insertion, das IDS dropt ein Paket, welches das Zielsystem akzeptiert, wird Evasion genannt.
Erläuterung der Evasionattacke an einem Beispiel:
Annahme:

- jedes Kästchen sei ein Paket
- das IDS akzeptiert Pakete mit einem * nicht, das Zielsystem schon
- die Pakete werden in der indizierten Reihenfolge vom IDS untersucht
- erhält das Zielsystem die Pakete H, A, L, T in genau dieser Reihenfolge stürzt es ab
- das IDS kann diesen Angriff verhindern

1.) Der Angreifer schleust folgende Pakete ins Netz ein:

$\boxed{H_1}\ \boxed{A_2^*}\ \boxed{L_3}\ \boxed{T_4}$
2.) Das IDS untersucht alle Pakete, dropt aber A_2^*, liest also `H L T`
3.) es existiert keine Angriffssignatur zu `H L T`
4.) alle Pakete werden an das Zielsystem weitergeleitet
5.) das Zielsystem erhält alle Pakete, akzeptiert aber A_2^*
6.) das Zielsystem erhält $\boxed{H_1}\ \boxed{A_2^*}\ \boxed{L_3}\ \boxed{T_4}$ und stirbt qualvoll

17.3.3 *Programme*

17.3.3.1 *AIDE*

AIDE ist ein reaktives, HIDS bzw. genauer gesagt ein Dateisystemintegritätsprüfer. AIDE erzeugt eine einfache Datenbank, in der Dateiname, Dateistatus und Prüfsumme über der Datei abgelegt werden. Sofern man diese initiale Datenbank entsprechend schützt (z. B. auf einem schreibgeschützten Medium oder verschlüsselt), kann man diese Datenbank erneut über das aktuelle System erzeugen lassen und mit der initialen Datenbank vergleichen. Weicht die Prüfsumme einer Datei ab wurde diese offensichtlich manipuliert.

Ziel des Systems ist es, insbesondere Rootkits zu enttarnen, diese Kits werden in der Regel nach einem erfolgreichen Einbruch installiert und dienen dazu die Anwesenheit eines Einbrechers zu verschleiern. Meist werden hierzu manipulierte Binaries wichtiger Systemprogramme installiert, um z. B. zu verhindern daß `ps -aux` die laufenden Prozeße des Einbrechers anzeigt. Nachteilig ist hierbei allerdings das AIDE reaktiv funktioniert, d. h. es gelingt dem Einbrecher das Rootkit zu installieren und er wird erst beim nächsten Durchlauf von AIDE enttarnt.

17.3.3.2 *`verifiedexec(4)`*

Ähnlich wie AIDE, allerdings proaktiv, funktioniert `verifiedexec`. `verifiedexec` implementiert einen Prüfsummenvergleich in den `exec()`-Pfad des Kernels. Dazu wird ebenfalls eine initiale Datenbank (bspw. mit `/usr/share/examples/veriexecctl/gen\_sha1`) erzeugt und im Bootprozeß via `veriexecctl(8)` in den Kernelspeicher geladen. Ruft man nun ein Binary auf, wird die Prüfsumme erneut gebildet und mit der Datenbank verglichen. Im Securitylevel 1 warnt der Kernel bei abweichender Prüfsumme vor Ausführung, im Level 2 wird die Ausführung des Binaries verweigert.

17.3.3.3 *Portsentry*

Portsentry ist ein vergleichbar einfaches IDS, das dazu eingesetzt wird auf einem Server die Verbindungsversuche auf bestimmte Ports zu überwachen und so Portscans aufzuklären. Wurde ein (vermeintlicher) Portscan detektiert, kann Portsentry die `/etc/hosts.deny` anpassen und so die Quelle des Angriffs blockieren.

Portsentry lässt sich bequem aus pkgsrc heraus installieren und anschließend auch recht einfach konfigurieren. Es müssen lediglich `/usr/pkg/etc/portsentry.conf` und `/usr/pkg/etc/portsentry.ignore` angepast werden. Erstere definiert die zu überwachenden Hosts und die entsprechende Reaktion bei Angriffen, zweitere zu ignorierende IP Adressen/Subnetze. Insbesondere die `portsentry.conf` ist ausreichend kommentiert und bereits mit Beispielen gefüllt.

17.3.3.4 *snort*

Snort ist *das* NIDS schlechthin. Es ist unter der GPL als Open Source erhältlich und erfreut sich weltweit großer Beliebtheit. Es funktioniert als NIDS in dem es den Netzwerkverkehr an einem Netzwerkinterface mitschneidet und bezgl. Angriffsignaturen analysiert. Trifft eine Angriffssignatur, kann es über die Rules bestimmte Aktionen starten.

Die Aktualisierung der Signaturdatenbank ist unter Snort sehr wichtig, da eben nur bereits bekannte Angriffe erkannt werden können.

Der Einsatz und die Konfiguration eines Snort-IDS ist für größere Netze relativ komplex und würde den Rahmen des Artikels sprengen. Außerdem habe ich keine Lust die FAQ abzuschreiben ;-), daher verweise ich hier auf (2004), (2006), (2006), (2003) und [groups,www].google.de

17.4 Multilayer IDS (mIDS)

Ein Problem der Systemüberwachung ist die Analyse verschiedenster Logdateien, die jeweils eine eigene Syntax und Semantik haben und nur bestimmte Teile der Infrastruktur überwachen. Die manuelle Analyse ist zeitaufwändig und fehleranfällig und ermöglicht nur sehr eingeschränkt das Erkennen von Verbindungen in den einzelnen Logs. Ein mIDS soll nun dazu dienen die verschiedenen Logdateiein in ein einziges Log zu aggregieren und ein Analyse aller Logs auf Zusammenhänge zu ermöglichen. Dazu ist es notwendig die Syntax und Semantik der Rohlogs zu kennen und diese ohne Informationsverlust in ein wohldefiniertes Einheitsformat des mIDS zu überführen.

Vorteile sind die Automatisierung algorithmisierbarer Vorgänge, so wird der Administrator entlastet, indem er nicht jedes Log manuell analysieren muss. Weiterhin wird die Analyse erleichtert indem logische Zusammenhänge zusammenfassend dargestellt werden.

Notwendige Design- und Implementierungsschritte:

1. Datensammlung:
   Zusammführen aller notwendigen Daten aus verteilten Logs im mIDS
2. Datenaufbereitung:
   verlustfreie Konvertierung der Rohdaten in das mIDS-Datenformat,
3. Analyse der aufbereiteten Daten:
   Aggregation der Informationen um Beziehungen aufzudecken
4. Alarmierung / Reaktion:
   regelbasierte Analyse der Daten bezgl. Angriffe und Alarmierung bzw. auch Gegenmaßnahmen

5. Präsentation der Ergebnisse:
   Darstellung der Analyse, von der blinken LED bis zum atomaren Erstschlag

Prinzipiell lassen sich nach den ersten beiden Schritten (Sammlung und Aufbereitung) beliebige Technologien ansetzen, man kann die Daten bspw. statistisch oder topologisch analysieren.

17.5 HONEYPOT

Ein Honeypot (oder Honeynet, wenn ein komplettes Netz betrieben wird), ist ein System, das als Lockmittel für Angreifer dient. Es läuft auf einem extra dafür bereitgestelltem Server und bietet keine sinnvollen Dienste an. Seine Existenz ist den normalen Netzbenutzern nicht bekannt, so das im Prinzip schon jeder einfache Verbindungsversuch als Angriff gewertet werden kann. Zusätzlich simuliert der Honeypot Dienste oder Betriebsyssteme und versucht so den Angreifer aus sich zu lenken. Im Hintergrund werden die Aktionen des Angreifers protokolliert und ggf. analysiert.

Der Honeypot dient also in der Regel folgenden Zielen:

- Anlocken und Binden des Angreifers
- Erkennung und Analyse der Angriffe
- Schutz der Produktivsysteme als »Roter Hering«[6]
- juristische Beweissicherung

Man möchte im Allgmeinen herausfinden wie ein Einbrecher in ein System eindringt und ob es neue Sicherheitslücken gibt. Hierzu ist es notwendig das Hostsystem gut zu sichern und die Aktivitäten des Honeypots adäquat zu loggen und auszuwerten. Man kann die so gewonnen Erkenntnisse verarbeiten und Sicherheitsrisiken ausmerzen.

Weiterhin kann man einen Honeypot dazu benutzen von den Produktivsystemen abzulenken und diese so aus dem Blickfeld des Angreifers bringen. Einfache Honeypots gibt es als sogenannte Tarpit,

das sind Programme die Portscanner (LaBrea) oder Mailspammer aufhalten sollen. Dazu werden bspw. offene Mailrelays emuliert, die bei Verbindungsaufnahme und versuchter Maileinlieferung die IP-Adresse auf den echten Mailservern blockt. Mehr dazu in (2005)

Da eigentlich ja schon eine einfache Kontaktaufnahme (telnet oder scan) zum Honeypot Angriffsabsichten verrät, enttarnt sich der Angreifer somit selbst und man kann geeignete Gegenmaßnahmen einleiten, während der Angreifer seine Zeit damit zubringt in ein, für ihn, nutzloses System einzudringen und dabei analysiert wird.

Wichtig ist, daß es dem Angreifer nicht gestattet wird den Honeypot für weitere Angriffe zu missbrauchen, daher sollte er im Netz entsprechend abgegrenzt werden und idealerweise mit einem Not-Aus ausgestattet werden, das den Hopeypot deaktiviert und so weiteren Schaden abwendet.

[6] bewusst ausgelegte falsche Fährte

17.5.1 *Analyse*

Die Daten, die ein Honeypot abwirft, können beachtliche Ausmaße annehmen und sollten daher nach bestimmten Kriterien untersucht werden. Interessante Analysepunkte sind bspw.:

- Angriffsquelle(n)
  woher stammt der Angriff? (IP-Adresse, Region (geoiplookup))
  wieviele Angreifer?
- Ausrüstung des Angreifers
  Betriebssystem, -version und Architektur mittels passivem Fingerprinting (pof, disco)
- well known attacker?
  ist die Angriffsquelle für Angriffe bekannt? (Blacklists)
- sequentielle/parallele Angriffe?
  wurden die Angriffe gleichzeitig oder nacheinander ausgeführt?

Wichtige Orte der Analyse sind:

- Netzwerkverkehr
  Mitschnitte des Honeypots, Gateways, Switch/Bridge
- Logdateien
  Logdateien relevanter Systeme wie Gateways, Firewalls, NIDS, physischen Systemen
- Systemplatten des physischen Honeypots
  Dateisystemintegrität prüfen und ggf. Manipulationen untersuchen

Honeypots lassen sich in verschiedene grobe Kategorien einteilen:

- **Tarpits/Traps**
  simulieren Netzwerkdienste (bspw. offene Ports oder Mailserver) und behindern den Angreifer indem sie die Verbindung unnötig lange offenhalten. Bsp: LaBrea, Portsentry
- **physisches Sytem**
  ein einfacher Rechner fungiert als äußerlich normales System, wird aber überwacht
- **logisches System**
  ein physisches System emuliert ein (1:1) oder mehqrere (1:n) Systeme, bspw. mit honeyd oder NetBSD/xen

Ist ein beliebiger Honeypot installiert, ist es ratsam ihn vor dem Produktiveinsatz zu testen, dazu kann man beliebige Angriffsszenarien durchspielen, bspw. in dem man ein Angriffsteam (Samurai) anmietet oder bekannte Schwachstellenscanner (SATAN, Nesus, Saint, Sara, ADMsmb) einsetzt.

Überwachen lässt sich ein logischer Honeypot am einfachsten, indem der physische Host die logischen System überwacht, bspw. in dem XEN unter NetBSD eingesetzt wird. Weiterhin ist es ratsam den gesamten anfallenden Netzverkehr zum System mitzuschneiden, bspw. mit dem guten alten tcpdump(8) (siehe (2006b)) oder einem Gateway-IDS wie hogwash. Ebenfalls nützlich ist es den Netzwerkvehr zum passiven Fingerprinting zu verwenden, in dem Programme wie pof den Datenverkehr analysieren und anhand von Signaturen das OS des Angreifers passiv identifizieren. Passiv heißt in diesem Fall, das kein Portscan oder ähnliches Verfahren gegen den Angreifer eingesetzt wird, da ihn dies verschrecken könnte, sondern einfach der sowieso von ihm erzeugte Netzwerkverkehr analysiert wird.

17.5.2 *Implementierungsprobleme bzw. Angriffsmöglichkeiten*

Ein Honeypot lebt wie ein Scharfschütze von der Tarnung. Findet der Angreifer heraus, das er an einen Honeypot geraten ist, wird er den Einbruchsversuch abbrechen und der Honeypot ist nutzlos geworden, daher ist es das höchste Ziel einen Honeypot zu erschaffen, der sich wie ein reales System verhält und somit nicht mehr enttarnbar ist. Leider steigt mit der Realitätsnähe des Honeypots auch die Implementierungshöhe und die Wartbarkeit des Systems.

Ein sehr guter Honeypot wäre ein echtes System, das als Roter Hering ausgelegt wird, da dies dem Angreifer nicht auffallen würde. Nützlich ist dieser Honeypot für den Administrator aber nur, wenn er in der Lage ist ihn idealerweise komplett zu überwachen. Dies ist bei einem normalen System recht problematisch, man kann zwar den Netzwerkverkehr mitschneiden, verwendet der Einbrecher aber selbst Krypotgraphie wie SSH nutzen die Aufzeichnungen nichts. Daher wäre es eine weitere Möglichkeit den Kernel dahingehend zu manipulieren bestimmte Ereignisse (`read()`, `execute()`) mitzuloggen. Nun entstehen aber zwei weitere Probleme:

- Wie bekomme ich die geloggten Daten sicher und unauffällig auf einen anderen Host?
- Wie verhindere ich die Enttarnung der Loggingmechanismen?

Beide Probleme sind nichttrivial, da man die Logdaten zwingend benötigt, sind sie doch der Hauptgrund für eine Installation des Honeypots. Eine Übertragung der Daten über Netzwerk, selbst verschlüsselt, könnte das Misstrauen des Einbrechers erregen und die Tarnung auffliegen lassen, einen Nadeldrucker mitlaufen zu lassen wäre zwar möglich, dazu muss aber der Druckprozeß sehr gut versteckt werden.
Das zweite Problem ist auch nichttrivial zu lösen, da ein solcher Logmechanismus im Kernelspace die Systemlast hochtreibt. Logt der Kernel alle `read()`-Aufrufe mit, entsteht eine sehr hohe Systemlast, die sofort auffällt. Dies kann sehr trivial mit `dd` und `ping` geschehen:

```
dd if=/dev/zero of=/dev/null bs=1
```

erzeugt sehr viele Leseaufrufe, die bei einem anschließenden

```
ping 127.0.0.1
```

die RTT signifikant ansteigen lassen.

Die Uberwachungsprobleme lassen sich durch den Einsatz virtueller Maschinen umgehen, also ein physisches System emuliert $1, ..., n$ logische Systeme. Das physische System übernimmt die Loggingfunktionen und ist solange vertrauenswürdig, bis es selbst kompromittiert wurde. Die Tarnung der logische Systeme hängt hier von der Realitätsnähe der Implementierung ab:

- gibt die Emulation real existierende Fehler des emulierten Systems wieder?
- Differenzen zwischem echten Datenverkehr und emuliertem? (Router, Traffic)
- MAC-Adressen der NICs?

Tarpits erzeugen oftmals keinen Traffic, so simuliert FakeAP zwar WLAN-Access Points, kann für diese aber keinen Traffic generieren, so daß diese gefälschten APs leicht erkennbar sind. Ebenso öffnet LaBrea als Tarpit zwar eine TCP/IP-Verbindung zum Angreifer, setzt danach aber die Windowsize auf 0, so das der Angreifer keine Daten mehr übertragen kann. Dieses Verhalten ist deterministisch und lässt daher eine solche Tarpit leicht enttarnen.

17.5.3 *Programme*

17.5.3.1 *LaBrea*

Im Prinzip so etwas wie Portsentry im Netz, LaBrea kann dazu virtuelle Hosts mit echten IP Adressen erstellen und Verbindungen an diese durch manipulierte IP-Pakete offen halten und verzögern. Dadurch werden Angreifer (z. B. Würmer, die das Netz durchsuchen oder Portscans) verlangsamt und im Idealfall sogar aufgehalten.

17.5.3.2 *honeyd*

honeyd von Niels Provos emuliert logische Hosts oder komplette Netze auf einem physischen System und ist in der Lage zu jedem emulierten Port eine bestimmte Reaktion auszuführen. Weiterhin ermöglicht es bestimmte Reaktionen auf Verbindungen, z. B. ein RESET oder ein Skript, das einen Webserver simuliert.

Dokumentation und Konfigurationshinweise zu honeyd finden sich auf
http://www.honeyd.org/.

Es wird eine Konfiguration namens »pit« erstellt, die ein Windows 98SE System emuliert und alle TCP-Ports öffnet, aber als tarpit den Werkehr darauf verzögert. UDP Ports werden blockiert und abschließend wird diese Konfiguration an der Adresse 192.168.1.1 gebunden.

[!h]

```
create pit
set pit personality "Microsoft Windows 98SE"
set pit default tcp action tarpit open
set pit default udp action block
bind 192.168.1.1 pit
```

Listing 17.1: Konfigurationsbeispiel für honeyd

17.6 FAZIT

IDS sind ein sehr effizientes Mittel um die Sicherheit eines Netzwerkes zu erhöhen. Prinzipiell gilt aber auch hier, das es einfacher und besser ist die Angriffsfläche zu minimieren und alle weiteren Maßnahmen zur Absicherung des Netzes zu ergreifen.

Ebenso sind alle Arten von Honeypots ein weiteres effizientes Mittel um Angreifer zu binden und zu analysieren, außerdem ist diese Methodik sehr gut geeignet um Angreifer aus dem Inneren abzulenken und zu enttarnen.

Soll ein IDS und/oder Honeypot eingesetzt werden, sind entsprechende vorbereitende Maßnahmen notwendig, da ein solches Projekt alle Methoden der Informatik erfordert und sehr schnell eine außerordentliche Komplexität erreichen kann.

Teil VI

DEN FEIND KENNEN

18 EIGENE SYSTEME AUFKLÄREN UND ANGREIFEN

Solange man selbst den Krieg nicht kennt, begreift man nicht, wo die Schwierigkeiten der Sache liegen.

(Generalmajor Carl von Clausewitz)

18.1 SCHWACHSTELLEN-SCANNER

Viele Schwachstellen in Systemen sind leicht aufzuspüren - werden aber auch häufig übersehen und erlauben so den Einbrechern in ein System einzusteigen. Vergegenwärtigt man sich aber, das ein steigender Anteil an Einbrüchen und Einbruchsversuchen von automatisierten Skripten ausgeführt wird, kann man zumindest diese Lücken schließen.

Ähnlich einem Virenscanner oder der Signaturerkennung in einem Einbruchserkennungssystem funktionieren Schwachstellenscanner. Ein Zielsystem wird auf Schwachstellen hin untersucht, welche in einem Signaturlexikon oder einer Bibliothek definiert werden. Erkennt man so Sicherheitslücken, kann man sie stopfen.

Auch hier gelten wieder die selben Einschränkungen wie bspw. bei Virenscannern, es können nur Schwachstellen erkannt werden, die bereits bekannt und im Lexikon sind. Gegen neuartige und unbekannte oder komplexe Sicherheitslücken oder Angriffsmethoden sind auch diese Automatismen wehrlos.

Schwachstellen-Scanner sind somit *ein* nützliches Werkzeug um automatisiert nach Lücken zu suchen, wie dies auch viele Skript-Kiddies oder anderes Gewürm tun. Man kann aber auch hier wieder davon ausgehen, das ein System, auf dem keine Sicherheitslücken gefunden wurden, nicht zwangsläufig »sicher« ist.

18.1.1 *Nessus*

[!h]

```
root@balmung {5} nessus-adduser
Using /var/tmp as a temporary file holder

Add a new nessusd user
----------------------


Login : nessustest
Authentication (pass/cert) [pass] :
Login password :
Login password (again) :

User rules
----------
nessusd has a rules system which allows you to restrict the hosts
that nessustest has the right to test. For instance, you may want
him to be able to scan his own host only.

Please see the nessus-adduser(8) man page for the rules syntax

Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)
accept 192.168.0.0/24
accept 127.0.0.1
default deny
^D

Login             : nessustest
Password          : ************
DN                :
Rules             :
accept 192.168.0.0/24
accept 127.0.0.1
default deny


Is that ok ? (y/n) [y] y
user added.
root@balmung {6}
```

Listing 18.1: Einen Nessus-Benutzer anlegen

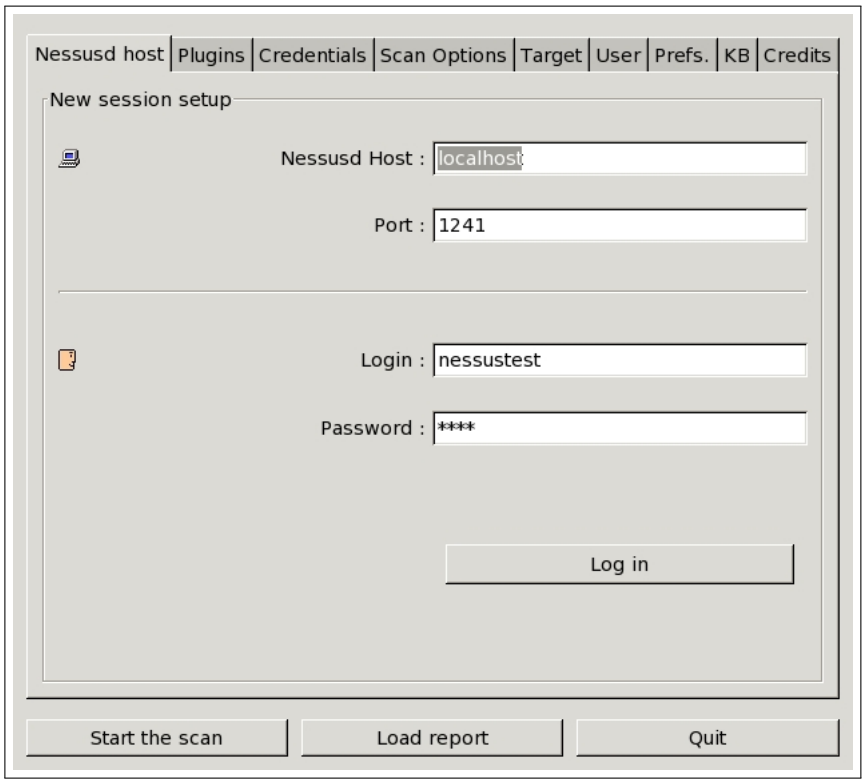


Abbildung 18.1.: Nessus starten und einloggen

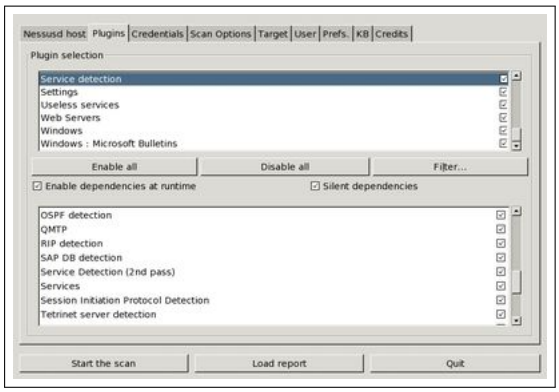


Abbildung 18.2.: Nessus-Plugins und Prüfoptionen konfigurieren

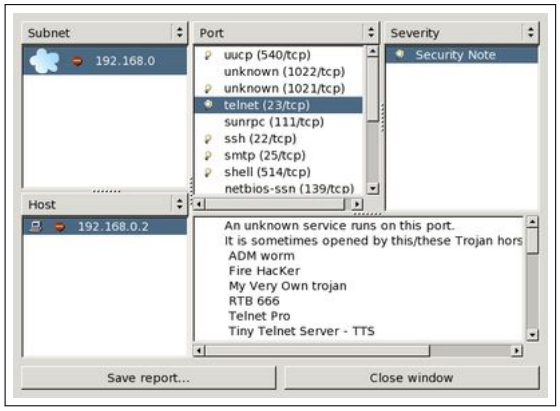


Abbildung 18.3.: Nessus-Prüfbericht

19 FEINDAUFKLÄRUNG: INFORMATIONEN ÜBER SICHERHEITSLÜCKEN UND EINBRECHER

Ihr seid alle Idioten zu glauben, aus Eurer Erfahrung etwas lernen zu können, ich ziehe es vor, aus den Fehlern anderer zu lernen, um eigene Fehler zu vermeiden.

(Otto von Bismarck)

19.1 NETBSD UND SICHERHEIT

NetBSD-Sicherheitsangelegenheiten werden vom NetBSD-Security-Officer oder dem NetBSD-Security-Team behandelt. Erreichbar ist der Security-Officer unter security-officer@NetBSD.org, sein PGP-Schlüssel ist 0xF8376205. Das Team ist unter security-alert@NetBSD.org mit PGP-Schlüssel 0xF8376205 erreichbar.

Außerdem können Fehler oder Probleme in NetBSD mit send-pr(1) eingeschickt werden. Bei schwerwiegenden Sicherheitslücken sollte allerdings eine verschlüsselte Email an die o.g. Adressen bevorzugt werden.

Sicherheitshinweise für das Basis-System werden über die »netbsd-announce«-Mailingliste verteilt und auf der NetBSD-Webseite unter `http://www.netbsd.org/Security/` archiviert.

Sicherheitsprobleme in installierten Pkgsrc-Paketen können mit dem Paket »audit-packages« aufgelöst werden, siehe Kap. 13.1, »Installierte Pakete auf Sicherheitslücken überprüfen« auf Seite 191 ff..

Neben NetBSD-spezifischen Informationsquellen existieren auch weitere Quellen die sich mit Sicherheit allgemein oder für spezielle Programme wie z. B. Apache befassen.

Empfehlenswert sind zum Beispiel die Mailinglisten »Bugtraq« und »Full Disclosure« sowie ein Abonnement der Sicherheitshinweise vom CERT.

Weitere Informationsquellen zu Sicherheitsangelegenheiten finden Sie in Kap. D, »Weitere Informationsquellen« auf Seite 273 ff..

SIEHE AUCH:
Kap. D, »Weitere Informationsquellen« auf Seite 273 ff.
http://www.netbsd.org/MailingLists/#netbsd-announce
Kap. 13.1, »Installierte Pakete auf Sicherheitslücken überprüfen« auf Seite 191 ff.

Teil VII

ANHÄNGE UND VERZEICHNISSE

A

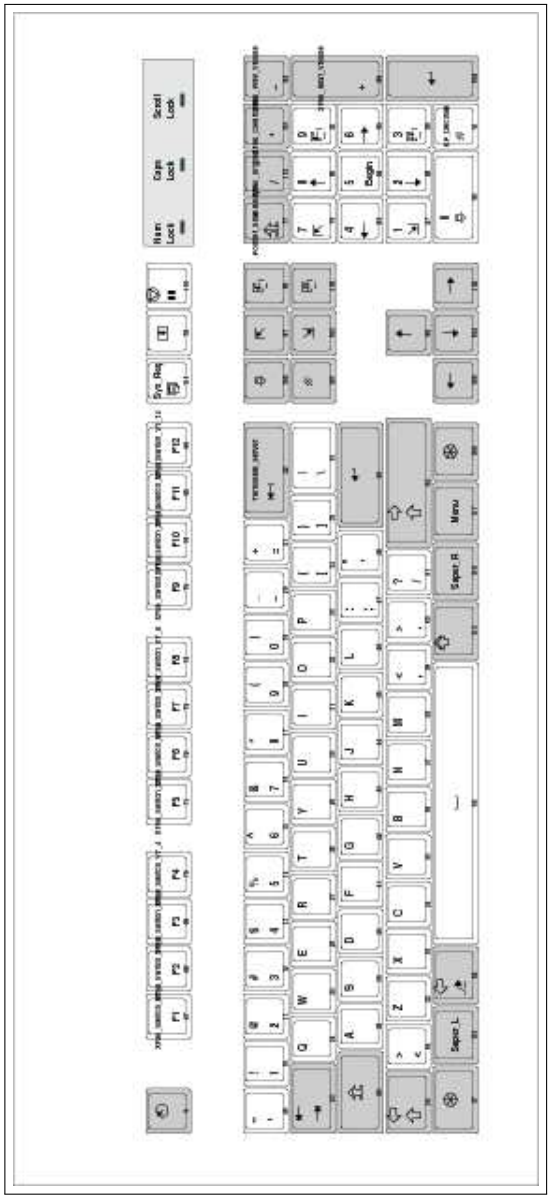
ICMP-PAKETTYPEN UND CODES

0	**Echo Reply**	
3	**Destination Unreachable**	
		0: Net Unreachable 1: Host Unreachable 2: Protocol Unreachable 3: Port Unreachable 4: Fragmentation Needed and Don't Fragment was Set 5: Source Route Failed 6: Destination Network Unknown 7: Destination Host Unknown 8: Source Host Isolated 9: Communication with Destination Network is Administratively Prohibited 10: Communication with Destination Host is Administratively Prohibited 11: Destination Network Unreachable for Type of Service 12: Destination Host Unreachable for Type of Service 13: Communication Administratively Prohibited 14: Host Precedence Violation 15: Precedence cutoff in effect
4	**Source Quench**	
5	**Redirect**	
		0: Redirect Datagram for the Network (or subnet) 1: Redirect Datagram for the Host 2: Redirect Datagram for the Type of Service and Network 3: Redirect Datagram for the Type of Service and Host
6	**Alternate Host Address**	
7	**Unassigned**	
8	**Echo**	

9	**Router Advertisement**	
		0: Normal router advertisement 16: Does not route common traffic
10	**Router Selection**	
11	**Time Exceeded**	
		0: Time to Live exceeded in Transit 1: Fragment Reassembly Time Exceeded
12	**Parameter Problem**	
		0: Pointer indicates the error 1: Missing a Required Option 2: Bad Length
13	**Timestamp**	
14	**Timestamp Reply**	
15	**Information Request**	
16	**Information Reply**	
17	**Address Mask Request**	
18	**Address Mask Reply**	
19 - 29	**Reserved**	
30	**Traceroute**	
31	**Datagram Conversion Error**	
32	**Mobile Host Redirect**	
33	**IPv6 Where-Are-You**	
34	**IPv6 I-Am-Here**	
35	**Mobile Registration Request**	
36	**Mobile Registration Reply**	
39	**SKIP**	
40	**Photuris**	
		0: Bad SPI 1: Authentication Failed 2: Decompression Failed 3: Decryption Failed 4: Need Authentication 5: Need Authorization

B

US-TASTATURLAYOUT



C THE HACKER MANIFESTO

Jeder wird als Faust geboren, um alles zu erfassen, alles zu erproben, alles auszudrücken. Daß Faust Gelehrter wurde, dafür sorgten die Fehler seiner Vorgänger und seiner Zeitgenossen.

(Boris Pasternak, »Doktor Schiwago«)

```
                              ==Phrack Inc.==

                   Volume One, Issue 7, Phile 3 of 10

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
The following was written shortly after my arrest...

                        \textbackslash/\textbackslash The Conscience of a Hacker/\textbackslash/

                                        by

                                   +++The Mentor+++

                              Written on January 8, 1986
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

        Another one got caught today, it's all over the papers.  ''Teenager
Arrested in Computer Crime Scandal'', ''Hacker Arrested after Bank Tampering''...
        Damn kids.  They're all alike.

        But did you, in your three-piece psychology and 1950's technobrain,
ever take a look behind the eyes of the hacker?  Did you ever wonder what
made him tick, what forces shaped him, what may have molded him?
        I am a hacker, enter my world...
        Mine is a world that begins with school... I'm smarter than most of
the other kids, this crap they teach us bores me...
        Damn underachiever.  They're all alike.

        I'm in junior high or high school.  I've listened to teachers explain
for the fifteenth time how to reduce a fraction.  I understand it.  ''No, Ms.
```

Smith, I didn't show my work. I did it in my head...''

Damn kid. Probably copied it. They're all alike.

I made a discovery today. I found a computer. Wait a second, this is cool. It does what I want it to. If it makes a mistake, it's because I screwed it up. Not because it doesn't like me...

Or feels threatened by me...

Or thinks I'm a smart ass...

Or doesn't like teaching and shouldn't be here...

Damn kid. All he does is play games. They're all alike.

And then it happened... a door opened to a world... rushing through the phone line like heroin through an addict's veins, an electronic pulse is sent out, a refuge from the day-to-day incompetencies is sought... a board is found.

''This is it... this is where I belong...''

I know everyone here... even if I've never met them, never talked to them, may never hear from them again... I know you all...

Damn kid. Tying up the phone line again. They're all alike...

You bet your ass we're all alike... we've been spoon-fed baby food at school when we hungered for steak... the bits of meat that you did let slip through were pre-chewed and tasteless. We've been dominated by sadists, or ignored by the apathetic. The few that had something to teach found us willing pupils, but those few are like drops of water in the desert.

This is our world now... the world of the electron and the switch, the beauty of the baud. We make use of a service already existing without paying for what could be dirt-cheap if it wasn't run by profiteering gluttons, and you call us criminals. We explore... and you call us criminals. We seek after knowledge... and you call us criminals. We exist without skin color, without nationality, without religious bias... and you call us criminals. You build atomic bombs, you wage wars, you murder, cheat, and lie to us and try to make us believe it's for our own good, yet we're the criminals.

Yes, I am a criminal. My crime is that of curiosity. My crime is that of judging people by what they say and think, not what they look like. My crime is that of outsmarting you, something that you will never forgive me for.

I am a hacker, and this is my manifesto. You may stop this individual, but you can't stop us all... after all, we're all alike.

+++The Mentor+++

Ach Gott! Die Kunst ist lang! Und kurz ist unser Leben.

(Johann Wolfgang von Goethe)

Weitere Informationsquellen, die sich auf Sicherheit und / oder NetBSD beziehen.

- **FreeBSD Sicherheitsinformationen**
  http://www.freebsd.org/security/index.html
- **OpenBSD Sicherheitsinformationen**
  http://openbsd.org/security.html
- **Computer Emergency Response Team / Coordination Center (CERT/CC)**
  veröffentlicht Sicherheitshinweise, Informationen zu Sicherheitslücken, Statistiken, Technische Informationen, Forschungsergebnisse und -analysen und vieles mehr http://www.cert.org/
- **Deutsches Forschungsnetz - Computer Emergency Response Team (DFN-CERT)**
  CERT-Dienst für das Deutsche Forschungsnetz http://www.dfn-cert.de/
- **Bugtraq**
  BugTraq is a full disclosure moderated mailing list for the *detailed* discussion and announcement of computer security vulnerabilities: what they are, how to exploit them, and how to fix them.
  FAQ: http://seclists.org/about/bugtraq.txt
  Archive: http://seclists.org/lists/bugtraq/
- **full disclosure**
  Unmoderierte Mailingliste über Sicherheitsthemen
  FAQ: http://lists.grok.org.uk/full-disclosure-charter.html
  Archive: http://seclists.org/lists/fulldisclosure/
- **Seclists.Org Security Mailing List Archive**
  Sicherheitsportal, das Sicherheitslisten aufführt und beschreibt
  http://seclists.org/
- **Astalavista**
  Verschiedene sicherheitsrelevante Themen wie Sicherheitslücken, Anleitungen und Nachrichten
  http://astalavista.com/

- **The Information Warfare Site (IWS)**
  IWS ist eine Diskussionsseite, die sich mit Sicherheitsthemen befaßt.
  http://www.iwar.org.uk/
- **SANS Newsletters**
  Computersicherheitsnachrichten des SANS
  http://www.sans.org/newsletters/
- **Packet Storm**
  Packet Storm offers an abundant resource of up-to-date and historical security tools, exploits, and advisories.
  http://www.packetstormsecurity.org/
- **2600**
  Viertelmonatliche Hacker-Postille
  http://www.2600.org/
- **Phrack**
  Hacker-Magazin
  http://www.phrack.org/
- **Die Datenschleuder**
  Chaos-Computer-Club-Magazin, Archiv auf der Webseite verfügbar
  http://ds.ccc.de/
- **de.comp.security.misc**, **de.comp.security.firewall**
  Deutsche Newsgruppen mit Sicherheitsbezogenen Diskussionsthemen
  Google Archive: http://groups.google.de/group/de.comp.security.misc
  Google Archive: http://groups.google.de/group/de.comp.security.firewall
  FAQ: http://www.iks-jena.de/mitarb/lutz/usenet/Firewall.html

E

VERWENDETE PAKETE

Nichts ist wahr, alles ist erlaubt.

(Friedrich Nietzsche)

[PP1] **crack**
The 'Sensible' Unix Password Cracker
pkgsrc/security/crack
gipkgsrc/security/crack/README.html
http://www.crypticide.com/users/alecm/

[PP2] **chkrootkit**
Locally checks for signs of a rootkit
pkgsrc/security/chkrootkit
pkgsrc/security/chkrootkit/README.html
http://www.chkrootkit.org/

[PP3] **apg**
Tool set for random password generation
pkgsrc/security/apg
pkgsrc/security/apg/README.html
http://www.adel.nursat.kz/apg/

[PP4] **hydra**
Login password cracker
pkgsrc/security/hydra
pkgsrc/security/hydra/README.html
http://www.thc.org/thc-hydra/

[PP5] **john**
Unix Password Cracker
pkgsrc/security/john
pkgsrc/security/john/README.html
http://www.openwall.com/john/

[PP6] **AIDE**
Free intrusion detection system for checking file integrity
pkgsrc/security/aide
pkgsrc/security/aide/README.html
http://sourceforge.net/projects/aide

[PP7] **snort**
The Open Source Network Intrusion Detection System
pkgsrc/net/snort
pkgsrc/net/snort/README.html
http://www.snort.org/

[PP8] **apache2**
Apache HTTP (Web) server, version 2
pkgsrc/www/apache2
pkgsrc/www/apache2/README.html
http://httpd.apache.org

[PP9] **postgresql**
Robust, next generation, object-relational DBMS
pkgsrc/databases/postgresql80
pkgsrc/databases/postgresql80/README.html
http://www.postgresql.org

[PP10] **arpwatch**
Monitor ARP activity on a network, maintaining a database
pkgsrc/net/arpwatch
pkgsrc/net/arpwatch/README.html
http://www-nrg.ee.lbl.gov/

[PP11] **fetchmail**
Batch mail retrieval/forwarding utility for pop2, pop3, apop, imap
pkgsrc/mail/fetchmail
pkgsrc/mail/fetchmail/README.html
http://catb.org/ esr/fetchmail/

[PP12] **fetchmailconf**
Configuration tool for fetchmail
pkgsrc/mail/fetchmailconf
pkgsrc/mail/fetchmailconf/README.html
http://catb.org/ esr/fetchmailconf/

[PP13] **anacron**
Configuration tool for fetchmail
pkgsrc/time/anacron
pkgsrc/time/anacron/README.html
http://anacron.sourceforge.net/

[PP14] **slurm**
Realtime traffic statistics
pkgsrc/net/slurm
pkgsrc/net/slurm/README.html

http://www.raisdorf.net/projects/slurm/

[PP15] **bmon**
Bmon is an interface bandwidth monitor
pkgsrc/net/bmon
pkgsrc/net/bmon/README.html
http://people.suug.ch/ tgr/bmon/

[PP16] **pkg_chk**
Check installed package versions against pkgsrc
pkgsrc/pkgtools/pkg_chk
pkgsrc/pkgtools/pkg_chk/README.html

[PP17] **audit-packages**
Tools to show vulnerabilities in installed packages
pkgsrc/security/audit-packages
pkgsrc/security/audit-packages/README.html

[PP18] **sudo**
Allow others to run commands as root
pkgsrc/security/sudo
pkgsrc/security/sudo/README.html
http://www.courtesan.com/sudo/

[PP19] **ddos-scan**
Scan for a limited set of distributed denial of service (ddos) agents
pkgsrc/security/ddos-scan
pkgsrc/security/ddos-scan/README.html
http://staff.washington.edu/dittrich/

[PP20] **adjustkernel**
Generate a NetBSD kernel config
pkgsrc/sysutils/adjustkernel
pkgsrc/sysutils/adjustkernel/README.html

[PP21] **cfs**
Encrypting file system, using NFS as its interface
pkgsrc/security/cfs
pkgsrc/security/cfs/README.html
http://www.crypto.com/software/

[PP22] **xscreensaver**
Screen saver and locker for the X window system
pkgsrc/x11/xscreensaver
pkgsrc/x11/xscreensaver/README.html
http://www.jwz.org/xscreensaver/

[PP23] **xlockmore-lite**
lite version of xlockmore

pkgsrc/x11/xlockmore-lite
pkgsrc/x11/xlockmore-lite/README.html
http://www.tux.org/ bagleyd/xlockmore.html

[PP24] **xlockmore**
Like the XLock session-locker/screensaver
pkgsrc/x11/xlockmore
pkgsrc/x11/xlockmore/README.html
http://www.tux.org/ bagleyd/xlockmore.html

[PP25] **xautolock**
Activate xlock after a user defined time of inactivity
pkgsrc/x11/xautolock
pkgsrc/x11/xautolock/README.html

[PP26] **xscreensaver-demo**
Demo viewer and configuration tool for xscreensaver
pkgsrc/x11/xscreensaver-demo
pkgsrc/x11/xscreensaver-demo/README.html
http://www.jwz.org/xscreensaver/

[PP27] **gpart**
Partition table rescue/guessing tool pkgsrc/sysutils/gpa
pkgsrc/x11/xscreensaver-demo/README.html
http://www.jwz.org/xscreensaver/

[PP28] **wipe**
Secure data destruction
pkgsrc/
pkgsrc//README.html

[PP29] **destroy**
Program to securely destroy files on the hard disk
pkgsrc/sysutils/destroy
pkgsrc//README.html

Hans Helmut Kirst (1954). *08/15 in der Kaserne, 08/15 im Krieg, 08/15 bis zum Ende*. Verlag Kurt Desch (siehe Seite 36).

Albanese, David J. u. a. (2004). „The Case for Using Layered Defenses to Stop Worms“. Engl. In: Seite 47. URL: http://www.nsa.gov/snac/support/WORMPAPER.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

Allen, Julia, Alan Christie und John McHugh (Sep. 2000). „Defending Yourself: The Role of Intrusion Detection Systems“. In: URL: http://www.cert.org/archive/pdf/IEEE_IDS.pdf (besucht am 06. 10. 2006) (siehe Seite 241).

Bundesamt für Sicherheit in der Informationstechnik (2006a). „BSI Leitfaden IT-Sicherheit“. De. In: URL: http://www.bsi.de/gshb/Leitfaden/GS-Leitfaden.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

– (2006b). „BSI-Standard 100-1: Managementsysteme für Informationssicherheit (ISMS)“. De. In: URL: http://www.bsi.de/literat/bsi_standard/standard_1001.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

– (2006c). „BSI-Standard 100-2: IT-Grundschutz-Vorgehensweise, Version 1.0“. De. In: URL: http://www.bsi.de/literat/bsi_standard/standard_1002.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

– (2006d). „BSI-Standard 100-3: Risikoanalyse auf der Basis von IT-Grundschutz“. De. In: URL: http://www.bsi.de/literat/bsi_standard/standard_1003.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

Bundesdruckerei (2007). „Sicherheitsmerkmale der Personalausweiskarte“. De. In: URL: http://www.bundesdruckerei.de/de/iddok/2_1/2_1_6.html (besucht am 07. 07. 2007) (siehe Seiten 54, 55).

Bundesregierung, Die (2007). „Drucksache 16/5507“. In: Antwort der Bundesregierung auf die Kleine Anfrage der Abgeordneten Jan Korte, Petra Pau, Ulla Jelpke, weiterer Abgeordneter und der Fraktion DIE LINKE.
– Drucksache 16/5228 –
Notwendigkeit neuer biometrischer Pässe aus Sicherheitsgründen. URL: http://dip.bundestag.de/btd/16/055/1605507.pdf (siehe Seite 52).

Byrne, D. (1971). *The attraction paradigm*. New York: Academic Press (siehe Seite 50).

Capability, CERT Coordination Center & Federal Computer Incident Response (2000). „CERT Advisory CA-2000-01: Denial-of-Service Developments“. In: URL: http://www.cert.org/advisories/CA-2000-01.html (besucht am 06. 10. 2006) (siehe Seite 145).

Clausewitz, Carl von (1832). *Vom Kriege*. Ferdinand Dümmler (siehe Seite 15).

Dowdeswell, Roland C. (2003). „The Cryptographic Disk Driver“. In: URL: http://www.imrryr.org/~elric/cgd/cgd.pdf (siehe Seite 178).

Eagly, A. H. u. a. (1991). „What is beautiful is good, but . . . : A meta-analytic review of research on the physical attractiveness stereotype“. In: *Psychological Bulletin* 110, 109
bibrangessep –
bibrangessep 128 (siehe Seite 50).

Efran, M. G. und E. W. J. Patterson (1976). „The politics of apperance“. In: *Unveröffentlichtes Manuscript an der Universität Toronto* (siehe Seite 50).

Emswiller, T., K. Deaux und J. Willits (1971). „Similarity, sex and requests for small favors“. In: *Journal of Applied Social Psychology* 1, 284
bibrangessep –
bibrangessep 291 (siehe Seite 50).

Eriksen, Marius Aamodt und Niels Provos (Jan. 2003). „Enges Korsett: Systrace setzt Regeln für erlaubte Systemaufrufe durch“. In: *Linux Magazin* 2003. URL: http://www.www.de/url (siehe Seite 102).

Goethe, Johann Wolfgang von (1774). *Die Leiden des jungen Werthers*. Leipzig: Weygand, 1
bibrangessep –
bibrangessep 224 (siehe Seite 48).

Goldberg, Ian u. a. (1996). „A Secure Environment for Untrusted Helper Applications“. In: *Proceedings of the 6th Usenix Security Symposium* (siehe Seite 116).

Grunwald, Lukas (2005). „Aufbau und Betrieb von Honeypot-Systemen“. In: URL: http://www.linuxdays.lu/agenda/lxd2005_security_tutorial/Attachment00089954/lxd2005_honeypot.pdf (besucht am 22. 12. 2005) (siehe Seite 247).

Gutmann, Peter (Juli 1996). „Secure Deletion of Data from Magnetic and Solid-State Memory“. In: *USENIX Security Symposium Proceedings*. San Jose, California: USENIX. URL: http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html (siehe Seite 124).

Hammermesh, D. und J. E. Biddle (1994). „Beauty and the labor market“. In: *The American Economic Review* 84, 1174
bibrangessep –
bibrangessep 1194 (siehe Seite 50).

Hill, G. W. (1982). „Group versus individual performance: Are N+1 heads better than one?“ In: *Psychological Bulletin* 91, 517
bibrangessep –
bibrangessep 539 (siehe Seite 49).

CERT-CC (1999). „Incident Note IN-99-07: Distributed Denial of Service Tools“. In: URL: http://www.cert.org/incident_notes/IN-99-07.html (besucht am 06. 10. 2006) (siehe Seite 145).

Ioannidis, Sotiris, Steven M. Bellovin und Jonathan M. Smith (2006). „Sub-Operating Systems: A New Approach to Application Security“. In: *Proceedings of the SIGOPS European Workshop*. SIGOPS@ (siehe Seite 116).

Laughlin, P. R. (1980). „Social combination processes in cooperative problem-solving groups in verbal intellective tasks.“ In: *Progress in social psychology*. Herausgegeben von M. Fishbein. Hillsdale, NJ: Lawrence Erlbaum (siehe Seite 49).

Lindner†, Studiendirektor Helmut, Dr. Harry Bauer† und Prof. Dr. Constans Lehmann (1999). *Taschenbuch der Elektrotechnik und Elektronik*. De. 7. völlig neubearbeitete Auflage. Leipzig: Fachbuchverlag Leipzig. ISBN: 3-446-21056-3 (siehe Seite 118).

Lupi, Federico und The NetBSD Foundation (2006). *The NetBSD Guide*. http://netbsd.org/guide/en/ (siehe Seite 178).

Mack, D. und D. Rainey (1990). „Female applicants' grooming and personnel selection". In: *Journal of Social Behavior and Personality* 9, 399
bibrangessep –-
bibrangessep 407 (siehe Seite 50).

Mackey, Richard (2002). „Layered Insecurity". In: URL: http://infosecuritymag.techtarget.com/2002/jun/insecurity.shtml (besucht am 06. 10. 2006) (siehe Seite 31).

McKusick, Marshall Kirk u. a. (1999). *The Design and the Implementation of the 4.4BSD Operating System*. En. 1. Auflage. Readin, Massachusetts: Addison-Wesley. ISBN: 0-201-54979-4 (siehe Seite 121).

Messmer, Hans Peter (1998). *Das PC-Hardware-Buch*. De. 1. Nachdruck der 5. Auflage von 1997. Bonn: Addison-Wesley. ISBN: 3-8273-1302-3 (siehe Seiten 118, 120, 133).

Moore, C. H. u. a. (1994). „The effects of physical attractiveness and social desirability on judgments regarding a sexual harassment case". In: *Journal of Social Behavior and Personality* 9, 715
bibrangessep –-
bibrangessep 730. URL: http://core.ecu.edu/psyc/wuenschk/Articles/JSB&P1994/JSB&P1994.htm (siehe Seite 50).

News (1988). „News". In: *Stanford Business School Magazine* 56, Seite 3 (siehe Seite 49).

OECD (2003). „Guidance on Safety Performance Indicators". Engl. In: (Interim Publication scheduled to be tested in 2003 - 2004 and revised in 2005), Seite 214. URL: http://www.oecd.org/dataoecd/60/39/21568440.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

Peterman, Matthias (2005). „Systrace-Restricted Login-Shell mit NetBSD". In: URL: http://wiki.bsd-crew.de/index.php/Systrace-Restricted_Login-Shell_mit_NetBSD (besucht am 21. 01. 2007) (siehe Seite 113).

Phillips, D. P. (1974). „The influence of suggestion on suicide: Substantive and theoretical implications of the Werther effect". In: *American Sociological Review* 39, 340
bibrangessep –-
bibrangessep 354 (siehe Seite 48).

– (1979). „Suicide, motor vehicle fatalities, and the mass media: Evidence toward a theory of suggestion". In: *American Journal of Sociology* 84, 1150
bibrangessep –-
bibrangessep 1174 (siehe Seite 48).

– (1980). „Airplane accidents, murder, and the mass media: Towards a theory of imitation and suggestion". In: *Social Forces* 58, 1001
bibrangessep –-
bibrangessep 1024 (siehe Seite 48).

Phillips, D. P. und L. L. Carstensen (1986). „Clustering of teenage suicides after television news stories about suicide“. In: *The New England Journal of Medicine* 315, 685
bibrangessep –-
bibrangessep 689 (siehe Seite 48).

– (1988). „The effect of suicide stories on various demongraphic groups, 1968 – 1985“. In: *Suicide and Life-Threatening Behavior* 18, 100
bibrangessep –-
bibrangessep 114 (siehe Seite 48).

Provos, Niels (2006). „Improving Host Security with System Call Policies“. In: URL: http://www.citi.umich.edu/articles/reports/citi-tr-02-3.pdf (besucht am 04. 11. 2006) (siehe Seite 102).

Rehman, Rafeeq Ur (2003). *Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*. Prentice Hall. ISBN: 978-0-13-140733-6. URL: http://phptr.com/content/images/0131407333/downloads/0131407333.pdf (besucht am 06. 10. 2006) (siehe Seite 246).

Reinke, Dr. Thomas (2004). „Sicheres Löschen magnetischer Datenträger“. In: URL: http://fhh.hamburg.de/stadt/Aktuell/weitere-einrichtungen/datenschutzbeauftragter/informationsmaterial/informationstechnik/sicheres-loeschen-pdf,property=source.pdf (siehe Seite 124).

Rivest, R. (Apr. 1992). *The MD5 Message-Digest Algorithm*. RFC 1321 (Informational). URL: http://www.ietf.org/rfc/rfc1321.txt (siehe Seite 62).

Roesch, Martin und Chris Green (2006). *SnortUsers Manual 2.3.0*. 1. Columbia, MD, USA: Sourcefire Inc. URL: http://www.snort.org/docs/snort_manual/ (besucht am 06. 10. 2006) (siehe Seite 246).

Schneier, Bruce (1997). *Applied Cryptography*. Addison-Wesley. ISBN: 0-471-11709-9 (siehe Seite 123).

Schumacher, Stefan (2004). „Einführung in kryptographische Methoden“. In: URL: http://cryptomancer.ath.cx/~stefan/21c3/21c3-kryptograhie-paper.pdf (siehe Seite 134).

– (2006a). „Methoden zur Datensicherung – Strategien und Techniken für NetBSD“. In: 154 Seiten. URL: http://www.net-tex.de/backup.pdf (siehe Seite 135).

– (2006b). „Spass im Netzwerk mit tcpdump & Co.“ In: URL: http://www.net-tex.de/tcpdump.pdf (siehe Seite 249).

– (2006c). „Verschlüsselte Dateisysteme für NetBSD“. In: *GUUG UpTimes*. ISSN: 1860-7683. URL: http://www.net-tex.de/netbsd/advocacy/guug-uptimes-cgd_cfs.pdf (siehe Seiten 117, 134).

Sedlacik, Lubomir (2003). „Encrypted swap with cgd(4)“. In: *Archiv einer Mailinglisten-Diskussion*. URL: http://mail-index.netbsd.org/tech-security/2003/04/17/0000.html (besucht am 06. 10. 2006) (siehe Seite 178).

Snort Core Team, The (2006). „The Snort FAQ“. In: URL: http://www.snort.org/docs/faq/1Q05/faq.pdf (besucht am 06. 10. 2006) (siehe Seite 246).

Spenneberg, Ralf (2004). „Sensoren im Netzwerk“. In: *Linux-Magazin Security Edition*. http://www.spenneberg.com/Snort(SecurityEdition)/2134.html (siehe Seite 246).

Steinbuch, Herausgeber Dr.-Ing. K. (1967). *Taschenbuch der Nachrichtenverarbeitung*. De. 2. überarbeitete Auflage von 1968. Berlin/Heidelberg: Springer-Verlag (siehe Seite 118).

Stewart II., J. E. (1980). „Defendant's attractiveness as a factor in the outcome of criminal trials: An observational study." In: *Journal of Applied Social Psychology* 10, 348
bibrangessep –-
bibrangessep 361 (siehe Seite 50).

Suedfeld, P., S. Bochner und C. Matas (1971). „Petitioner's attire and petition signing by peace demonstrators: A field experiment". In: *Journal of Applied Social Psychology* 10, 278
bibrangessep –-
bibrangessep 283 (siehe Seite 50).

Sun Microsystems (2005). *Von Grund auf sicher: Durch integrierte Schutzmerkmale sorgen Sun Produkte für interoperable, skalierbare und einfach zu managende Sicherheit im gesamten Unternehmen*. URL: http://de.sun.com/homepage/feature/2005/security/index.html (besucht am 06. 10. 2006) (siehe Seiten 19, 31).

Systems and Network Attack Center, Herausgeber (2001). *Defense in Depth: A practical strategy for achieving Information Assurance in today's highly networked environments*. Englisch. Ft. Meade, MD, USA: National Security Agency, Seite 5. URL: http://www.nsa.gov/snac/support/defenseindepth.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

– Herausgeber (2006). *The 60 Minute Network Security Guide: First Steps Towards a Secure Network Environment*. Engl. Ft. Meade, MD, USA: National Security Agency, Seite 48. URL: http://www.nsa.gov/snac/support/I33-011R-2006.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

Thompson, Ken (1984). „Reflections on Trusting Trust". In: *Communication of the ACM*. Association for Computing Machinery, Inc, 761
bibrangessep –-
bibrangessep 763. URL: http://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf (besucht am 26. 12. 2006) (siehe Seite 101).

National Security Agency (2001). *U.S. DoD 5220.22-M National Industrial Security Program Operating Manual (»NISPOM«)*. En. National Security Agency. URL: http://tscm.com/NISPOMSU.html (besucht am 15. 02. 2007) (siehe Seite 123).

„Die Verteidigung in der Tiefe angewandt auf IT-Systeme (Memento)" (2004). In: Herausgegeben von Secrétariat général de la défense nationale. URL: http://www.ssi.gouv.fr/de/vertrauen/documents/methods/mementodep-V1.1_de.pdf (besucht am 03. 01. 2005) (siehe Seite 31).

Warnick, D. H. und G. S. Sanders (1980). „The effects of group discussion on eyewitness accuracy". In: *Journal of Applied Social Psychology* 10, 249
bibrangessep –-
bibrangessep 259 (siehe Seite 49).

Wiedemann, P. u. a. (2002). „Neuordnung der Verfahren zur Risikobewertung und Standardsetzung im Umwelt- und Gesundheitsschutz, Übersicht über die praktizierten Verfahren und Reformvorschläge für die Verfahren zur Risikobewertung

und Standardsetzung im nationalen und internationalen Vergleich, Abschlussbericht zum BfS-Projekt StSch 4337“. De. In: Seite 86. URL: http://www.bmu.de/files/bilder/allgemein/application/pdf/rk_standards.pdf (besucht am 06. 10. 2006) (siehe Seite 31).

Wissenschaftlicher Beirat der Bundesregierung (1998). *Globale Umweltveränderungen Welt im Wandel: Strategien zur Bewältigung globaler Umweltrisiken, Jahresgutachten 1998*. De. Berlin: Springer-Verlag, Seite 409. ISBN: 3-540-65605-7. URL: http://www.wbgu.de/wbgu_jg1998.pdf (besucht am 06. 10. 2006) (siehe Seiten 23, 26, 27, 31).