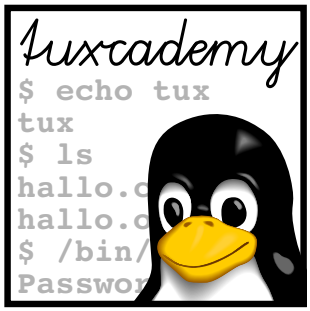
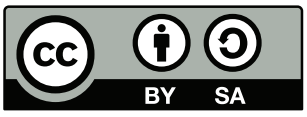
Version 4.0

Linux-Administration II

Linux im Netz





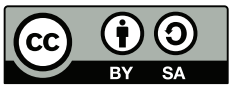
tuxcademy – Linux- und Open-Source-Lernunterlagen für alle
www.tuxcademy.org · info@tuxcademy.org

Diese Schulungsunterlage ist inhaltlich und didaktisch auf Inhalte der Zertifizierungsprüfung LPI-102 (LPIC-1, Version 4.0) des Linux Professional Institute *abgestimmt. Weitere Details stehen in Anhang B.*

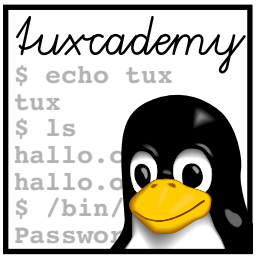
Das Linux Professional Institute *empfiehlt keine speziellen Prüfungsvorbereitungsmaterialien oder -techniken – wenden Sie sich für Details an* `info@lpi.org`.

Linux-Administration II **Linux im Netz**

Revision: `adm2:5c9bb3bd38d9a696:2015-08-20`
`adm2:7ff0ace1377051dd:2015-08-20` 1–13, B
`adm2:4uLNYnGzwqSXIf8a80ZVE`



Autoren: Tobias Elsner, Anselm Lingnau
Technische Redaktion: Anselm Lingnau ⟨*anselm@tuxcademy.org*⟩
Gesetzt in Palatino, Optima und DejaVu Sans Mono



Inhalt

1 Systemprotokollierung **13**
1.1 Das Problem . . . 14
1.2 Der Syslog-Daemon . . . 14
1.3 Die Protokolldateien . . . 17
1.4 Protokoll des Systemkerns . . . 18
1.5 Erweiterte Möglichkeiten: Rsyslog . . . 19
1.6 Die »nächste Generation«: Syslog-NG . . . 23
1.7 Das Programm `logrotate` . . . 28

2 Systemprotokollierung mit systemd und »dem Journal« **33**
2.1 Grundlagen . . . 34
2.2 Systemd und journald . . . 35
2.3 Protokollauswertung . . . 38

3 Grundlagen von TCP/IP **43**
3.1 Geschichte und Grundlagen . . . 44
3.1.1 Die Geschichte des Internet . . . 44
3.1.2 Verwaltung des Internet . . . 45
3.2 Technik . . . 46
3.2.1 Überblick . . . 46
3.2.2 Protokolle . . . 47
3.3 TCP/IP . . . 50
3.3.1 Überblick . . . 50
3.3.2 Kommunikation von Ende zu Ende: IP und ICMP . . . 51
3.3.3 Die Basis für Dienste: TCP und UDP . . . 54
3.3.4 Die wichtigsten Anwendungsprotokolle . . . 58
3.4 Adressen, Wegleitung und Subnetting . . . 59
3.4.1 Grundlagen . . . 59
3.4.2 Wegleitung . . . 60
3.4.3 IP-Netzklassen . . . 61
3.4.4 Subnetting . . . 62
3.4.5 Private IP-Adressen . . . 62
3.4.6 Masquerading und Portweiterleitung . . . 63
3.5 IPv6 . . . 64
3.5.1 Überblick . . . 64
3.5.2 IPv6-Adressierung . . . 65

4 Linux-Netzkonfiguration **69**
4.1 Netzschnittstellen . . . 70
4.1.1 Hardware und Treiber . . . 70
4.1.2 Netzwerkkarten konfigurieren mit `ifconfig` . . . 71
4.1.3 Wegleitung konfigurieren mit `route` . . . 72
4.1.4 Netzkonfiguration mit `ip` . . . 75

4.2 Dauerhafte Netzkonfiguration . . . 76
4.3 DHCP . . . 78
4.4 IPv6-Konfiguration . . . 80
4.5 Namensauflösung und DNS . . . 81

5 Fehlersuche und Fehlerbehebung im Netz 85
5.1 Einführung . . . 86
5.2 Lokale Probleme . . . 86
5.3 Erreichbarkeit von Stationen prüfen mit `ping` . . . 87
5.4 Wegleitung testen mit `traceroute` und `tracepath` . . . 89
5.5 Dienste überprüfen mit `netstat` und `nmap` . . . 92
5.6 DNS testen mit `host` und `dig` . . . 95
5.7 Andere nützliche Diagnosewerkzeuge . . . 98
5.7.1 `telnet` und `netcat` . . . 98
5.7.2 `tcpdump` . . . 100
5.7.3 `wireshark` . . . 100

6 `inetd` und `xinetd` 103
6.1 Netzdienste anbieten mit `inetd` . . . 104
6.1.1 Überblick . . . 104
6.1.2 Die Konfiguration des `inetd` . . . 104
6.2 Der TCP-Wrapper `tcpd` . . . 106
6.3 Der `xinetd` . . . 109
6.3.1 Überblick . . . 109
6.3.2 Die Konfiguration des `xinetd` . . . 109
6.3.3 Starten des `xinetd` . . . 111
6.3.4 Parallelverarbeitung von Anfragen . . . 111
6.3.5 `inetd` durch `xinetd` ersetzen . . . 112

7 Netzwerkdienste mit systemd 115
7.1 Vorbemerkung . . . 116
7.2 Persistente Netzwerkdienste . . . 116
7.3 Socket-Aktivierung . . . 118

8 Die Systemzeit 123
8.1 Einführung . . . 124
8.2 Uhren und Zeit unter Linux . . . 124
8.3 Zeitsynchronisation mit NTP . . . 126

9 Drucken unter Linux 135
9.1 Überblick . . . 136
9.2 Kommandos zum Drucken . . . 138
9.3 CUPS-Konfiguration . . . 142
9.3.1 Grundlagen . . . 142
9.3.2 Installation und Konfiguration eines CUPS-Servers . . . 144
9.3.3 Tipps und Tricks . . . 148

10 Die Secure Shell 151
10.1 Einführung . . . 152
10.2 Anmelden auf entfernten Rechnern mit `ssh` . . . 152
10.3 Andere nützliche Anwendungen: `scp` und `sftp` . . . 156
10.4 Client-Authentisierung über Schlüsselpaare . . . 156
10.5 Portweiterleitung über SSH . . . 159
10.5.1 X11-Weiterleitung . . . 159
10.5.2 Beliebige TCP-Ports weiterleiten . . . 160

11 Elektronische Post 165

11.1 Grundlagen . . . 166
11.2 MTAs für Linux . . . 166
11.3 Grundlegende Funktionen . . . 167
11.4 Verwaltung der Nachrichten-Warteschlange . . . 168
11.5 Lokale Zustellung, Aliasadressen und benutzerspezifische Weiterleitung . . . 169

12 Grundlagen von GnuPG **171**
12.1 Asymmetrische Kryptografie und das »Netz des Vertrauens« . . . 172
12.2 GnuPG-Schlüssel generieren und verwalten . . . 175
12.2.1 Schlüsselpaare generieren . . . 175
12.2.2 Öffentliche Schlüssel publizieren . . . 178
12.2.3 Öffentliche Schlüssel importieren und beglaubigen . . . 179
12.3 Daten verschlüsseln und entschlüsseln . . . 183
12.4 Dateien signieren und Signaturen prüfen . . . 184
12.5 GnuPG-Konfiguration . . . 186

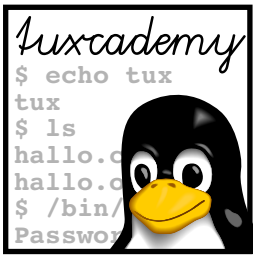
13 Linux und Sicherheit: Ein Einstieg **189**
13.1 Einführung . . . 190
13.2 Sicherheit im Dateisystem . . . 190
13.3 Benutzer und Dateien . . . 193
13.4 Ressourcenlimits . . . 197
13.5 Administratorprivilegien mit `sudo` . . . 201
13.6 Grundlegende Netzsicherheit . . . 205

A Musterlösungen **209**

B LPIC-1-Zertifizierung **219**
B.1 Überblick . . . 219
B.2 Prüfung LPI-102 . . . 220
B.3 LPI-Prüfungsziele in dieser Schulungsunterlage . . . 220

C Kommando-Index **225**

Index **229**



Tabellenverzeichnis

1.1 Kategorien für den `syslogd` 15
1.2 Prioritäten für den `syslogd` (nach aufsteigender Dringlichkeit) 16
1.3 Filterfunktionen für Syslog-NG 25

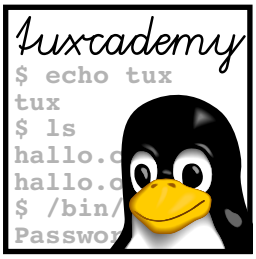
3.1 Gängige Anwendungsprotokolle auf TCP/IP-Basis 58
3.2 Beispiel für Adressenvergabe 60
3.3 Traditionelle IP-Netzklassen 61
3.4 Beispiel für Subnetting 62
3.5 Private IP-Adressbereiche nach RFC 1918 63

4.1 Optionen innerhalb `/etc/resolv.conf` 82

5.1 Wichtige Optionen von `ping` 88

6.1 Textersetzungen in Kommando-Einträgen bei `/etc/hosts.allow` und `/etc/hosts.deny` 107
6.2 Attribute in der Datei `/etc/xinetd.conf` 110
6.3 `xinetd` und Signale 111

13.1 Zugriffscodes für Prozesse bei `fuser` 195



Abbildungsverzeichnis

1.1 Beispiel-Konfiguration für `logrotate` (Debian GNU/Linux 8.0) . . . 28

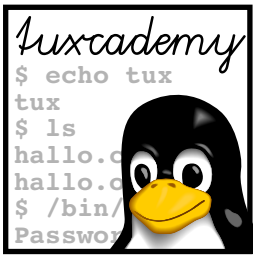
2.1 Vollständige Protokollausgabe mit `journalctl` 40

3.1 Protokolle und Dienstschnittstellen 48
3.2 ISO/OSI-Referenzmodell 49
3.3 Aufbau eines IP-Datagramms 52
3.4 Aufbau eines ICMP-Pakets 53
3.5 Aufbau eines TCP-Segments 54
3.6 Aufbau einer TCP-Verbindung: Der Drei-Wege-Handshake 55
3.7 Aufbau eines UDP-Datagramms 56
3.8 Die Datei `/etc/services` (Auszug) 57

4.1 Beispiel für `/etc/resolv.conf` 82
4.2 Die Datei `/etc/hosts` (SUSE) 83

7.1 Unit-Datei für Secure-Shell-Daemon (Debian 8) 121

9.1 Die Datei `mime.types` (Auszug) 142
9.2 Die Datei `/etc/cups/mime.convs` (Auszug) 143
9.3 Die CUPS-Web-Oberfläche 144
9.4 Die CUPS-Web-Oberfläche: Druckerverwaltung 145
9.5 Die CUPS-Web-Oberfläche: Drucker hinzufügen 145
9.6 Datei `/etc/cups/printers.conf` (Auszug) 147



Vorwort

Diese Schulungsunterlage vermittelt das Wissen, das für Konfiguration und Betrieb eines Linux-Arbeitsplatzrechners in einem (existierenden) lokalen Netz erforderlich ist.

Sie wendet sich an fortgeschrittene Linux-Administratoren und setzt Kenntnisse voraus, wie sie etwa in der LPI-101-Prüfung abgefragt werden. Dazu gehört eine solide Beherrschung der Shell, eines Texteditors und der grundlegenden Linux-Kommandos sowie der Grundzüge der Linux-Administration. Ferner baut diese Schulungsunterlage auf der Unterlage *Linux für Fortgeschrittene* auf, in der Themen wie Shellprogrammierung, `sed` und `awk`, `cron` und `at` behandelt worden sind.

Nach einer Einführung in den Systemprotokolldienst, die Grundlagen von TCP/IP und der Linux-Netzkonfiguration widmet diese Schulungsunterlage sich ausführlich dem Thema „Fehlersuche im Netz“ und erklärt das Starten von Diensten über `inetd` und `xinetd`. Ferner behandeln wir Themen wie die Verwaltung der Systemzeit, das Drucken und wichtige Netzdienste wie die Secure Shell und die Anbindung eines Clients an einen Mailserver. Die Unterlage schließt mit einer Einführung in die Verschlüsselung von Dateien mit GnuPG und einen Überblick über Linux-Sicherheit.

Das erfolgreiche Durcharbeiten dieser Schulungsunterlage oder vergleichbare Kenntnisse sind Voraussetzung für den erfolgreichen Besuch weiterer Linux-Kurse und für eine Zertifizierung beim *Linux Professional Institute*.

Diese Schulungsunterlage soll den Kurs möglichst effektiv unterstützen, indem das Kursmaterial in geschlossener, ausführlicher Form zum Mitlesen, Nach- oder Vorarbeiten präsentiert wird. Das Material ist in Kapitel eingeteilt, die jeweils für sich genommen einen Teilaspekt umfassend beschreiben; am Anfang jedes Kapitels sind dessen Lernziele und Voraussetzungen kurz zusammengefasst, am Ende finden sich eine Zusammenfassung und (wo sinnvoll) Angaben zu weiterführender Literatur oder WWW-Seiten mit mehr Informationen.

Kapitel

Lernziele
Voraussetzungen

Zusätzliches Material oder weitere Hintergrundinformationen sind durch das »Glühbirnen«-Sinnbild am Absatzanfang gekennzeichnet. Zuweilen benutzen diese Absätze Aspekte, die eigentlich erst später in der Schulungsunterlage erklärt werden, und bringen das eigentlich gerade Vorgestellte so in einen breiteren Kontext; solche »Glühbirnen«-Absätze sind möglicherweise erst beim zweiten Durcharbeiten der Schulungsunterlage auf dem Wege der Kursnachbereitung voll verständlich.

Absätze mit dem »Warnschild« weisen auf mögliche Probleme oder »gefährliche Stellen« hin, bei denen besondere Vorsicht angebracht ist. Achten Sie auf die scharfen Kurven!

Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind numeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad angegeben. Aufgaben, die mit einem Ausrufungszeichen (»!«) gekennzeichnet sind, sind besonders empfehlenswert.

Übungsaufgaben

Auszüge aus Konfigurationsdateien, Kommandobeispiele und Beispiele für die Ausgabe des Rechners erscheinen in `Schreibmaschinenschrift`. Bei mehrzeiligen Dialogen zwischen Benutzer und Rechner werden die Benutzereingaben in **`fetter Schreibmaschinenschrift`** angegeben, um Missverständnisse zu vermeiden. Wenn Teile einer Kommandoausgabe ausgelassen wurden, wird das durch »◁◁◁◁◁« kenntlich gemacht. Manchmal sind aus typografischen Gründen Zeilenumbrüche erforderlich, die in der Vorlage auf dem Rechner nicht stehen; diese werden als »▷ ◁« dargestellt. Bei Syntaxdarstellungen stehen Wörter in spitzen Klammern (»⟨*Wort*⟩«) für »Variable«, die von Fall zu Fall anders eingesetzt werden können; Material in eckigen Klammern (»[`-f` ⟨*Datei*⟩]«) kann entfallen und ein vertikaler Balken trennt Alternativen (»`-a`|`-b`«).

Wichtige Konzepte
Definitionen

Wichtige Konzepte werden durch »Randnotizen« hervorgehoben; die **Definitionen** wesentlicher Begriffe sind im Text fett gedruckt und erscheinen ebenfalls am Rand.

Verweise auf Literatur und interessante Web-Seiten erscheinen im Text in der Form »[GPL91]« und werden am Ende jedes Kapitels ausführlich angegeben.

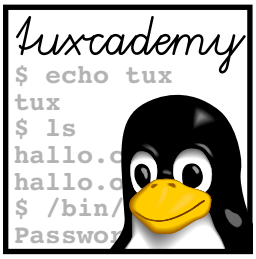
Wir sind bemüht, diese Schulungsunterlage möglichst aktuell, vollständig und fehlerfrei zu gestalten. Trotzdem kann es passieren, dass sich Probleme oder Ungenauigkeiten einschleichen. Wenn Sie etwas bemerken, was Sie für verbesserungsfähig halten, dann lassen Sie es uns wissen, etwa indem Sie eine elektronische Nachricht an

```
info@tuxcademy.org
```

schicken. (Zur Vereinfachung geben Sie am besten den Titel der Schulungsunterlage, die auf der Rückseite des Titelblatts enthaltene Revisionsnummer sowie die betreffende(n) Seitenzahl(en) an.) Vielen Dank!

LPIC-1-Zertifizierung

Diese Unterlage ist Teil eines Kurskonzepts zur Vorbereitung auf die LPIC-1-Zertifizierung. Anhang B enthält hierzu mehr Informationen.



1

Systemprotokollierung

Inhalt

1.1 Das Problem 14
1.2 Der Syslog-Daemon 14
1.3 Die Protokolldateien 17
1.4 Protokoll des Systemkerns 18
1.5 Erweiterte Möglichkeiten: Rsyslog 19
1.6 Die »nächste Generation«: Syslog-NG 23
1.7 Das Programm `logrotate` 28

Lernziele

- Den Syslog-Daemon kennen und konfigurieren können
- Protokolldateien mit `logrotate` verwalten können
- Verstehen, wie der Linux-Kernel mit Nachrichten umgeht

Vorkenntnisse

- Grundkenntnisse der Komponenten eines Linux-Systems
- Umgang mit Konfigurationsdateien

1.1 Das Problem

Anwendungsprogramme haben ihren Benutzern von Zeit zu Zeit etwas mitzuteilen. Der Vollzug einer Aufgabe, aber auch eine Fehlersituation oder Warnung müssen in geeigneter Form gemeldet werden. Textorientierte Programme geben entsprechende Nachrichten auf ihrem »Terminal« aus; Programme mit grafischer Oberfläche verwenden zum Beispiel »Alarmboxen« oder Statuszeilen mit wechselndem Inhalt.

Der Betriebssystemkern und die diversen im Hintergrund laufende System- und Netzwerkdienste sind dagegen mit keinem Benutzerterminal direkt verbunden. Wenn ein solcher Prozess eine Nachricht ausgeben will, dann tut er das normalerweise auf dem Bildschirm der Systemkonsole direkt; unter X11 erscheinen solche Nachrichten im `xconsole`-Fenster.

Im Mehrbenutzerbetrieb ist die Methode, eine Systemmeldung ausschließlich als Fehlermeldung auf die Systemkonsole zu schreiben, nicht ausreichend. Zum einen ist nicht sichergestellt, dass die Meldung auch von `root` gelesen werden, zum anderen lassen sich die Bildschirmmeldungen nicht sichern und gehen leicht verloren.

1.2 Der Syslog-Daemon

Die Lösung dieses Problems bietet der Syslog-Daemon oder `syslogd`. Anstelle der direkten Ausgabe einer Meldung können Systemmeldungen von spezieller Bedeutung mit der `syslog()`-Funktion ausgegeben werden, die Bestandteil der Standard-C-Bibliothek für Linux ist. Solche Meldungen werden vom `syslogd` entgegengenommen. Der `syslogd` erhält die Nachrichten über das Socket `/dev/log`.

Um Nachrichten vom Systemkern kümmert sich eigentlich ein anderes Programm namens `klogd`. Dieses Programm verarbeitet die Nachrichten vor und leitet sie normalerweise an den `syslogd` weiter. Siehe hierzu Abschnitt 1.4

Protokoll

Der `syslogd` erweist sich bei der Fehlersuche als sehr nützlich. Er protokolliert die verschiedenen Systemmeldungen und ist – wie der Name schon andeutet – ein Daemon-Programm. Der `syslogd` wird in der Regel beim Booten über ein Init-Skript gestartet. Erhält der `syslogd` Meldungen, so schreibt er sie z. B. in eine Datei oder sendet sie über das Netz weiter an einen Rechner, der ein zentrales Protokoll verwaltet.

Die gängigen Distributionen (Debian GNU/Linux, Ubuntu, Red Hat Enterprise Linux, Fedora, openSUSE, …) benutzen alle schon seit mehr oder weniger langer Zeit ein Paket namens »Rsyslog«, das eine modernere Implementierung eines `syslogd` mit mehr Konfigurationsmöglichkeiten enthält. Die zusätzlichen Möglichkeiten sind für den Einstieg und die Zwecke der LPI-Prüfung allerdings nicht wirklich wichtig. Wenn Sie den Anfang der Rsyslog-Konfiguration überlesen, entspricht das Ende im Wesentlichen dem, was in diesem Kapitel besprochen wird. Mehr über Rsyslog steht in Abschnitt 1.5.

Gewisse Versionen der SUSE-Distributionen, insbesondere des SUSE Linux Enterprise Servers (SLES) verwenden statt `syslogd` das Paket Syslog-NG, das substanziell anders konfiguriert wird. Für die LPI-Prüfung müssen Sie wissen, dass Syslog-NG existiert und was es in etwa macht; siehe dazu Abschnitt 1.6.

`/etc/syslog.conf`

Der Administrator bestimmt, was genau mit den einzelnen Meldungen geschehen soll. In der Konfigurationsdatei `/etc/syslog.conf` ist festgelegt, welche Meldungen wohin geschrieben werden sollen.

Tabelle 1.1: Kategorien für den `syslogd`

Kategorie	Bedeutung
`authpriv`	Vertrauliche Meldungen der Sicherheitsdienste
`cron`	Meldungen von `cron` und `at`
`daemon`	Meldungen von Daemon-Programmen ohne eigene Kategorie
`ftp`	Meldungen des FTP-Daemons
`kern`	Systemmeldungen aus dem Betriebssystemkern
`lpr`	Meldungen des Druckersystems
`mail`	Meldungen des Mailsystems
`news`	Meldungen des Usenet-News-Systems
`syslog`	Meldungen des `syslogd`
`user`	Meldungen, die mit Benutzern zu tun haben
`uucp`	Meldungen des UUCP-Systems
`local`*r*	($0 \leqslant r \leqslant 7$) Frei verwendbar für lokale Nachrichten

Der Rsyslog verwendet standardmäßig `/etc/rsyslog.conf` als Konfigurationsdatei. Diese ist in weiten Teilen kompatibel zu dem, was `syslogd` benutzen würde. Ignorieren Sie einfach alle Zeilen, die mit einem Dollarzeichen (`$`) anfangen.

Die Konfigurationsdatei besteht aus zwei Spalten und könnte folgendermaßen aussehen:

```
kern.warn;*.err;authpriv.none    /dev/tty10
kern.warn;*.err;authpriv.none   |/dev/xconsole
*.emerg                          *
*.=warn;*.=err                  -/var/log/warn
*.crit                           /var/log/warn
*.*;mail.none;news.none         -/var/log/messages
```

Die erste Spalte jeder Zeile bestimmt, welche Meldungen ausgewählt werden sollen, und die zweite Spalte gibt an, wohin die Meldungen geschrieben werden. Die erste Spalte hat das Format:

⟨*Kategorie*⟩.⟨*Priorität*⟩[;⟨*Kategorie*⟩.⟨*Priorität*⟩]…

Kategorien

Dabei bezeichnet die ⟨*Kategorie*⟩ das Systemprogramm oder die Komponente des Systems, die die Meldung verursacht. Das kann zum Beispiel der Mail-Daemon sein, der Kernel selbst oder auch Programme, die den Zugang zum System kontrollieren. Tabelle 1.1 zeigt die gültigen Kategorien. Wenn Sie anstelle der Kategorie einen Stern (»*«) setzen, steht dieser als Platzhalter für alle Kategorien. Es ist nicht ohne Weiteres möglich, eigene zusätzliche Kategorien zu definieren; die »lokalen« Kategorien `local0` bis `local7` sollten aber für die meisten Zwecke ausreichen.

Prioritäten

Die ⟨*Priorität*⟩ gibt an, wie schwerwiegend die Meldung ist. Die gültigen Prioritäten stehen in Tabelle 1.2.

Wer legt eigentlich fest, in welche Kategorie und mit welcher Priorität eine Meldung verschickt wird? Die Lösung ist simpel: Derjenige, der die `syslog()`-Funktion verwendet, nämlich der Entwickler der jeweiligen Software, muss Kategorie und Priorität der Meldungen seiner Routinen festlegen. Viele Programme lassen es zu, dass der Administrator zumindest die Kategorie der Meldungen umdefiniert.

Auswahlkriterien

Ein Auswahlkriterium der Form `mail.info` bedeutet »alle Meldungen des Mail-Daemon mit der Priorität `info` und höher«. Möchten Sie nur die Meldungen einer einzigen Prioritätsstufe erfassen, dann können Sie das über ein Auswahlkriterium wie `mail.=info` tun. Der Stern (»*«) steht für alle Prioritäten (Sie könnten

Tabelle 1.2: Prioritäten für den `syslogd` (nach aufsteigender Dringlichkeit)

Priorität	Bedeutung
`none`	Keine Priorität im eigentlichen Sinne – dient dazu, alle Nachrichten einer Herkunftskategorie auszuschließen
`debug`	Mitteilung innerer Programmzustände bei der Fehlersuche
`info`	Protokollierung des normalen Betriebsgeschehens
`notice`	Dokumentation besonders bemerkenswerter Situationen im Rahmen des normalen Betriebs
`warning`	(oder `warn`) Warnung über Zustände, die nicht gravierend sind, aber nicht mehr zum normalen Betrieb gehören
`err`	Fehlermeldungen aller Art
`crit`	Kritische Fehlermeldungen (die Grenze zur Priorität `err` ist nicht definiert)
`alert`	»Alarmierende« Nachrichten, die sofortiges Eingreifen erfordern
`emerg`	Die letzten Meldungen vor dem Absturz

auch »`debug`« angeben). Ein vorgestelltes `!` bedeutet Negation: `mail.!info` deselektiert Meldungen des Mail-Daemons mit einer Priorität von `info` und höher; dies ist vor allem in Kombinationen wie `mail.*;mail.!err` sinnvoll, um gewisse Meldungen niedriger Prioritätsstufen auszuwählen. `!` und `=` können kombiniert werden; `mail.!=info` deselektiert (genau) die Nachrichten des Mail-Daemons mit der Priorität `info`.

Mehrere Kategorien – selbe Priorität

Sie können auch mehrere Kategorien mit derselben Priorität in der Form `mail,news.info` angeben; dieser Ausdruck selektiert Meldungen der Priorität `info` und höher, die zu den Kategorien `mail` oder `news` gehören.

Aktionen

Nun zur rechten Spalte, dem Ziel der Meldungen. Die Nachrichten können auf verschiedene Weise verarbeitet werden:

- Sie können in eine *Datei* geschrieben werden. Dazu wird der absolute Dateiname angegeben. Normalerweise schreibt `syslogd` seine Meldungen *synchron*, also ohne Zwischenspeicherung im Platten-Cache des Linux-Kerns, auf die Platte, damit bei einem Systemabsturz möglichst viele Indizien gesichert sind. Steht vor der Pfadangabe ein »`-`«, ist eine Pufferung im Platten-Cache möglich. Das geht schneller, aber heißt, dass Sie im Fall eines Absturzes möglicherweise ausstehende Protokollnachrichten verlieren – für relativ unwichtige Nachrichten, etwa solche der Priorität `notice` und darunter, oder für Nachrichten »geschwätziger« Kategorien wie Mail und News ist das aber vielleicht nicht wirklich ein Problem.

  Der Dateiname darf sich auch auf ein Gerät (etwa `/dev/tty10` im obigen Beispiel) beziehen.

- Protokollnachrichten können in eine *benannte Pipe* (FIFO) geschrieben werden. Dazu muss der FIFO-Name als absoluter Pfad mit einem davorstehenden »`|`« angegeben werden. Ein solcher FIFO ist `/dev/xconsole`.

- Sie können *über das Netz* an einen anderen `syslogd` weitergeleitet werden. Dazu wird der Name oder die IP-Adresse des Zielrechners mit einem vorgestellten `@`-Zeichen angegeben. Das ist besonders sinnvoll, wenn es zu einem kritischen Systemzustand kommt, bei dem der Zugriff auf die lokale Nachrichtendatei nicht mehr möglich ist; um das Protokoll dem Zugriff allfälliger Cracker zu entziehen, die gerne ihre Spuren verwischen würden; oder um die Protokollnachrichten aller Rechner im Netz auf einem Rechner zu sammeln und gemeinsam auszuwerten.

  Auf dem Zielrechner muss der `syslogd` mit der Option `-r` (engl. *remote*) gestartet worden sein, damit er Meldungen anderer Rechner annimmt. Wie Sie das am geschicktesten machen, hängt von Ihrer Distribution ab.

- Sie können *direkt an Benutzer* geschickt werden. Die Benutzernamen müssen dazu in einer durch Kommata getrennten Liste aufgeführt werden. Die Nachricht wird den aufgelisteten Benutzern auf dem Terminal angezeigt, sofern diese im Augenblick des Eingangs der Nachricht angemeldet sind.
- Sie können *an alle eingeloggten Benutzer* geschickt werden, indem ein Stern »*« anstelle des Loginnamens angegeben wird.

In der Regel enthält Ihr System nach der Installation bereits einen laufenden `syslogd` und die Datei `/etc/syslog.conf` in einer brauchbaren Konfiguration. Falls Sie weitere Meldungen protokollieren lassen möchten, z. B. weil spezielle Probleme auftreten, sollten Sie die Datei `syslog.conf` editieren und dann den `syslogd` mit dem Signal `SIGHUP` anweisen, seine Konfigurationsdatei erneut zu lesen. Konfiguration ändern

Testen können Sie den `syslogd`-Mechanismus übrigens mit dem Programm `logger`. Ein Aufruf der Form

```
$ logger -p local0.err -t TEST "Hallo Welt"
```

produziert eine Protokollnachricht der Form

```
Aug  7 18:54:34 red TEST: Hallo Welt
```

Die meisten modernen Programmiersprachen enthalten eine Zugriffsmöglichkeit auf die `syslog()`-Funktion.

Übungen

1.1 [!2] Finden Sie heraus, wann zuletzt jemand per `su` auf Ihrem Rechner die Identität von `root` angenommen hat.

1.2 [!2] Konfigurieren Sie den `syslogd` so, dass er zusätzlich zur aktuellen Konfiguration alle (!) Meldungen in eine neue Datei `/var/log/test` protokolliert. Testen Sie das Ergebnis.

1.3 [3] (Braucht zwei Rechner und eine funktionierende Netzverbindung.) Konfigurieren Sie den `syslogd` auf einem Rechner so, dass er Protokollnachrichten über das Netz entgegennimmt. Konfigurieren Sie den `syslogd` auf dem anderen Rechner so, dass er Nachrichten der Kategorie `local0` auf den ursprünglichen Rechner schickt. Testen Sie die Konfiguration.

1.4 [2] Wie können Sie einen Protokollmechanismus implementieren, der sicher vor Angreifern ist, die den protokollierenden Rechner unter ihre Kontrolle bringen? (Ein Angreifer kann immer verhindern, dass weitere Protokollnachrichten geschrieben werden. Wir möchten gerne erreichen, dass der Angreifer bereits geschriebene Protokollnachrichten nicht verfälschen oder löschen kann.)

1.3 Die Protokolldateien

Protokolldateien werden in der Regel unter `/var/log` abgelegt. Die Namen der einzelnen Dateien variieren – prüfen Sie gegebenenfalls die `syslog.conf`-Datei. Hier sind einige Beispiele: /var/log

Debian GNU/Linux sammelt sämtliche Nachrichten außer denen, die mit Authentisierung zu tun haben, in der Datei `/var/log/syslog`. Für die Kategorien `auth`, `daemon`, `kern`, `lpr`, `mail`, `user` und `uucp` gibt es jeweils eigene Protokolldateien `auth.log` usw. Das Mailsystem verwendet außerdem die Dateien

mail.info, mail.warn und mail.err, die jeweils nur die Nachrichten der Prioritäten info usw. (und darüber) enthalten. Debug-Nachrichten aller Kategorien außer authpriv, news und mail landen in /var/log/debug und Nachrichten der Prioritäten info, notice und warn in allen Kategorien außer den eben genannten sowie cron und daemon in /var/log/messages.


Die Voreinstellungen von Ubuntu entsprechen denen von Debian GNU/Linux.


Bei den Red-Hat-Distributionen kommen alle Meldungen mit der Priorität info oder darüber außer denen von authpriv und cron in die Datei /var/log/messages, die Meldungen von authpriv in die Datei /var/log/secure und die von cron nach /var/log/cron. Alle Nachrichten des Mailsystems landen in /var/log/maillog.


OpenSUSE protokolliert alle Nachrichten außer denen von iptables und den Kategorien news und mail nach /var/log/messages. Nachrichten von iptables landen in /var/log/firewall. Nachrichten, die nicht von iptables kommen und außerdem die Priorität warn, err oder crit haben, werden ferner nach /var/log/warn geschrieben. Außerdem gibt es die Protokolldateien /var/log/localmessages für Nachrichten aus den local*-Kategorien, /var/log/NetworkManager für Nachrichten des NetworkManager-Programms und /var/log/acpid für Nachrichten des ACPI-Daemons. Das Mailsystem schreibt sein Protokoll sowohl nach /var/log/mail (alle Nachrichten) sowie außerdem in die Dateien mail.info, mail.warn und mail.err (letztere für die Prioritäten err und crit), das Newssystem schreibt sein Protokoll hingegen nach news/news.notice, news/news.err und news/news.crit (je nach Priorität) – eine Gesamtprotokolldatei gibt es bei News nicht. (Wenn Sie das inkonsistent und verwirrend finden, sind Sie nicht alleine.)


Einige Protokolldateien enthalten auch Meldungen, die die Privatsphäre der Benutzer betreffen, sie sollten deshalb nur für root lesbar sein. In den meisten Fällen sind die Distributionen lieber vorsichtig und behandeln die Zugriffsrechte auf Protokolldateien eher restriktiv.

Protokolldateien anschauen

Die vom syslogd erzeugten Logdateien können Sie mit less anschauen. Bei langen Dateien bietet sich tail an (evtl. mit der Option -f). Außerdem gibt es spezielle Programme zum Beobachten von Logdateien, die bekanntesten sind logsurfer und xlogmaster.

Meldungen

Die von syslogd protokollierten Meldungen enthalten normalerweise das Datum, den Rechnernamen, einen Hinweis auf den Prozess oder die Komponente, die die Meldung verursacht hat, sowie die Meldung selbst. Typische Meldungen sehen beispielsweise so aus:

```
Mar 31 09:56:09 red modprobe: modprobe: Can't locate ...
Mar 31 11:10:08 red su: (to root) user1 on /dev/pts/2
Mar 31 11:10:08 red su: pam-unix2: session started for ...
```

Eine zu groß gewordene Logdatei können Sie mit rm löschen oder durch Umbenennen mit der Erweiterung .old erst einmal sichern. Beim nächsten Neustart von syslogd wird gegebenenfalls eine neue Protokolldatei angelegt. Es geht aber auch komfortabler, wie der nächste Abschnitt zeigt.

1.4 Protokoll des Systemkerns

Der Kernel schickt seine Nachrichten nicht an den syslogd, sondern stellt sie in einen internen »Ringpuffer«. Von dort können sie auf verschiedene Weise ausgelesen werden – über einen speziellen Systemaufruf oder die »Datei« /proc/kmsg. Traditionell kümmert sich ein Programm namens klogd darum, /proc/kmsg zu lesen und die Nachrichten an den syslogd weiterzuleiten.

Der rsyslog kommt ohne ein separates `klogd`-Programm aus, weil er sich um Protokollnachrichten des Kernels direkt selber kümmern kann. Wenn Sie auf Ihrem System also keinen `klogd` finden können, dann könnte das daran liegen, dass es rsyslog benutzt.

Beim Systemstart stehen `syslogd` und gegebenenfalls `klogd` nicht sofort zur Verfügung – sie müssen ja als Programme gestartet werden und können so die Meldungen des Kernels beim Start nicht direkt verarbeiten. Das Kommando `dmesg` macht es möglich, rückwirkend auf den Inhalt des Puffers zuzugreifen und das Startprotokoll einzusehen. Mit einem Kommando wie

```
# dmesg >boot.msg
```

können Sie die Nachrichten in eine Datei schreiben und sie einem Kernelentwickler zuschicken.

Mit dem `dmesg`-Kommando können Sie den Kernel-Ringpuffer auch löschen (Option `-c`) und die Priorität setzen, ab der Nachrichten direkt auf die Konsole ausgegeben werden (Option `-n`). Kernel-Nachrichten haben Prioritäten von 0 bis 7, die mit den `syslogd`-Prioritäten `emerg` bis `debug` korrespondieren. Mit dem Kommando

```
# dmesg -n 1
```

werden zum Beispiel nur noch `emerg`-Nachrichten direkt auf die Konsole ausgegeben. Über `/proc/kmsg` werden auf jeden Fall immer alle Nachrichten protokolliert – hier ist es die Aufgabe von nachgeschalteter Software wie `syslogd`, die gewünschten Nachrichten auszusortieren.

Übungen

1.5 [2] Was verrät die Ausgabe von `dmesg` Ihnen über die Hardware in Ihrem Rechner?

1.5 Erweiterte Möglichkeiten: Rsyslog

Rsyslog von Rainer Gerhards hat bei den meisten gängigen Linux-Distributionen inzwischen den traditionellen BSD-`syslogd` ersetzt. Das Ziel von rsyslog ist neben größerer Effizienz auch die Unterstützung von diversen Quellen und Senken für Protokollnachrichten. Zum Beispiel schreibt er Nachrichten nicht nur in Textdateien und auf Terminals, sondern auch in eine breite Auswahl von Datenbanken.

Laut seiner eigenen Webseite steht »rsyslog« für *rocket-fast syslog*. Natürlich darf man auf solche Selbstanpreisungen nicht unbedingt viel geben, aber in diesem Fall ist das Eigenlob nicht völlig unangebracht.

Die grundlegenden Ideen hinter rsyslog sind in etwa wie folgt:

- »Quellen« reichen Nachrichten an »Regelsätze« weiter. Es gibt standardmäßig einen eingebauten Regelsatz (`RSYSLOG_DefaultRuleset`), aber Sie als Benutzer können weitere definieren.
- Jeder Regelsatz enthält beliebig viele Regeln (auch keine, selbst wenn das keinen großen Sinn ergibt).
- Eine Regel besteht aus einem »Filter« und einer »Aktionsliste«. Filter treffen Ja-Nein-Entscheidungen darüber, ob die zugehörige Aktionsliste ausgeführt wird.

- Für jede Nachricht werden alle Regeln im Regelsatz in ihrer Reihenfolge von der ersten zur letzten ausgeführt (und keine anderen). Es werden immer alle Regeln ausgeführt, egal wie die Filter-Entscheidungen ausfallen, wobei es auch die Aktion »Bearbeitung abbrechen« gibt.
- Eine Aktionsliste kann viele Aktionen enthalten (mindestens eine). Innerhalb einer Aktionsliste sind keine weiteren Filter möglich. Die Aktionen bestimmen, was mit passenden Protokollnachrichten geschieht.
- Das genaue Aussehen von Protokollnachrichten in der Ausgabe läßt sich über »Templates« steuern.

Die Konfiguration von rsyslog steht in der Datei `/etc/rsyslog.conf`. Innerhalb dieser Datei können Sie parallel drei verschiedene Arten von Konfigurationseinstellungen verwenden:

- Die traditionelle Syntax von `/etc/syslog.conf` (»sysklogd«).
- Eine veraltete Syntax von rsyslog (»legacy rsyslog«). Diese erkennen Sie daran, dass Kommandos mit einem Dollarzeichen (`$`) anfangen.
- Die aktuelle Syntax von rsyslog (»RainerScript«). Sie ist für komplexe Situationen am besten geeignet.

Die ersten beiden Geschmacksrichtungen sind zeilenbasiert. In der aktuellen Syntax sind Zeilenumbrüche irrelevant.

Für sehr einfache Anwendungen können – und sollten! – Sie die sysklogd-Syntax verwenden (wie wir sie in den vorigen Abschnitten besprochen haben). Wenn Sie Konfigurationsparameter einstellen oder aufwendigen Kontrollfluss ausdrücken wollen, ist RainerScript empfehlenswerter. Um die veraltete rsyslog-Syntax sollten Sie einen Bogen machen (auch wenn diverse Linux-Distributionen das in ihren Voreinstellungen nicht tun), bis darauf, dass manche Eigenschaften von rsyslog nur in dieser Syntax zugänglich sind.

Wie üblich werden Leerzeilen und Kommentarzeilen ignoriert. Als Kommentarzeilen gelten sowohl Zeilen (oder Teile von Zeilen), die mit `#` anfangen (der Kommentar geht dann bis zum jeweiligen Zeilenende) als auch C-artige Kommentare, die von einem `/*` ohne Ansehen von Zeilenumbrüchen bis zu einem `*/` reichen.

C-artige Kommentare dürfen Sie nicht verschachteln[1], aber innerhalb von C-artigen Kommentaren können `#`-Kommentare auftauchen. Damit sind C-artige Kommentare vor allem dazu nützlich, große Stücke einer Konfigurationsdatei »auszukommentieren«, also für rsyslog unsichtbar zu machen.

Rsyslog bietet diverse Möglichkeiten, die über die des BSD-`syslogd` hinausgehen. Sie können zum Beispiel erweiterte Filterausdrücke für Nachrichten verwenden:

```
:msg, contains, "FOO"    /var/log/foo.log
```

Die erweiterten Filterausdrücke bestehen immer aus einem Doppelpunkt am linken Rand, einer »Property«, die rsyslog der Nachricht entnimmt, einem Filteroperator (hier `contains`) und einem Suchbegriff. In unserem Beispiel werden alle Protokollnachrichten, deren Text die Zeichenkette `FOO` enthält, in die Datei `/var/log/foo.log` geschrieben.

Zu den »Properties«, die Sie verwenden können, gehören außer `msg` (der Protokollnachricht im eigentlichen Sinne) zum Beispiel `hostname` (der Name des Rechners, von dem die Nachricht ausging), `fromhost` (der Name des Rechners, von dem rsyslog die Nachricht bekommen hat), `pri` (die Kategorie und

[1] Das dürfen Sie in C auch nicht, also sollte es Sie nicht übermäßig belästigen.

Priorität der Nachricht als undekodierte Zahl), `pri-text` (Kategorie und Priorität als Text, mit der Zahl dahinter, etwa »`local0.err<133>`«), `syslogfacility` und `syslogseverity` sowie `syslogfacility-text` und `syslogseverity-text` zum gezielten Zugriff auf Kategorie und Priorität, `timegenerated` (wann die Nachricht empfangen wurde) oder `inputname` (der rsyslog-Modulname der Quelle der Nachricht). Es gibt noch einige mehr; schauen Sie sich die rsyslog-Dokumentation an.

Die erlaubten Vergleichsoperationen sind `contains`, `isequal`, `startswith`, `regex` und `ereregex`. Diese sprechen für sich selbst, bis auf die beiden letzten – `regex` betrachtet seinen Parameter als einfachen und `ereregex` als »erweiterten« regulären Ausdruck gemäß POSIX. Groß- und Kleinschreibung wird bei allen Vergleichsoperationen beachtet.

Der `startswith`-Vergleich ist nützlich, weil er deutlich effizienzer ist als ein am Anfang der Nachricht verankerter regulärer Ausdruck (jedenfalls solange Sie nur nach einer konstanten Zeichenkette suchen). Allerdings sollten Sie aufpassen, denn was Sie für den Anfang der Nachricht halten und was rsyslog darüber denkt, können durchaus zwei Paar Schuhe sein. Wenn rsyslog über den syslog-Dienst eine Nachricht empfängt, sieht diese zum Beispiel aus wie

```
<131>Jul 22 14:25:50 root: error found
```

Für rsyslog beginnt `msg` aber nicht, wie man naiverweise denken könnte, mit dem `e` von `error`, sondern mit dem Leerzeichen davor. Wenn Sie also nach Nachrichten suchen, die mit `error` anfangen, sollten Sie

```
:msg, startswith, " error"   /var/log/error.log
```

schreiben.

Es gibt eine nette Erweiterung auf der »Aktionsseite« einfacher Regeln: Sie hatten ja schon beim traditionellen `syslogd` gesehen, dass Sie mit einem Eintrag wie

```
local0.*       @red.example.com
```

Protokollnachrichten über das (UDP-basierte) syslog-Protokoll an einen entfernten Rechner weiterreichen können. Bei rsyslog können Sie auch

```
local0.*       @@red.example.com
```

schreiben, damit die Übertragung per TCP erfolgt. Das ist potentiell zuverlässiger, vor allem, wenn Firewalls im Spiel sind.

Am anderen Ende der TCP-Verbindung muss natürlich ein geeignet konfigurierter rsyslog lauschen. Das können Sie zum Beispiel über

```
module(load="imtcp" MaxSessions="500")
input(type="imtcp" port="514")
```

ermöglichen. In der veralteten Syntax ist das

```
$ModLoad imtcp
$InputTCPMaxSessions 500
$InputTCPServerRun 514
```

Achten Sie darauf, dass nur der UDP-Port 514 offiziell für das syslog-Protokoll reserviert ist. Der TCP-Port 514 ist eigentlich anderweitig vergeben[2]. Sie können gegebenenfalls einen anderen Port festlegen:

```
local0.*       @@red.example.com:10514
```

(und das funktioniert, wenn nötig, auch für UDP). Die serverseitig nötigen Änderungen finden Sie sicher leicht selber raus.

Die nächste Komplexitätsstufe sind Filter auf der Basis von Ausdrücken, die beliebige Boolesche, arithmetische und Zeichenketten-Operationen umfassen können. Diese beginnen immer mit einem `if` ganz am linken Rand einer neuen Zeile:

```
if $syslogfacility-text == "local0" and $msg startswith " FOO" ▷
 ◁ and ($msg contains "BAR" or $msg contains "BAZ") ▷
 ◁ then /var/log/foo.log
```

(in Ihrer Datei sollte das alles in einer Zeile stehen). Mit dieser Regel werden Nachrichten der Kategorie `local0` in `/var/log/foo.log` geschrieben, die mit `FOO` anfangen und außerdem `BAR` oder `BAZ` (oder beides) enthalten. (Achten Sie auf die Dollarzeichen vor den Namen der Properties.)

Rsyslog unterstützt eine große Anzahl von Modulen, die bestimmen, was mit Protokollnachrichten passieren soll. Sie könnten beispielsweise wichtige Nachrichten per E-Mail verschicken. Dazu können Sie in `/etc/rsyslog.conf` etwas schreiben wie

```
module(load="ommail")
template(name="mailBody" type="string" string="ALERT\\r\\n%msg%")
if $msg contains "disk error" then {
    action(type="ommail" server="mail.example.com" port="25"
           mailfrom="rsyslog@example.com" mailto="admins@example.com"
           subject.text="disk error detected"
           body.enable="on" template="mailBody"
           action.execonlyonceeveryinterval="3600")
}
```

Wenn Sie eine ältere Version von rsyslog haben (vor 8.5.0), müssen Sie die veraltete Syntax nehmen, um das `ommail`-Modul zu konfigurieren. Das sieht dann zum Beispiel so aus:

```
$ModLoad ommail
$ActionMailSMTPServer mail.example.com
$ActionMailFrom rsyslog@example.com
$ActionMailTo admins@example.com
$template mailSubject,"disk error detected"
$template mailBody,"ALERT\\r\\n%msg%"
$ActionMailSubject mailSubject
$ActionExecOnlyOnceEveryInterval 3600
if $msg contains "disk error" then :ommail:;mailBody
$ActionExecOnlyOnceEveryInterval 0q
```

Die SMTP-Implementierung von rsyslog ist einigermaßen primitiv, da sie keine Verschlüsselung oder Authentisierung zuläßt. Das heißt, der Mailserver, den Sie in der rsyslog-Konfiguration angeben, muss in der Lage sein, auch ohne Verschlüsselung oder Authentisierung Mail von rsyslog anzunehmen.

[2] … auch wenn sich heutzutage niemand mehr für den Remote-Shell-Dienst interessiert. Niemand Vernünftiges jedenfalls.

Ach ja: Rsyslog kümmert sich direkt um Protokollnachrichten aus dem Linux-Kern. Dazu müssen Sie lediglich das `imklog`-Eingabemodul aktivieren:

```
module(load="imklog")
```

oder (veraltete Syntax)

```
$ModLoad imklog
```

Ein separater `klogd`-Prozess ist nicht erforderlich.

Detaillierte Informationen über rsyslog finden Sie zum Beispiel in der Online-Dokumentation [rsyslog].

Übungen

1.6 [!3] (Falls Ihre Distribution nicht sowieso schon rsyslog benutzt.) Installieren Sie rsyslog und erstellen Sie eine Konfiguration, die Ihrer bisherigen `syslogd`-Konfiguration möglichst nahe kommt. Testen Sie sie zum Beispiel mit `logger`. Wo sehen Sie Verbesserungsmöglichkeiten?

1.7 [2] PAM, das Anmelde- und Authentisierungssystem, protokolliert An- und Abmeldevorgänge in der folgenden Form:

```
kdm: :0[5244]: (pam_unix) session opened for user hugo by (uid=0)
◁◁◁◁◁
kdm: :0[5244]: (pam_unix) session closed for user hugo
```

Konfigurieren Sie rsyslog so, dass jedesmal, wenn sich ein bestimmter Benutzer (etwa Sie) an- oder abmeldet, eine Meldung auf dem Terminal des Systemverwalters (`root`) erscheint, falls dieser angemeldet ist. (*Tipp:* Die PAM-Nachrichten erscheinen in der Kategorie `authpriv`.)

1.8 [3] (Arbeiten Sie für diese Aufgabe gegebenenfalls mit einem anderen Kursteilnehmer zusammen.) Konfigurieren Sie rsyslog so, dass alle Protokollnachrichten von einem Rechner über eine TCP-Verbindung an einen anderen Rechner weitergeleitet werden. Testen Sie die Verbindung mit `logger`.

1.6 Die »nächste Generation«: Syslog-NG

Syslog-NG (»NG« für »New Generation«) ist eine kompatible, aber erweiterte Neuimplementierung eines Syslog-Daemons von Balazs Scheidler. Einige der Hauptvorteile von Syslog-NG gegenüber dem herkömmlichen `syslogd` sind: Hauptvorteile

- Filterung von Nachrichten auf Inhaltsbasis (nicht nur Kategorien und Prioritäten)
- Verknüpfung mehrerer Filter möglich
- Ausgefeilteres Ein-/Ausgabesystem, inklusive Weiterleitung über TCP und in Unterprozesse

Das Programm selbst heißt `syslog-ng`.

Für die Syslog-Clients ändert sich zunächst nichts; Sie können einen `syslogd` also problemlos durch Syslog-NG ersetzen.

Sie finden Informationen über Syslog-NG in den Handbuchseiten sowie unter [syslog-ng]. Dort gibt es Dokumentation und auch eine sehr nützliche FAQ-Sammlung.

Konfigurationsdatei Syslog-NG liest seine Konfiguration aus einer Datei, normalerweise `/etc/syslog-ng/syslog-ng.conf`. Im Gegensatz zum `syslogd` unterscheidet Syslog-NG verschiedene Arten von Einträgen in seiner Konfigurationsdatei:

Arten von Einträgen

Globale Optionen Diese Einstellungen gelten für alle Nachrichtenquellen oder den Syslog-NG-Daemon selbst.

Quellen von Nachrichten Syslog-NG kann Nachrichten auf diverse Weisen lesen: per Unix-Domain-Socket und UDP wie `syslogd`, aber auch zum Beispiel aus Dateien, FIFOs oder TCP-Sockets. Jede Nachrichtenquelle bekommt einen Namen.

Filter Filter sind Boolesche Ausdrücke auf der Basis interner Funktionen, die sich zum Beispiel auf die Herkunft, Kategorie, Priorität oder auch den textuellen Inhalt einer Protokollnachricht beziehen können. Auch Filter werden benannt.

Senken für Nachrichten Syslog-NG erlaubt alle Protokollierungsmethoden von `syslogd` und noch ein paar mehr.

Protokoll-Pfade Ein »Protokoll-Pfad« verbindet eine oder mehrere Nachrichtenquellen, Filter und Senken zu einem Ganzen: Treffen aus den Nachrichtenquellen Protokollmeldungen ein, auf die der oder die Filter passen, werden sie an die angegebene(n) Senke(n) weitergeleitet. Die Konfigurationsdatei besteht letzten Endes aus einer Reihe solcher Protokoll-Pfade.

Optionen Sie können diverse »globale« Optionen angeben, die das allgemeine Verhalten von Syslog-NG beeinflussen oder Standardwerte für einzelne Nachrichtenquellen oder -senken vorgeben (spezielle Angaben bei den Quellen oder Senken haben Vorrang). Eine komplette Liste steht in der Syslog-NG-Dokumentation. Zu den allgemeinen Optionen gehören diverse Einstellungen für den Umgang mit DNS und das Weiterreichen oder Umschreiben der Absender-Rechnernamen von Nachrichten.

Wenn der Syslog-NG auf Recher *A* eine Nachricht von Rechner *B* empfängt, prüft er die Option `keep_hostnames()`. Hat diese den Wert `yes`, wird *B* als Rechnername für das Protokoll beibehalten. Sonst kommt es auf die Einstellung `chain_hostnames()` an; ist diese `no`, wird *A* als Rechnername ins Protokoll geschrieben, ist sie `yes`, protokolliert Syslog-NG *B*/*A*. Dies ist insbesondere wichtig, wenn das Protokoll an noch einen anderen Rechner weitergeleitet wird.

Quellen von Nachrichten Nachrichtenquellen definieren Sie bei Syslog-NG über das Schlüsselwort `source`. Eine Nachrichtenquelle fasst einen oder mehrere »Treiber« zusammen. Um dasselbe zu erreichen wie ein »normaler« `syslogd`, würden Sie zum Beispiel die Zeile

```
source src { unix-stream("/dev/log"); internal(); };
```

in Ihre Konfiguration aufnehmen; dies weist Syslog-NG an, auf dem Unix-Domain-Socket `/dev/log` zu lauschen. `internal()` bezieht sich auf Nachrichten, die der Syslog-NG selbst erzeugt.

Eine Syslog-NG-Nachrichtenquelle, die der `syslogd`-Option `-r` entspricht, sähe zum Beispiel so aus:

```
source s_remote { udp(ip(0.0.0.0) port(514)); };
```

Da das der Standardwert ist, reicht auch ein

Tabelle 1.3: Filterfunktionen für Syslog-NG

Syntax	Beschreibung
`facility(`⟨*Kategorie*⟩`[,`⟨*Kategorie*⟩ …`])`	Passt auf Nachrichten mit einer der angegebenen Kategorien
`level(`⟨*Priorität*⟩`[,`⟨*Priorität*⟩ …`])`	Passt auf Nachrichten mit einer der angegebenen Prioritäten
`priority(`⟨*Priorität*⟩`[,`⟨*Priorität*⟩ …`])`	Äquivalent zu `level()`
`program(`⟨*reg. Ausdruck*⟩`)`	Passt auf Nachrichten, bei denen der Name des Absenderprogramms auf den ⟨*reg. Ausdruck*⟩ passt
`host(`⟨*reg. Ausdruck*⟩`)`	Passt auf Nachrichten, deren Absenderrechner auf den ⟨*reg. Ausdruck*⟩ passt
`match(`⟨*reg. Ausdruck*⟩`)`	Passt auf Nachrichten, die selbst auf den ⟨*reg. Ausdruck*⟩ passen
`filter(`⟨*Name*⟩`)`	Ruft eine andere Filterregel auf und liefert ihren Wert zurück
`netmask(`⟨*IP-Adresse*⟩`/`⟨*Netzmaske*⟩`)`	Prüft, ob die IP-Adresse im angegebenen Netz liegt

```
source s_remote { udp(); };
```

Mit `ip()` können Sie den Syslog-NG nur auf bestimmten lokalen IP-Adressen lauschen lassen. Mit `syslogd` geht das nicht.

Die folgende Quellenangabe erlaubt es Syslog-NG, das `klogd`-Programm zu ersetzen:

```
source kmsg { file("/proc/kmsg" log_prefix("kernel: ")); };
```

Alle Nachrichtenquellen unterstützen einen weiteren Parameter, `log_msg_size()`, der die maximale Länge einer Nachricht in Bytes angibt.

Filter Filter dienen dazu, Protokollnachrichten auszusortieren oder auf verschiedene Senken zu verteilen. Sie beruhen auf internen Funktionen, die bestimmte Aspekte der Nachrichten betrachten; diese Funktionen können über die logischen Funktionen `and`, `or` und `not` miteinander verknüpft werden. Eine Liste der möglichen Funktionen finden Sie in Tabelle 1.3.

Beispielsweise könnten Sie einen Filter definieren, der auf alle Nachrichten vom Rechner `green` passt, die den Text `error` enthalten:

```
filter f_green { host("green") and match("error"); };
```

Bei der Funktion `level()` (bzw. `priority()`) können Sie entweder eine oder mehrere durch Kommas getrennte Prioritäten oder aber einen Bereich von Prioritäten in der Form »`warn .. emerg`« angeben.

Senken für Nachrichten Genau wie Quellen bestehen auch Senken aus verschiedenen »Treibern« für Protokollierungsmethoden. Zum Beispiel könnten Sie Nachrichten in eine Datei schreiben:

```
destination d_file { file("/var/log/messages"); };
```

Sie können auch eine »Schablone« vorgeben, die beschreibt, in welchem Format die Nachricht an die betreffende Senke geschrieben werden soll. Dabei können Sie auf »Makros« zurückgreifen, die diverse Bestandteile der Nachricht zugänglich machen. Zum Beispiel:

```
destination d_file {
    file("/var/log/$YEAR.$MONTH.$DAY/messages"
         template("$HOUR:$MIN:$SEC $TZ $HOST [$LEVEL] $MSG\n")
         template_escape(no)
         create_dirs(yes)
    );
};
```

Die Makros `$YEAR`, `$MONTH` usw. werden dabei durch ihre naheliegenden Werte ersetzt. `$TZ` ist die aktuelle Zeitzone, `$LEVEL` die Priorität der Nachricht und `$MSG` die Nachricht selbst (inklusive der Prozess-ID des Absenders). Eine komplette Liste der Makros finden Sie in der Dokumentation zu Syslog-NG.

Der Parameter `template_escape()` steuert, ob die Anführungszeichen (»'« und »"«) in der Ausgabe »versteckt« werden sollen. Dies ist wichtig, falls Sie Protokollnachrichten zum Beispiel an einen SQL-Server verfüttern wollen.

Im Gegensatz zum `syslogd` erlaubt Syslog-NG die Weiterleitung von Nachrichten per TCP. Dies ist nicht nur bequemer, wenn Firewalls im Spiel sind, sondern stellt auch sicher, dass keine Protokollnachrichten verloren gehen (was bei UDP passieren könnte). Eine TCP-Weiterleitung könnten Sie zum Beispiel so definieren:

```
destination d_tcp { tcp("10.11.12.13" port(514); localport(514)); };
```

Sehr nützlich ist auch die Weiterleitung in Programme mit `program()`. Syslog-NG startet das Programm, wenn er selbst hochfährt, und lässt es weiterlaufen bis zu seinem eigenen Ende oder bis er ein `SIGHUP` empfängt. Dies dient nicht nur der Effizienz, sondern ist eine Vorsichtsmaßnahme gegen Denial-of-Service-Angriffe – wenn für jede Meldung ein Prozess gestartet würde, könnte ein Angreifer die Protokollierung außer Gefecht setzen, indem er jede Menge passende Protokollnachrichten schickt. (Andere Protokollnachrichten, die über seine Machenschaften Aufschluss geben würden, fallen dann vielleicht herunter.)

Protokoll-Pfade Protokoll-Pfade dienen dazu, Quellen, Filter und Senken zusammenzubringen und tatsächlich Nachrichten auszuwerten. Sie beginnen immer mit dem Schlüsselwort `log`. Hier sind ein paar Beispiele für Protokoll-Pfade auf der Basis von Regeln, die Sie schon aus der Diskussion von `/etc/syslog.conf` kennen:

```
# Vorbedingungen
source s_all { internal(); unix-stream("/dev/log"); };
filter f_auth { facility(auth, authpriv); };
destination df_auth { file("/var/log/auth.log"); };

# auth,authpriv.*   /var/log/auth.log
log {
        source(s_all);
        filter(f_auth);
        destination(df_auth);
};
```

Bei dieser Regel werden alle Nachrichten, die mit Authentisierung zu tun haben, in die Datei `/var/log/auth.log` geschrieben. Natürlich ist das mit dem `syslogd` in einer Zeile zu erledigen …

Hier ist ein etwas komplexeres Beispiel:

```
# kern.warn;*.err;authpriv.none    /dev/tty10
filter f_nearly_all {
    (facility(kern) and priority(warn .. emerg))
    or (not facility(authpriv,kern));
};
destination df_tty { file("/dev/tty10"); };
log {
        source(s_all);
        filter(f_nearly_all);
        destination(df_tty);
};
```

Auch hier ist die Schreibweise des `syslogd` etwas kompakter, dafür haben Sie hier eher die Chance, nachzuvollziehen, was passiert.

Jede Nachricht durchläuft alle Protokoll-Pfade und wird auch von allen passenden protokolliert (dieses Verhalten entspricht dem von `syslogd`). Wenn Sie wollen, dass eine Nachricht, die einen bestimmten Protokoll-Pfad durchlaufen hat, nicht mehr weiter betrachtet wird, können Sie diesem Pfad die Option `flags(final)` hinzufügen.

`flags(final)` bedeutet nicht, dass die Nachricht nur einmal protokolliert wird; sie könnte vor dem betreffenden Pfad noch von anderen Pfaden protokolliert worden sein.

Mit `flags(fallback)` können Sie einen Pfad zum »Ersatzpfad« erklären. Dann wird er nur für Nachrichten betrachtet, auf die keine nicht mit `flags(fallback)` gekennzeichneten Pfade gepasst haben.

Übungen

1.9 [!3] Installieren Sie Syslog-NG und erstellen Sie eine Konfiguration, die Ihrer bisherigen `syslogd`-Konfiguration möglichst nahe kommt. Testen Sie sie zum Beispiel mit `logger`. Wo sehen Sie Verbesserungsmöglichkeiten?

1.10 [2] PAM, das Anmelde- und Authentisierungssystem, protokolliert An- und Abmeldevorgänge in der folgenden Form:

```
kdm: :0[5244]: (pam_unix) session opened for user hugo by (uid=0)
◁◁◁◁◁
kdm: :0[5244]: (pam_unix) session closed for user hugo
```

Konfigurieren Sie Syslog-NG so, dass jedesmal, wenn sich ein bestimmter Benutzer (etwa Sie) an- oder abmeldet, eine Meldung auf dem Terminal des Systemverwalters (`root`) erscheint, falls dieser angemeldet ist. (*Tipp:* Die PAM-Nachrichten erscheinen in der Kategorie `authpriv`.)

1.11 [3] (Arbeiten Sie für diese Aufgabe gegebenenfalls mit einem anderen Kursteilnehmer zusammen.) Konfigurieren Sie Syslog-NG so, dass alle Protokollnachrichten von einem Rechner über eine TCP-Verbindung an einen anderen Rechner weitergeleitet werden. Testen Sie die Verbindung mit `logger`. Experimentieren Sie mit verschiedenen Einstellungen für `keep_hostnames()` und `chain_hostnames()`.

```
/var/log/syslog
{
    rotate 7
    daily
    missingok
    notifempty
    delaycompress
    compress
    postrotate
        invoke-rc.d rsyslog rotate >/dev/null
    endscript
}
```

Bild 1.1: Beispiel-Konfiguration für `logrotate` (Debian GNU/Linux 8.0)

1.7 Das Programm `logrotate`

Je nach der Anzahl der Benutzer und der Anzahl und Art der laufenden Dienste können die Protokolldateien ziemlich schnell ziemlich groß werden. Um ein »Zumüllen« des Systems zu verhindern, sollten Sie einerseits zumindest auf vielbeschäftigten Serversystemen versuchen, die entsprechenden Verzeichnisse (z. B. `/var/log` oder `/var`) auf eine eigene Partition zu legen. Andererseits gibt es Software, die die Protokolldateien regelmäßig nach verschiedenen Kriterien, etwa der Größe, überwacht, kürzt und alte Protokolle löscht oder archiviert. Dieser Vorgang heißt »Rotieren«, und ein solches Programm ist `logrotate`.

`logrotate` ist kein Daemon, sondern wird zum Beispiel einmal täglich über `cron` – oder einen ähnlichen Dienst – gestartet.

`logrotate` weigert sich, eine Protokolldatei mehr als einmal am Tag zu modifizieren, es sei denn, die Entscheidung hängt von der Größe der Protokolldatei ab, Sie verwenden das Kriterium `hourly`, oder beim Aufruf wurde die Option `--force` (kurz `-f`) angegeben.

`/etc/logrotate.conf` `/etc/logrotate.d`

`logrotate` wird nach Konvention über die Datei `/etc/logrotate.conf` und über die Dateien in `/etc/logrotate.d` konfiguriert. Die Datei `/etc/logrotate.conf` setzt allgemeine Parameter, die aber von den Dateien in `/etc/logrotate.d` wieder überschrieben werden können. In `/etc/logrotate.conf` steht unter anderem der Parameter `include /etc/logrotate.d`, der bewirkt, dass die dort befindlichen Dateien an der entsprechenden Stelle als Teil der Datei `/etc/logrotate.conf` integriert werden.

Prinzipiell liest `logrotate` alle auf der Kommandozeile übergebenen Dateinamen als Konfigurationsdateien ein, wobei die Inhalte später genannter Dateien die von weiter vorne genannten überschreiben. Die Sache mit `/etc/logrotate.conf` ist nur eine (sinnvolle) Vereinbarung, die durch einen entsprechenden Aufruf von `logrotate` in `/etc/cron.daily/logrotate` (oder Äquivalent) realisiert wird.

Wir erwähnen das hier, weil Ihnen das grundsätzlich die Möglichkeit eröffnet, ohne große Mühe separate `logrotate`-Läufe für Protokolldateien zu machen, die nicht Bestandteil der regulären Konfiguration sind. Wenn Sie zum Beispiel eine extrem schnell wachsende Protokolldatei etwa eines populären Web-Servers haben, können Sie diese über eine eigene Instanz von `logrotate` verwalten, die öfter als einmal täglich läuft.

`logrotate` überwacht alle Dateien, die ihm über die genannten Konfigurationsdateien mitgeteilt werden, also nicht nur die Ausgabe von `syslogd`. Bild 1.1 zeigt als Beispiel einen Ausschnitt aus der Konfigurationsdatei für rsyslog von Debian GNU/Linux 8.

Die erste Zeile des Beispiels gibt an, für welche Dateien die Konfiguration gilt (hier /var/log/syslog). Sie können mehrere Dateien aufzählen oder auch Shell-Suchmuster verwenden. Anschließend steht in geschweiften Klammern ein Block von Direktiven, die beschreiben, wie logrotate mit den benannten Dateien umgehen soll.

Typischerweise finden sich in /etc/logrotate.conf Direktiven, die außerhalb eines Blocks stehen. Diese Direktiven dienen als Voreinstellungen, die für alle Protokolldateien in der Konfiguration gelten, sofern in deren Blöcken nicht etwas Spezifischeres verfügt wird.

alte Versionen

»rotate 7« heißt, dass maximal sieben alte Versionen jeder Protokolldatei aufgehoben werden. Wenn dieses Maximum erreicht ist, wird die älteste Version der Protokolldatei gelöscht.

Wenn Sie mit mail eine Adresse angeben, werden Dateien nicht gelöscht, sondern an die betreffende Adresse geschickt.

»rotate 0« wirft »rotierte« Protokolldateien sofort weg, ohne sie zu aufzuheben.

Die rotierten Dateien werden einfach fortlaufend durchnummeriert, das heißt, wenn die aktuelle Version der Datei /var/log/syslog heißt, dann heißt die unmittelbar vorhergehende Version /var/log/syslog.1, die Version unmittelbar davor /var/log/syslog.2 und so weiter.

Sie können statt einer fortlaufenden Nummer auch das Datum als Dateiendung benutzen. Das heißt, wenn heute der 20. Juli 2015 ist und Ihr logrotate-Lauf täglich in den frühen Morgenstunden stattfindet, dann heißt die unmittelbar vorhergehende Version der Datei nicht /var/log/syslog.1, sondern /var/log/syslog-20150720, die Version unmittelbar davor entsprechend /var/log/syslog-20150719 und so fort. Um das zu erreichen, müssen Sie die Direktive »dateext« angeben.

Mit »dateformat« können Sie bestimmen, wie genau die Datums-Dateiendung aussehen kann. Sie müssen dazu eine Zeichenkette übergeben, die die Schlüssel %Y, %m, %d und %s enthalten darf. Sie stehen respektive für das (vierstellige) Jahr, Kalendermonat und Kalendertag (jeweils zweistellig und gegebenfalls mit führender Null) und die Sekunden seit dem 1. Januar 1970, 0 Uhr UTC. Die Standardvorgabe ist – wie Sie aus dem vorigen Absatz schließen können – »-%Y%m%d«.

Wenn Sie dateformat benutzen, sollten Sie darauf achten, dass logrotate beim Rotieren die Dateinamen lexikografisch sortiert, um zu entscheiden, was die älteste Datei ist. Mit »-%Y%m%d« klappt das, aber mit »-%d%m%Y« nicht.

Zeiträume

»daily« heißt, dass Protokolldateien täglich rotiert werden sollen. Im Zusammenspiel mit »rotate 7« bedeutet das, dass Sie immer Zugriff auf die Protokolle der letzten Woche haben.

Es gibt auch noch weekly, monthly and yearly. Mit weekly wird die Datei rotiert, wenn der aktuelle Wochentag vor dem Wochentag der letzten Rotation liegt oder mehr als eine Woche seit der letzten Rotation vergangen ist (unter dem Strich heißt das, dass am ersten Wochentag rotiert wird, wobei das gemäß US-amerikanischer Gepflogenheit der Sonntag ist). Mit monthly wird die Datei beim ersten Lauf von logrotate in jedem Monat rotiert (normalerweise am Monatsersten). Mit yearly erfolgt die Rotation beim ersten Lauf von logrotate in jedem Jahr. Theoretisch rotiert hourly die Protokolldatei in stündlichem Rhythmus, aber da logrotate normalerweise nur einmal am Tag aufgerufen wird, müssen Sie gegebenenfalls selber dafür sorgen, dass es tatsächlich so oft läuft.

Ein alternatives Kriterium ist »size«. Damit können Sie eine Protokolldatei rotieren, wenn sie eine bestimmte Größe überschritten haben. Die Dateigröße wird als Parameter angegeben – ohne Einheit wird sie als Bytes interpretiert, während die Einheiten k (oder K), M, G respektive für Kibibytes (2^{10} Bytes), Mebibytes (2^{20} Bytes) oder Gibibytes (2^{30} Bytes) stehen.

»size« und die zeitbasierten Kriterien schließen einander aus, das heißt, wenn Sie ein »size«-Kriterium angeben, entscheidet allein die Dateigröße über das Rotieren, egal wann die Datei zuletzt rotiert wurde.

Dateigröße und Zeit als Kriterien verbinden können Sie über die Direktiven »maxsize« und »minsize«. Mit »maxsize« können Sie eine Größe angeben, deren Überschreiten logrotate die Datei rotieren läßt, selbst wenn der nächste offizielle Zeitpunkt dafür noch nicht erreicht wurde. Bei »minsize« wird die Datei zum per Zeit-Kriterium vorgesehenen Zeitpunkt nur rotiert, wenn sie die angegebene Größe überschritten hat (kleine Dateien werden übergangen).

Fehlermeldungen

»missingok« unterdrückt Fehlermeldungen, wenn eine Protokolldatei nicht gefunden wird. (Der Standard ist »nomissingok«.) »notifempty« rotiert eine Datei nicht, wenn sie leer ist (die Vorgabe ist hier »ifempty«).

Mit »compress« können Sie verfügen, dass rotierte Versionen der Protokolldatei komprimiert werden sollen.

Standardmäßig passiert das mit gzip, wenn Sie nicht mit »compresscmd« ein anderes Kommando benennen. Mit »compressoptions« können Sie Optionen für dieses Kommando angeben (die Sie sonst auf der Kommandozeile übergeben würden). Der Standard für gzip ist »-6«.

Die Direktive »delaycompress« sorgt dafür, dass eine frisch rotierte Datei nicht unmittelbar nach dem Rotieren komprimiert wird, sondern erst im nächsten Durchgang. Während normalerweise die Folge der Dateien aussähe wie

```
/var/log/syslog  /var/log/syslog.1.gz  /var/log/syslog.2.gz  …
```

bekommen Sie mit »delaycompress« die Folge

```
/var/log/syslog  /var/log/syslog.1  /var/log/syslog.2.gz …
```

(mit anderen Worten, /var/log/syslog.1 bleibt noch unkomprimiert). Diese Einstellung brauchen Sie, wenn die Möglichkeit besteht, dass das schreibende Programm (hier rsyslog) nach dem Umbenennen (Rotieren) der Datei noch Daten an sie anhängt – das kann passieren, weil rsyslog die Protokolldatei dauerhaft offenhält und Umbenennungen für das Schreiben in die Datei irrelevant sind.

Daraus folgt, dass Sie rsyslog benachrichtigen müssen, dass es eine neue Protokolldatei gibt. Dafür ist die Direktive

```
postrotate
  invoke-rc.d rsyslog rotate >/dev/null
endscript
```

gedacht. Die Shell-Kommandos zwischen »postrotate« und »endscript« führt logrotate jedesmal aus, nachdem die Protokolldatei rotiert wurde.

Das Kommando selbst ist für uns im Grunde egal (die Idee zählt), aber was hier unter dem Strich passiert, ist, dass das Init-Skript von rsyslog aufgerufen wird und dem Programm ein SIGHUP schickt. Andere Distributionen haben dafür auch Mittel und Wege.

Das SIGHUP bewegt rsyslog dann dazu, seine Konfigurationsdatei neu einzulesen und alle Protokolldateien zu schließen und wieder zu öffnen. Da /var/log/syslog inzwischen umbenannt wurde, öffnet rsyslog unter diesem Namen eine neue Protokolldatei. – An dieser Stelle könnte logrotate die Datei /var/log/syslog.1 komprimieren, aber es hat keine Möglichkeit, zu erfahren, wann rsyslog tatsächlich mit der Datei »fertig hat« (G. Trapattoni). Darum wird das beim nächsten Rotieren der Datei nachgeholt.

Zwischen »postrotate« und »endscript« dürfen mehrere Zeilen mit Kommandos stehen. logrotate hängt sie alle aneinander und übergibt sie als Ganzes der Shell (/bin/sh) zur Ausführung. Das Kommando bekommt den Namen der Protokolldatei als Parameter übergeben; dieses steht dort wie üblich als »$1« zur Verfügung.

Die postrotate-Kommandos werden einzeln für jede Protokolldatei ausgeführt, die am Anfang des Konfigurationsblocks aufgezählt wurde. Das heißt, dass die Kommandos möglicherweise mehrmals aufgerufen werden. Mit der Direktive »sharedscripts« können Sie dafür sorgen, dass die Kommandos für alle Dateien, auf die das Suchmuster passt, insgesamt nur einmal ausgeführt werden (oder gar nicht, wenn von den Dateien keine rotiert werden musste).

Mit »create« können Sie dafür sorgen, dass unmittelbar nach dem Rotieren und vor der Ausführung der postrotate-Kommandos die Protokolldatei neu angelegt wird. Dabei wird der Name der alten Datei verwendet. Zugriffsmodus, Eigentümer und Gruppe richten sich nach den Parametern von create, es gibt die drei Möglichkeiten

`create 600 root adm`	*Zugriffsmodus, Eigentümer und Gruppe*
`create root adm`	*Nur Eigentümer und Gruppe*
`create`	*Gar nichts*

Nicht angegebene Dateieigenschaften werden von der vorigen Version der Datei übernommen.

Dies ist nur eine Auswahl der wichtigsten Konfigurationsparameter. Studieren Sie logrotate(8), um die komplette Liste zu sehen.

Übungen

1.12 [!1] Welche systemweiten Voreinstellungen gelten für logrotate bei Ihrer Distribution?

1.13 [2] Konfigurieren Sie logrotate so, dass Ihre neue Protokolldatei /var/log/test rotiert wird, sobald sie eine Größe von 100 Byte überschreitet. Es sollen 10 Backups der rotierten Dateien aufgehoben werden, außerdem sollen diese Backups komprimiert werden und einen Namen tragen, der das Datum ihrer Erstellung enthält. Testen Sie Ihre Konfiguration.

Kommandos in diesem Kapitel

dmesg Gibt den Inhalt des Kernel-Nachrichtenpuffers aus dmesg(8) 19
klogd Akzeptiert Protokollnachrichten des Systemkerns klogd(8) 14, 18
logger Macht Einträge ins Systemprotokoll logger(1) 17
logrotate Verwaltet, kürzt und „rotiert“ Protokolldateien logrotate(8) 27
logsurfer Programm zum Durchsuchen des Systemprotokolls nach wichtigen Ereignissen www.cert.dfn.de/eng/logsurf/ 18
syslog-ng Bearbeitet Systemprotokoll-Meldungen (neuer und besser) syslog-ng(8) 23
syslogd Bearbeitet Systemprotokoll-Meldungen syslogd(8) 14
tail Zeigt das Ende einer Datei an tail(1) 18
xconsole Zeigt Systemmeldungen in einem X-Fenster an xconsole(1) 14
xlogmaster X11-basiertes Systembeobachtungsprogramm xlogmaster(1), www.gnu.org/software/xlogmaster/ 18

Zusammenfassung

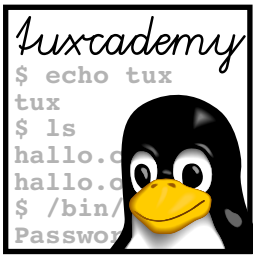
- Der syslogd kann Protokollnachrichten verschiedener Systemkomponenten annehmen, in Dateien schreiben, an Benutzer oder andere Rechner weiterreichen.
- Protokollnachrichten können aus verschiedenen Kategorien kommen und unterschiedliche Prioritäten haben.
- Mit dem Kommando logger kann man Nachrichten an den syslogd schicken.
- Protokolldateien stehen für gewöhnlich im Verzeichnis /var/log.
- Syslog-NG ist eine kompatible, aber erweiterte Neuimplementierung eines Syslog-Daemons.
- Mit logrotate können Protokolldateien verwaltet und archiviert werden.

Literaturverzeichnis

RFC3164 C. Lonvick. »The BSD syslog Protocol«, August 2001.
http://www.ietf.org/rfc/rfc3164.txt

rsyslog »Welcome to Rsyslog«. Online-Dokumentation für rsyslog 8.*x*.
http://www.rsyslog.com/doc/v8-stable/index.html

syslog-ng »syslog-ng – Log Management Software«.
http://www.balabit.com/products/syslog_ng/



2

Systemprotokollierung mit systemd und »dem Journal«

Inhalt

2.1 Grundlagen 34
2.2 Systemd und journald 35
2.3 Protokollauswertung 38

Lernziele

- Grundprinzipien von journald verstehen
- Journald konfigurieren können
- Einfache Anfragen an die Protokolldatenbank formulieren können
- Verstehen, wie journald mit Protokolldateien umgeht

Vorkenntnisse

- Grundkenntnisse der Komponenten eines Linux-Systems
- Umgang mit Konfigurationsdateien
- Kenntnisse des traditionellen Systemprotokolldiensts (Kapitel 1)
- Kenntnisse über systemd

2.1 Grundlagen

Systemd ist eine durchgreifende Erneuerung der Software, die den grundlegenden Systembetrieb eines Linux-Rechners sichert. Im engeren Sinne kümmert systemd sich um das Starten und Nachverfolgen von Diensten und die Verwaltung von Ressourcen. Allerdings enthält systemd auch einen vom traditionellen syslogd-Verfahren deutlich verschiedenen Ansatz für die Systemprotokollierung, das »Journal«, und die dafür nötigen Systemkomponenten.

Während im traditionellen Ansatz der syslog-Daemon Protokollnachrichten auf dem UDP-Port 514 oder dem Socket `/dev/log` entgegennimmt und (typischerweise) in Textdateien schreibt (oder sie an andere Rechner weitergeleitet werden, wo sie dann in Textdateien geschrieben werden), können in der systemd-Welt Hintergrunddienste ihre Protokollnachrichten einfach auf die Standardfehlerausgabe schreiben, und systemd kümmert sich darum, sie an den Protokolldienst weiterzuleiten[1]. Protokolldateien sind bei systemd keine Textdateien (wobei jede Meldung möglicherweise in mehreren Dateien abgelegt wird), sondern Meldungen werden in eine (binäre) Datenbank geschrieben, die anschließend nach diversen Kriterien durchsucht werden kann.

Es ist zum Beispiel sehr einfach möglich, alle Nachrichten anzuzeigen, die ein bestimmter Dienst in einem bestimmten durch Anfangs- und Enddatum und -uhrzeit gegebenen Zeitraum protokolliert hat. Im traditionellen System ist das ziemlich aufwendig.

Fairerweise sollten wir nochmal hervorheben, dass auch die modernen syslog-Implementierungen wie Rsyslog oder Syslog-NG prinzipiell in der Lage sind, Nachrichten in eine Datenbank zu schreiben. Allerdings sind Sie dann selbst dafür verantwortlich, ein entsprechendes Datenbankschema zu entwerfen, Rsyslog bzw. Syslog-NG passend zu konfigurieren, und Software zu entwickeln, die Ihnen bequemen Zugang zu den Protokollnachrichten verschafft. Bei systemd ist das alles »ab Werk« dabei.

Das Journal ist nicht auf textuelle Protokollnachrichten beschränkt. Zum Beispiel ist es absolut möglich, Hauptspeicherabzüge (*core dumps*) abgestürzter Programme ins Journal zu schreiben (jedenfalls solange sie nicht übergroß sind). Ob das eine uneingeschränkt geniale Idee ist, ist natürlich debattierbar, und die systemd-Entwickler haben sich auch schon eine alternative Methode überlegt.

Wie üblich kann auch das Protokollsystem von systemd mit dem traditionellen Ansatz interoperieren. Es protokolliert auf Wunsch auch Nachrichten, die über `/dev/log` oder den UDP-Port 512 eingehen, und kann Nachrichten an einen traditionellen syslog-Daemon (oder eine moderne Neuimplementierung) weiterreichen.

Dem Journal verdanken Sie übrigens auch den (extrem bequemen) Umstand, dass Sie bei »`systemctl status`« die letzten Protokollnachrichten des betreffenden Dienstes angezeigt bekommen:

```
# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since Mo 2015-07-27 13:37:22 CEST; 8h ago
 Main PID: 428 (sshd)
   CGroup: /system.slice/ssh.service
           └─428 /usr/sbin/sshd -D

Jul 27 13:37:23 blue sshd[428]: Server listening on 0.0.0.0 port 22.
Jul 27 13:37:23 blue sshd[428]: Server listening on :: port 22.
Jul 27 13:56:50 blue sshd[912]: Accepted password for hugo from ...sh2
```

[1] Systemd hat außerdem noch ein eigenes API für Protokollnachrichten.

```
Jul 27 13:56:50 blue sshd[912]: pam_unix(sshd:session): session ...=0)
Hint: Some lines were ellipsized, use -l to show in full.
```

Wie die letzte Zeile der Ausgabe andeutet, werden überlange Zeilen so abgekürzt, dass sie gerade noch auf dem Bildschirm passen. Wenn Sie sie komplett sehen wollen, müssen Sie `systemctl` mit der Option `-l` aufrufen.

Übungen

2.1 [2] Welche Vor- und Nachteile hat der traditionelle Ansatz (Textdateien in `/var/log`) gegenüber dem datenbankbasierten Vorgehen des Journals?

2.2 Systemd und journald

Das Journal ist ein integraler Bestandteil von systemd. Im einfachsten Fall verwendet systemd einen Ringpuffer begrenzter Größe in `/run/log/journal`, um eine gewisse Anzahl von Protokollnachrichten im RAM zwischenzuspeichern (wenn Sie die Daten ohnehin an einen traditionellen Protokolldienst weiterleiten wollen, dann reicht das aus). Um alle Vorteile des Journals auszunutzen, sollten Sie aber dafür sorgen, dass die Protokollnachrichten dauerhaft auf der Platte gespeichert werden. Dazu genügt es, das Verzeichnis für die Speicherung anzulegen:

```
# mkdir -p /var/log/journal
# systemctl --signal=USR1 kill systemd-journald
```

(durch das `SIGUSR1` wird systemd dazu gebracht, das bisher im RAM angelegte Journal in die neue Datei auf Platte umzutragen).

Die Komponente von systemd, die sich um das Journal kümmert, heißt `systemd-journald` (oder unter Freunden journald).

Konfiguriert wird das Journal in der Datei `/etc/systemd/journald.conf`. Im `[Journal]`-Abschnitt der Datei (dem einzigen) steht zum Beispiel der Parameter `Storage`, der die folgenden Werte haben kann:

`volatile` Protokollnachrichten werden nur im RAM gespeichert (in `/run/log/journal`), auch wenn `/var/log/journal` existiert.

`persistent` Protokollnachrichten werden bevorzugt auf Platte gespeichert (in `/var/log/journal`). Das Verzeichnis wird angelegt, falls es nicht existiert. Während des frühen Startvorgangs und falls die Platte nicht schreibbar sein sollte, fällt systemd auf `/run/log/journal` zurück.

`auto` Ähnelt `persistent`, aber hier entscheidet die Existenz des Verzeichnisses `/var/log/journal` darüber, ob ein persistentes Journal geschrieben wird – wenn das Verzeichnis nicht existiert, bleibt es beim flüchtigen Journal im RAM.

`none` Im Journal werden überhaupt keine Protokollnachrichten gespeichert. Sie können die Nachrichten aber trotzdem zum Beispiel an den traditionellen syslog-Dienst weiterleiten.

Es gibt noch ein paar andere interessante Parameter. `Compress` gibt an, ob die Protokolldaten (jedenfalls die, die eine gewisse Größe überschreiten) transparent komprimiert werden sollen; der Standardwert ist `yes`. Mit `Seal` können Sie dafür sorgen, dass persistente Journaldateien über eine kryptografische Signatur gegen unbemerkte Manipulationen abgesichert werden. Dafür müssen Sie nur einen Schlüssel zur Verfügung stellen (die Dokumentation erklärt, wie).

Die Parameter `RateLimitInterval` und `RateLimitBurst` sollen es schwieriger machen, das Protokoll mit Nachrichten zu überfluten. Wenn ein Dienst in der durch `RateLimitInterval` gegebenen Zeitspanne mehr als `RateLimitBurst` Nachrichten einreicht, dann werden bis zum Ablauf der Zeitspanne alle weiteren Nachrichten ignoriert (im Protokoll steht dann nur eine Nachricht über die Anzahl der ignorierten Nachrichten. Standardmäßig ist die Grenze bei 1000 Nachrichten in 30 Sekunden; wenn Sie irgendeinen der Parameter auf 0 setzen, wird die Begrenzung aufgehoben.

`SyncIntervalSec` gibt an, in welchen Zeitabständen das Journal auf Platte gesichert wird. Eine Sicherung erfolgt immer unmittelbar nachdem eine Nachricht der Priorität `crit` (oder höher) geschrieben wurde; solange keine solche Nachricht eingeht, wird die mit `SyncIntervalSec` angegebene Zeitspanne abgewartet. Der Standardwert ist »5 Minuten«.

Anschauen können Sie das Protokoll mit dem Kommando `journalctl`:

```
# journalctl
-- Logs begin at Mo 2015-07-27 13:37:14 CEST, end at Mo 2015-07-27▷
 ◁ 22:20:47 CEST. --
Jul 27 13:37:14 blue systemd-journal[138]: Runtime journal is using 4.
Jul 27 13:37:14 blue systemd-journal[138]: Runtime journal is using 4.
Jul 27 13:37:14 blue kernel: Initializing cgroup subsys cpuset
Jul 27 13:37:14 blue kernel: Initializing cgroup subsys cpu
Jul 27 13:37:14 blue kernel: Initializing cgroup subsys cpuacct
Jul 27 13:37:14 blue kernel: Linux version 3.16.0-4-amd64 (debian-kern
Jul 27 13:37:14 blue kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-3.
```

Die Ausgabe erinnert stark an das, was Sie in `/var/log/messages` finden würden, aber hat in der Tat einige Verbesserungen (die hier in der Schulungsunterlage teils nur eingeschränkt zum Tragen kommen):

- Das Protokoll wird mit Ihrem Lieblings-Anzeigeprogramm für Textdateien dargestellt (typischerweise `less`). Mit `less` können Sie die Enden von überlangen Zeilen anschauen, indem Sie die horizontalen Pfeiltasten verwenden.

  Entscheidend dafür ist der Wert der Umgebungsvariable `SYSTEMD_PAGER`, ersatzweise der Wert von `PAGER`, ersatzweise `less`. Mit der Umgebungsvariable `SYSTEMD_LESS` können Sie Optionen für `less` angeben (wenn Sie nicht die Standard-Vorgabe benutzen, wird diese Variable ignoriert, aber Sie können die Optionen dann ja direkt in `SYSTEMD_PAGER` schreiben).

  Wenn Sie `journalctl` mit der Option `--no-pager` aufrufen oder `SYSTEMD_PAGER` auf den Wert `cat` oder eine leere Zeichenkette setzen, wird die Ausgabe nicht seitenweise dargestellt.

- Die Ausgabe umfasst alle zugänglichen Protokolldateien, auch rotierte (dazu später mehr).
- Zeitstempel sind in Zonenzeit (nicht UTC).
- Protokollnachrichten der Priorität `notice` oder `warning` werden im Fettdruck dargestellt.
- Protokollnachrichten der Priorität `error` (oder höher) erscheinen in rot.

`systemd-journald` versucht, den zur Verfügung stehenden Platz sinnvoll auszunutzen. Das heißt, normalerweise werden neue Nachrichten hinten ans Journal angehängt, aber wenn eine gewisse Obergrenze für die Größe des Journals erreicht ist, wird versucht, alte Protokollnachrichten zu entsorgen.


In `/etc/systemd/journald.conf` können Sie die Parameter `SystemMaxUse` und `RuntimeMaxUse` angeben, die beschreiben, wieviel Platz das Journal jeweils unter `/var/log/journal` und `/run/log/journal` einnehmen darf. Die Parameter `SystemKeepFree` und `RuntimeKeepFree` bestimmen dagegen, wieviel Platz in den entsprechenden Dateisystemen frei bleiben soll. `systemd-journald` beachtet beide Werte (`…MaxUse` und `…KeepFree`) und beschränkt sich auf das Minimum.


Die `Runtime…`-Werte kommen beim Systemstart zum Tragen oder wenn kein persistentes Journal verwendet wird. Die `System…`-Werte gelten für das persistente Journal, wenn das System weit genug gebootet ist. Bei der Bestimmung des vom Journal belegten Platzes werden nur Dateien berücksichtigt, deren Namen auf `.journal` endet.


Sie können die Werte in Bytes angeben oder eine der (binären) Einheiten `K`, `M`, `G`, `T`, `P` oder `E` anhängen[2].


Standardwert für `…MaxUse` ist 10% und für `…KeepFree` 15% der Größe des betreffenden Dateisystems. Wenn beim Start von `systemd-journald` schon weniger Platz auf dem Dateisystem frei ist, als der `…KeepFree`-Wert angibt, dann wird die Grenze auch noch weiter reduziert, so dass Platz für andere Sachen bleibt.

Ähnlich wie `logrotate` »rotiert« systemd das Journal, um Platz für neue Nachrichten zu schaffen. Dazu wird der verfügbare Platz in eine Reihe von Dateien aufgeteilt, damit von Zeit zu Zeit die älteste Protokolldatei verworfen werden kann. Diese Rotation ist für Benutzer transparent, da `systemd-journald` die Rotation selbsttätig bei Bedarf vornimmt und `journalctl` immer das komplette Journal auswertet, egal auf wieviele Dateien es verteilt ist.


Maßgeblich für die Aufteilung sind die Parameter `SystemMaxFileSize` und `RuntimeMaxFileSize` in der Datei `/etc/systemd/journald.conf`. Sie geben an, wie groß individuelle Journal-Dateien werden dürfen – der Standard ist »ein Achtel des für das Journal verfügbaren Platzes«, so dass Sie immer eine »Vorgeschichte« von sieben Dateien und die aktuelle Datei zur Verfügung haben.


Sie können das Rotieren von Protokolldateien auch von der Zeit abhängig machen: `MaxFileSec` gibt die maximale Zeitspanne an, bevor systemd eine neue Protokolldatei anfängt. (Normalerweise reicht die größenbasierte Rotation aber völlig aus.) Mit `MaxRetentionSec` können Sie eine Obergrenze für die Haltezeit alter Protokollnachrichten angeben. Der Standardwert für `MaxFileSec` ist `1month` (`0` steht ggf. für »beliebig lange«) und der für `MaxRetentionSec` ist 0 (der Mechanismus ist also ausgeschaltet).

Weiterleitung an syslog

Ebenfalls in `/etc/systemd/journald.conf` können Sie die Weiterleitung an ein traditionelles syslog-System konfigurieren. Dazu setzen Sie einfach

```
[Journal]
ForwardToSyslog=yes
```

Übungen


2.2 [!2] Konfigurieren Sie Ihren Rechner so, dass das Journal persistent auf der Platte liegt. Vergewissern Sie sich, dass das tatsächlich funktioniert hat (etwa indem Sie mit `logger` eine Nachricht ins Protokoll schreiben, den Rechner neu starten und dann prüfen, dass die Nachricht noch da ist).

[2] Wir nehmen mal an, dass es noch ein bisschen dauern wird, bis Sie eine Größenbeschränkung für das Journal in Exbibyte (2^{60} Bytes) angeben müssen, aber es ist beruhigend, dass die systemd-Entwickler anscheinend für die Zukunft planen.

2.3 [2] Hat Ihr Rechner noch einen traditionellen syslog-Daemon? Wenn nein, dann installieren Sie einen (BSD-`syslogd` oder Rsyslog bieten sich an) und sorgen Sie dafür, dass Protokollnachrichten an diesen weitergeleitet werden. Überzeugen Sie sich (etwa mit `logger`), dass das klappt.

2.3 Protokollauswertung

Mit `journalctl` können Sie sehr detaillierte Anfragen an das Journal richten. In diesem Abschnitt beleuchten wir das etwas genauer, aber zuerst noch ein paar Vorbemerkungen.

Zugriffsrechte Während Sie als `root` das komplette Protokoll zu sehen bekommen, steht Ihnen als normaler Benutzer nur Ihr eigenes Protokoll offen, also die Nachrichten von Programmen, die Sie selbst (oder der Computer in Ihrem Namen) gestartet hat. Wenn Sie auch als normaler Benutzer den Vollzugriff haben wollen – wir empfehlen ja, dass Sie auch als Administrator so viel wie möglich mit einem nicht privilegierten Benutzerkonto zu arbeiten –, dann müssen Sie dafür sorgen, dass Sie in der Gruppe `adm` sind:

```
# usermod -a -G adm hugo
```

Sie müssen sich abmelden und wieder anmelden, damit diese Einstellung zum Tragen kommt.

Protokollbeobachtung in Echtzeit In Analogie zum populären »`tail -f`«-Kommando können Sie beobachten, wie neue Nachrichten im Journal abgelegt werden:

```
$ journalctl -f
```

Auch hier bekommen Sie 10 Zeilen angezeigt, und anschließend wartet `journalctl` darauf, dass mehr Meldungen eingehen. Die Anzahl der Zeilen können Sie (wie beim guten alten `tail`) mit der Option `-n` einstellen, und das funktioniert auch ohne `-f`.

Dienste und Prioritäten Mit der Option `-u` können Sie die Ausgabe auf diejenigen Protokollnachrichten beschränken, die eine bestimmte systemd-Unit geschrieben hat:

```
$ journalctl -u ssh
-- Logs begin at Mo 2015-07-27 13:37:14 CEST, end at Di 2015-07-28 ▷
 ◁ 09:32:08 CEST. --
Jul 27 13:37:23 blue sshd[428]: Server listening on 0.0.0.0 port 22.
Jul 27 13:37:23 blue sshd[428]: Server listening on :: port 22.
Jul 27 13:56:50 blue sshd[912]: Accepted password for hugo from 192.16
Jul 27 13:56:50 blue sshd[912]: pam_unix(sshd:session): session opened
```

Statt eines gezielten Unit-Namens können Sie auch ein Shell-Suchmuster angeben, um mehrere Units zu erfassen. Oder Sie geben einfach mehrere `-u`-Optionen an.

Um nur Nachrichten bestimmter Prioritäten angezeigt zu bekommen, können Sie die Option `-p` benutzen. Diese übernimmt entweder eine einzelne numerische oder textuelle Priorität (`emerg` hat den Wert 0, `debug` hat den Wert 7) und beschränkt die Ausgabe dann auf Nachrichten der entsprechenden Priorität oder darüber (darunter, wenn Sie nach den numerischen Werten gehen wollen). Oder Sie geben einen Bereich in der Form

```
$ journalctl -p warning..crit
```

an, um nur die Nachrichten zu sehen, deren Priorität in diesem Bereich ist.

Selbstverständlich können Sie die Optionen `-u` und `-p` auch kombinieren:

```
$ journalctl -u apache2 -p err
```

zeigt Ihnen alle Fehlermeldungen (oder schlimmer) von Apache.

Die Option `-k` beschränkt die Ausgabe auf Nachrichten, die der Systemkern geschickt hat. Dabei werden nur Nachrichten seit dem letzten Systemstart berücksichtigt.

Zeit Wenn Sie sich nur für Nachrichten in einem bestimmten Zeitraum interessieren, können Sie die Ausgabe entsprechend einschränken. Die Optionen `--since` und `--until` gestatten die Angabe eines Datums oder einer Uhrzeit in der Form »`2015-07-27 15:36:11`«, und es werden nur Nachrichten ausgegeben, die seit oder bis zu dem gegebenen Zeitpunkt protokolliert wurden.

Sie können die Uhrzeit komplett weglassen, dann wird »`00:00:00`« angenommen. Oder Sie lassen nur die Sekunden weg, dann ist »`…:00`« impliziert. Wenn Sie das Datum weglassen (dann ist natürlich eine Uhrzeit nötig, mit oder ohne Sekunden), dann nimmt `journalctl` »heute« an.

Die Schlüsselwörter `yesterday`, `today` und `tomorrow` stehen respektive für »`00:00:00`« gestern, heute oder morgen.

Relative Zeitangaben sind auch erlaubt: »`-30m`« steht für »vor einer halben Stunde«. (»`+1h`« steht für »in einer Stunde«, aber es ist unwahrscheinlich, dass Ihr Systemprotokoll Einträge aus der Zukunft enthält[3].)

Jeder Systemstartvorgang bekommt eine eindeutige Kennung, und Sie können Ihre Suche auf den Teil des Journals beschränken, der zwischen einem bestimmten Systemstart und dem nächsten liegt. Im einfachsten Fall betrachtet »`journalctl -b`« nur die Meldungen des aktuellen Rechnerlaufs:

```
$ journalctl -b -u apache2
```

Mit der Option `--list-boots` gibt `journald` eine Liste der Startvorgänge aus, die im aktuellen Journal zu finden sind, zusammen mit den Zeiträumen, für die es Protokolleinträge gibt:

```
$ journalctl --list-boots
-1 30eb83c06e054feba2716a1512f64cfc Mo 2015-07-27 22:45:08 CEST–▷
 ◁ Di 2015-07-28 10:03:31 CEST
 0 8533257004154353b45e99d916d66b20 Di 2015-07-28 10:04:22 CEST–▷
 ◁ Di 2015-07-28 10:04:27 CEST
```

Sie können sich auf bestimmte Startvorgänge beziehen, indem Sie bei `-b` deren Index angeben (`1` steht für den chronologisch ersten Startvorgang im Protokoll, `2` für den zweiten und so weiter) oder den negativen Versatz in der ersten Spalte der Ausgabe von »`journalctl --list-boots`« (`-0` steht für den aktuellen Startvorgang, `-1` für den vorigen und so weiter).

[3] Es sei denn, Sie sind der Doktor und durchsuchen das Journal der TARDIS.

```
Mo 2015-07-27 13:37:23.580820 CEST [s=89256633e44649848747d32096fb42▷
 ◁ 68;i=1ca;b=30eb83c06e054feba2716a1512f64cfc;m=11a1309;t=51bd9c6f▷
 ◁ 8812e;x=f3d8849a4bcc3d87]
    PRIORITY=6
    _UID=0
    _GID=0
    _SYSTEMD_SLICE=system.slice
    _BOOT_ID=30eb83c06e054feba2716a1512f64cfc
    _MACHINE_ID=d2a0228dc98041409d7e68858cac6aba
    _HOSTNAME=blue
    _CAP_EFFECTIVE=3fffffffff
    _TRANSPORT=syslog
    SYSLOG_FACILITY=4
    SYSLOG_IDENTIFIER=sshd
    SYSLOG_PID=428
    MESSAGE=Server listening on 0.0.0.0 port 22.
    _PID=428
    _COMM=sshd
    _EXE=/usr/sbin/sshd
    _CMDLINE=/usr/sbin/sshd -D
    _SYSTEMD_CGROUP=/system.slice/ssh.service
    _SYSTEMD_UNIT=ssh.service
    _SOURCE_REALTIME_TIMESTAMP=1437997043580820
```

Bild 2.1: Vollständige Protokollausgabe mit `journalctl`

Sie können auch die 32 Zeichen lange alphanumerische Boot-ID angeben, die in der zweiten Spalte der Ausgabe von »`journalctl --list-boots`« steht, um das Journal nur für den betreffenden Startvorgang zu durchsuchen. Auch dabei können Sie einen positiven oder negativen Versatz anhängen, um Startvorgänge davor oder danach zu identifizieren: Im obigen Beispiel ist

```
$ journalctl -b 8533257004154353b45e99d916d66b20-1
```

eine umständliche Methode,

```
$ journalctl -b 1
```

zu sagen.

Beliebige Suchoperationen Wenn Sie einen Pfadnamen als Parameter angeben, versucht `journalctl` etwas Sinnvolles damit anzufangen:

- Wenn er sich auf eine ausführbare Datei bezieht, dann sucht es nach Protokolleinträgen, die das betreffende Programm gemacht hat.
- Wenn er sich auf eine Gerätedatei bezieht, sucht es nach Einträgen, die das angegebene Gerät betreffen.

Diese Suchoperationen sind Spezialfälle eines allgemeineren Suchmechanismus, den das Journal anbietet. Systemd protokolliert nämlich weitaus mehr Informationen, als der traditionelle syslog-Mechanismus das tut[4]. Sie sehen das, wenn Sie `journalctl` mit der Option `--output=verbose` aufrufen (siehe Bild 2.1).

[4] Auch hier muss man zur Ehrenrettung der modernen Implementierungen erwähnen, dass diese durchaus mehr machen, als sie müssen – auch wenn sie diese Funktionalität mitunter erst sehr kürzlich erworben haben, um mit dem Journal von systemd gleichzuziehen.


Die erste Zeile in Bild 2.1 ist ein Zeitstempel für die Nachricht zusammen mit einem »Cursor«. Der Cursor identifiziert die Nachricht innerhalb des Journals und wird zum Beispiel gebraucht, um Protokolle auf entfernten Rechnern abzulegen.


Die Folgezeilen sind Felder im Journal, die sich auf die betreffende Nachricht beziehen. Feldnamen ohne Unterstreichung am Anfang gehen auf Informationen zurück, die das protokollierende Programm geliefert hat, und sind darum nicht notwendigerweise vertrauenswürdig (das Programm könnte zum Beispiel versuchen, über seine PID oder seinen Namen – in `SYSLOG_IDENTIFIER` – Unfug zu erzählen). Feldnamen mit einer Unterstreichung am Anfang werden von systemd geliefert und können vom protokollierenden Programm nicht verfälscht werden.


`PRIORITY`, `SYSLOG_FACILITY`, `SYSLOG_IDENTIFIER`, `SYSLOG_PID` und `MESSAGE` stammen aus dem syslog-Protokoll und sind ziemlich selbsterklärend. `_UID`, `_GID`, `_HOSTNAME`, `_PID` und `_SYSTEMD_UNIT` erklären sich ebenfalls selber. `_BOOT_ID` ist die Kennung des aktuellen Startvorgangs und `_MACHINE_ID` identifiziert den protokollierenden Rechner aufgrund seiner Kennung in `/etc/machine-id`. `_CAP_EFFECTIVE` gibt an, welche Sonderrechte (*capabilities*) der protokollierende Prozess hat, und `_TRANSPORT` beschreibt, auf welchem Weg die Nachricht systemd erreicht hat (gängig sind außer `syslog` zum Beispiel `stdout` für Nachrichten, die das Programm auf seiner Standardausgabe oder Standardfehlerausgabe geschrieben hat oder `kernel` für Nachrichten, die der Systemkern über `/dev/klog` eingeliefert hat). `_COMM`, `_EXE` und `_CMDLINE` beschreiben alle das Kommando, das gerade ausgeführt wird. `_SYSTEMD_SLICE` und `_SYSTEMD_CGROUP` geben an, wo in der internen Prozessverwaltung von systemd der protokollierende Prozess zu finden ist. Eine genauere Erklärung finden Sie in `systemd.journal-fields`(7).

Nach allen diesen Feldern können Sie suchen, einfach indem Sie sie auf der Kommandozeile von `journalctl` angeben:

```
$ journalctl _HOSTNAME=red _SYSTEMD_UNIT=apache2.service
```


Suchbegriffe mit verschiedenen Feldern werden implizit UND-verknüpft. Taucht dasselbe Feld in mehreren Suchbegriffen auf, werden diese implizit ODER-verknüpft.


Es gibt auch ein explizites ODER:

```
$ journalctl _HOSTNAME=red _UID=70 + _HOSTNAME=blue _UID=80
```

zeigt alle Nachrichten von Prozessen mit der UID 70 auf dem Rechner `red` sowie von Prozessen mit der UID 80 auf dem Rechner `blue`. (Das funktioniert selbstredend nur, wenn Sie die Journale beider Rechner auf Ihrem Rechner konsolidieren.)


Natürlich können Sie diese Suchbegriffe beliebig mit Optionen kombinieren, etwa um zeitliche Einschränkungen zu machen oder sich Tipparbeit zu sparen:

```
$ journalctl -u apache2 _HOSTNAME=red
```

Wenn Sie sich (wie wir) nicht merken können, welche Werte für einen Suchbegriff in Frage kommen, können Sie einfach das Journal fragen:

```
$ journalctl -F _SYSTEMD_UNIT
session-2.scope
```

```
udisks2.service
session-1.scope
polkitd.service
dbus.service
user@1000.service
◁◁◁◁◁
```

Als weitere Vereinfachung funktioniert sogar Kommandozeilenvervollständigung für Feldnamen und -Werte:

```
$ journalctl _SYS⇥                                                    wird zu
$ journalctl _SYSTEMD_⇥
_SYSTEMD_CGROUP= _SYSTEMD_OWNER_UID= _SYSTEMD_SESSION= _SYSTEMD_UNIT=
$ journalctl _SYSTEMD_U⇥                                              wird zu
$ journalctl _SYSTEMD_UNIT=⇥⇥
acpid.service       lightdm.service       ssh.service
anacron.service     networking.service    systemd-journald.service
◁◁◁◁◁
$ journalctl _SYSTEMD_UNIT=ss⇥                                        wird zu
$ journalctl _SYSTEMD_UNIT=ssh.service
```

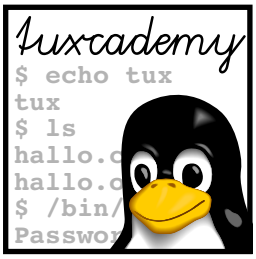
Das Journal und `journald` sind ungeheuer flexibel und leistungsfähig und lassen die traditionelle Methode (Textdateien in `/var/log`) im Vergleich ziemlich primitiv aussehen.

Übungen

2.4 [!2] Experimentieren Sie mit `journalctl`. Wie viele verschiedene Benutzeridentitäten haben auf Ihrem Rechner Nachrichten ans Journal geschickt? Ist gestern zwischen 12 und 13 Uhr etwas Interessantes passiert? Was waren die letzten 10 Meldungen der Priorität `warning`? Überlegen Sie sich selber einige interessante Fragen und beantworten Sie sie.

Zusammenfassung

- Das »Journal« ist ein moderner Systemprotokolldienst, der von systemd zur Verfügung gestellt wird. Es beruht auf binären, datenbankartigen Protokolldateien.
- Das Journal steht entweder in `/run/log/journal` oder (für persistente Protokollierung auf Platte) in `/var/log/journal`.
- Innerhalb von systemd kümmert sich `systemd-journald` um das Journal. Mit `journalctl` können Sie auf das Journal zugreifen.
- `journalctl` erlaubt sehr ausgefeilte Abfragen des Journals.



3

Grundlagen von TCP/IP

Inhalt

3.1 Geschichte und Grundlagen 44
3.1.1 Die Geschichte des Internet 44
3.1.2 Verwaltung des Internet 45
3.2 Technik 46
3.2.1 Überblick. 46
3.2.2 Protokolle 47
3.3 TCP/IP 50
3.3.1 Überblick. 50
3.3.2 Kommunikation von Ende zu Ende: IP und ICMP 51
3.3.3 Die Basis für Dienste: TCP und UDP 54
3.3.4 Die wichtigsten Anwendungsprotokolle 58
3.4 Adressen, Wegleitung und Subnetting 59
3.4.1 Grundlagen 59
3.4.2 Wegleitung 60
3.4.3 IP-Netzklassen 61
3.4.4 Subnetting 62
3.4.5 Private IP-Adressen 62
3.4.6 Masquerading und Portweiterleitung 63
3.5 IPv6. 64
3.5.1 Überblick. 64
3.5.2 IPv6-Adressierung 65

Lernziele

- Den grundlegenden Aufbau der TCP/IP-Protokollfamilie kennen
- Grundlagen der IP-Adressierung kennen
- Konzepte von Subnetting und Routing verstehen
- Die wichtigsten Eigenschaften und Unterschiede von TCP, UDP und ICMP kennen
- Die wichtigsten TCP- und UDP-Dienste kennen
- Die wesentlichen Unterschiede zwischen IPv4 und IPv6 kennen

Vorkenntnisse

- Grundlegende Kenntnisse von Rechnernetzen und TCP/IP-Diensten als Anwender sind hilfreich

3.1 Geschichte und Grundlagen

3.1.1 Die Geschichte des Internet

Die Geschichte der Vernetzung von Computern reicht zurück bis fast an den Anfang des »Computerzeitalters«. Die meisten Techniken der Frühzeit sind heute längst vergessen – das »Internet« hat sich weithin durchgesetzt. Aber was ist »das Internet« überhaupt und wo kommt es her? In diesem Abschnitt geben wir einen kurzen Überblick über seine Geschichte und die Entwicklung der weltweiten Rechnerkommunikation. Wenn Sie das schon woandersher wissen, können Sie gleich zum nächsten Abschnitt weiterblättern. Vielen Dank.

ARPAnet Ursprung des heutigen Internet ist das ARPAnet, dessen Entwicklung vom amerikanischen Verteidigungsministerium finanziert wurde. Wir schreiben die späten 1960er Jahre.

Ziel der Sache war nicht, wie gerne behauptet wird, die Konstruktion einer Kommunikationsinfrastruktur für den Fall eines Atomkriegs, sondern lediglich die Erforschung der Datenübertragung, wobei man gleichzeitig die Kommunikation zwischen den an der Rüstungsforschung beteiligten Firmen und Universitäten vereinfachen wollte.

1969 bestand das ARPAnet aus 4 Knoten; 1970 bis 1972 wurde das *Network Control Protocol* (NCP) als grundlegender Standard für die Kommunikation auf dem ARPAnet implementiert. Der wichtigste Dienst damals war elektronische Post.

NCP

In den 1970er Jahren gewann die Idee eines »Internet«, das verschiedene schon existierende Netze verbinden sollte, an Bedeutung. Man versuchte sich an der Implementierung von »TCP«, einem zuverlässigen Kommunikationsprotokoll auf der Basis eines unzuverlässigen Übertragungsprotokolls (die Idee, mit UDP auch ein unzuverlässiges Kommunikationsprotokoll zur Verfügung zu stellen, kam erst einige Zeit später dazu, was den Namen »TCP/IP« (anstelle von »TCP/UDP/IP« oder ähnlichem) erklären dürfte). Die ersten TCP-Implementierungen erschienen in den frühen 1970ern auf »großen« Systemen wie TOPS-20 oder Tenex; wenig später wurde bewiesen, dass die Implementierung von TCP auf Workstationrechnern wie dem Xerox Alto möglich war, dass also auch solche Rechner am Internet teilnehmen konnten. Das erste Ethernet wurde ebenfalls 1973 am Xerox PARC entwickelt.

Die heutigen grundlegenden TCP/IP-Standards kamen Anfang der 1980er Jahre heraus. Sie wurden testhalber im damaligen BSD – der an der *University of California at Berkeley* entwickelten Unix-Variante – implementiert, das dadurch das Fundament für seine Popularität bei Benutzern und Rechnerherstellern legte. Am 1. Januar 1983 erfolgte die Umstellung des ARPAnet von NCP auf TCP/IP. Wenig später wurde das ursprüngliche ARPAnet administrativ in die beiden Komponenten MILnet (für die militärischen Anwendungen) und ARPAnet (für die Rüstungsforschung) aufgeteilt. Ebenfalls in 1983 wurde mit der Entwicklung des DNS der Grundstein für die künftige Expansion gelegt. In den künftigen Jahren – 1984 bis 1986 – entstanden weitere Netze auf der Basis von TCP/IP, etwa das NSFNET der *National Science Foundation* der USA, und der Begriff »Internet« für die Zusammenfassung aller miteinander verbundenen TCP/IP-basierten Netze begann Fuß zu fassen.

TCP/IP Flag Day
MILnet
ARPAnet
NSFNET

Ende 1989 waren Australien, Deutschland, Großbritannien, Israel, Italien, Japan, Mexiko, die Niederlande und Neuseeland ans Internet angeschlossen. Man zählte jetzt über 160.000 Knoten.

1990 wurde das ARPAnet offiziell außer Dienst gestellt (es war längst im »Internet« aufgegangen) und 1991 das NSFNET für die kommerzielle Nutzung freigegeben. Kommerzielle Anbieter expandierten. Heute ist der überwiegende Teil der Netzinfrastruktur in privater Hand.

Heute existiert ein globales Leitungsnetz mit einem einheitlichen Adressraum. Verwendet werden offene Protokolle und einheitliche Kommunikationsverfahren, so dass sich jeder an der Entwicklung beteiligen kann und das Netz jedem zur

Verfügung steht. Die Weiterentwicklung des Internet ist noch lange nicht abgeschlossen; künftige Neuerungen werden versuchen, anstehende Probleme wie die Verknappung der Adressen und die gestiegenen Sicherheitsbedürfnisse zu beheben.

3.1.2 Verwaltung des Internet

Ein globales Netz wie das Internet kann nicht ohne Verwaltungsstrukturen funktionieren. Diese Aufgabe wurde in den USA angesiedelt, da hier zu Beginn die meisten Netze miteinander verbunden waren. Dort – genauer gesagt beim US-Handelsministerium – liegt die Verwaltung auch noch heute.

Diversen Leuten ist die Dominanz der USA in Fragen des Internets ein Dorn im Auge. Leider weiß man aber nicht so recht, was man dagegen unternehmen soll, da die Amerikaner das Heft nicht formell aus der Hand geben wollen. Auf der anderen Seite verfolgt das US-Handelsministerium einen ausgeprägten Laissez-faire-Ansatz, der die Kritiker bis zu einem gewissen Grad mit dem *status quo* versöhnt.

Internet Society

Theoretisch liegt die Kontrolle über das Internet in der Hand der »Internet Society« (ISOC), einer 1992 gegründeten internationalen nicht gewinnorientierten Organisation. Ihre Mitglieder sind Regierungen, Firmen, Universitäten, andere Organisationen und auch Einzelpersonen (jeder darf mitmachen).

Ziel der ISOC war, den bis dahin juristisch sehr schwammigen Institutionen wie der IETF (siehe unten) einen formellen Rahmen zu geben und auch deren finanzielle Unterstützung zu sichern. Ferner hält die ISOC das Urheberrecht an den RFCs, den Standarddokumenten für das Internet, die allen Interessenten frei zur Verfügung stehen.

Die Aktivitäten der ISOC teilen sich grob auf drei Kategorien auf:

Normen Die ISOC ist der Überbau für eine Reihe von Gremien, die sich um die technische Weiterentwicklung des Internet bemühen. Unter anderem:

- Das *Internet Architecture Board* (IAB) ist das Komitee, das die Aufsicht über die technische Weiterentwicklung des Internet hat. Das IAB kümmert sich zum Beispiel um die Veröffentlichung der RFCs und berät die Leitung der ISOC in technischen Fragen.

  Das IAB hat im Moment ein gutes Dutzend Mitglieder (Menschen), die vom »IETF-Nominierungskomitee« ausgewählt wurden, einen ebenfalls vom IETF-Nominierungskomitee eingesetzten Vorsitzenden und ein paar Ex-Officio-Mitglieder und Vertreter anderer Organisationen.

- Die *Internet Engineering Task Force* (IETF) kümmert sich um die tatsächliche Entwicklung von Internet-Standards und arbeitet dabei eng mit Institutionen wie ISO/IEC und dem World Wide Web Consortium (W3C) zusammen. Die IETF ist eine offene Organisation ohne Mitgliedschaft, die von »Freiwilligen« betrieben wird (deren Arbeitgeber in der Regel für die Kosten aufkommen). Innerhalb der IETF gibt es eine große Anzahl von »Arbeitsgruppen«, die sich gemäß ihrer Thematik in »Gebiete« (engl. *areas*) aufteilen. Jedes Gebiet hat einen oder zwei *area directors*, die zusammen mit dem Vorsitzenden der IETF die *Internet Engineering Steering Group* (IESG) bilden. Diese ist für die Arbeit der IETF verantwortlich.

  Wegen ihrer amorphen Struktur ist es schwer zu sagen, wie groß die IETF zu einem gegebenen Zeitpunkt tatsächlich ist. In den ersten Jahren seit ihrer Gründung 1986 schwankte die Anwesenheit bei den regelmäßigen Treffen zwischen 30 und 120 Personen. Seit

dem explosionsartigen Wachstum des Internet in den 1990er Jahren ist der Kreis etwas größer geworden, auch wenn er nach dem Platzen der »Dot-Com«-Blase von bis zu 3000 Personen im Jahre 2000 wieder auf um die 1200 abgesunken ist.

Das Mantra der IETF ist »Grober Konsens und laufender Code« (*rough consensus and running code*) – man besteht bei Entscheidungen nicht auf Einstimmigkeit, aber möchte den größten Teil der Gruppe hinter sich wissen. Außerdem wird sehr viel Wert auf Lösungen gelegt, die tatsächlich in der Praxis funktionieren. Dies und die Tatsache, dass die Arbeit zum größten Teil von Freiwilligen geleistet wird, können dazu führen, dass es mitunter lange dauert, bis eine IETF-Arbeitsgruppe Ergebnisse liefert – vor allem, wenn es zu wenige oder zu viele Interessenten gibt, die dazu beitragen möchten.

- Die *Internet Corporation for Assigned Names and Numbers*, kurz ICANN, ist eine andere nicht gewinnorientierte Organisation, die 1998 gegründet wurde, um einige Dinge zu übernehmen, die vorher andere Organisationen (vor allem die IANA, siehe nächster Punkt) direkt im Namen der US-Regierung erledigt haben. Insbesondere gemeint ist die Verteilung von IP-Adressen und DNS-Top-Level-Domainnamen. Gerade letztere ist ein extrem politisches Thema und immer wieder Quelle von Konflikten.
- Die *Internet Assigned Numbers Authority* (IANA) kümmert sich um die tatsächliche Zuteilung von IP-Adressen und die Verwaltung der DNS-Root-Server. Administrativ ist die IANA Teil der ICANN. Außerdem verwaltet die IANA alle global eindeutigen Namen und Zahlen in Internet-Protokollen, die als RFCs veröffentlicht sind. Dabei arbeitet sie eng mit der IETF und der RFC-Redaktion zusammen.

  Die Zuteilung von IP-Adressen delegiert die IANA weiter an sogenannte *Regional Internet Registries* (RIRs), die jeweils für einen Teil der Welt den »Vertrieb« (typischerweise) an Provider übernehmen. Im Moment gibt es fünf RIRs, zuständig für Deutschland ist das RIPE NCC.

Fortbildung Die ISOC veranstaltet Konferenzen, Seminare und Kurse über wichtige Internet-Themen, unterstützt lokale Internet-Organisationen und gibt Internet-Experten in Entwicklungsländern durch finanzielle Hilfen die Möglichkeit, an der Weiterentwicklung und Diskussion teilzunehmen.

Politische Willensbildung Die ISOC kooperiert mit Regierungen und nationalen und internationalen Organisationen, um die Ideen und Werte der ISOC voranzubringen. Erklärtes Ziel der ISOC ist »eine Zukunft, in der Menschen in allen Teilen der Welt das Internet verwenden können, um ihre Lebensqualität zu verbessern«.

3.2 Technik

3.2.1 Überblick

Computer verarbeiten digitale Informationen. Diese Informationen werden in der »wirklichen Welt« allerdings durch physikalische Phänomene wie Spannung, Ladung, Licht und ähnliches dargestellt, und die wirkliche Welt ist nun mal fundamental »analog«. Die erste Herausforderung der Datenkommunikation ist also, die digitalen Informationen im Computer für die Übertragung zu einem anderen Computer in etwas Analoges umzuwandeln – zum Beispiel eine Reihenfolge von elektrischen Impulsen auf einem Kabel – und am anderen Ende daraus wieder digitale Informationen zu machen. Die nächste Herausforderung besteht darin, das

zum Funktionieren zu bringen, wenn der eine Computer in Berlin und der andere in Neuseeland steht.

Man kann Datennetze ganz grob und ohne nähere Beleuchtung der verwendeten Technik in zwei Gruppen einteilen: **Lokale Netze** (engl. *Local Area Networks* oder »LANs«) verbinden eine kleine Anzahl von Stationen in einem räumlich begrenzten Areal, **Weitverkehrsnetze** (engl. *Wide Area Networks* oder »WANs«) eine potentiell große Anzahl von Stationen in einem räumlich sehr großen Areal.

Lokale Netze

Weitverkehrsnetze

Bei LANs ist der Eigentümer (eine Firma oder andere Organisation oder – heute oft – ein Haushalt) in der Regel auch der Betreiber und der einzige Nutzer, und das Netz verfügt über eine hohe Bandbreite (100 MBit/s und mehr). WANs dagegen verbinden eine Vielzahl von Nutzern, die normalerweise nicht Eigentümer des Netzes sind; die Bandbreiten sind kleiner und die Nutzung teurer.

Es gibt eine Vielzahl verschiedener Netzwerktechniken für die unterschiedlichsten Anforderungen – von drahtlosen Verbindungen über sehr kurze Strecken (Bluetooth) über typische LAN-Technik wie Ethernet bis zu Glasfaserverbindungen mit ATM für WANs. Als Programmierer und Systemadministratoren möchten wir mit deren ekligen elektro- und nachrichtentechnischen Details allerdings so wenig wie möglich konfrontiert werden. Deswegen sprechen wir von einem »Protokollstapel« und versuchen, die einzelnen Bestandteile – den »elektrischen« Teil, die prinzipielle Kommunikation zwischen Rechnern im selben Netz, die prinzipielle Kommunikation zwischen Rechnern in verschiedenen Netzen und schließlich konkrete »Dienste« wie E-Mail oder das World Wide Web sauber zu trennen. Aber der Reihe nach.

3.2.2 Protokolle

Ein »Protokoll« ist eine Vereinbarung dafür, wie zwei (oder mehr) Stationen in einem Netz sich unterhalten. Das Spektrum der möglichen Protokolle reicht von Regeln für elektrische Signale auf einem Ethernet-Kabel oder Funksignale in einem WLAN bis hin zu (beispielsweise) Protokollen, die den Zugriff auf einen SQL-Datenbankserver regeln. Protokolle lassen sich grob in drei Klassen einteilen:

Übertragungsprotokolle (oft auch »Zugriffsverfahren« genannt) regeln die Datenübertragung grob gesagt auf der Ebene von Netzwerkkarten und Leitungen. Ihre Ausgestaltung hängt von den elektro- und nachrichtentechnischen Eigenschaften und Einschränkungen ab, die aus der Realisierung in »Hardware« folgen. Die Kommunikation zwischen zwei Rechnern über ein serielles Nullmodem-Kabel läuft zum Beispiel ganz anders ab als die Übertragung von Daten über eine Funkverbindung im WLAN, und es existieren völlig andere Anforderungen an die verwendeten Zugriffsverfahren.

Das gängigste Zugriffsverfahren im LAN-Bereich ist Ethernet, auch wenn das heutige Ethernet mit dem gleichnamigen Original von 1973 so gut wie nichts mehr zu tun hat (OK, beide involvieren Elektrizität, aber da hört die Ähnlichkeit auch fast schon auf.) Andere Standards wie Token-Ring oder Feldbussysteme tauchen nur noch in Spezialanwendungen auf. Heute populär sind auch WLAN-Zugriffsverfahren wie IEEE 802.11.

Kommunikationsprotokolle dienen dazu, die Kommunikation zwischen Rechnern in verschiedenen Netzen zu regeln, ohne dass dafür eine genaue Kenntnis der verwendeten Zugriffsverfahren notwendig ist. Damit Sie auf Ihrem Heim-PC in Deutschland eine Webseite über Kängurus auf einem Server einer Universität in Australien anschauen können, möchten Sie nicht wissen müssen, dass Ihr PC über Ethernet mit Ihrem Heim-Router redet,

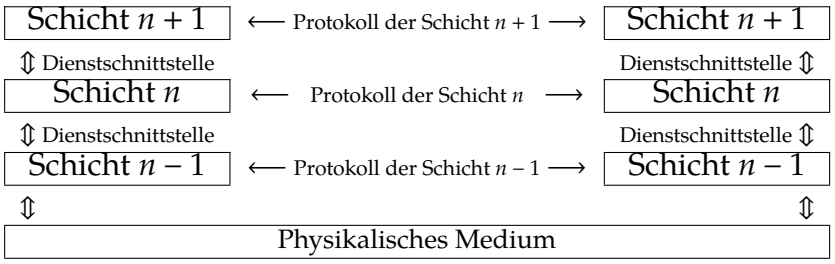


Bild 3.1: Protokolle und Dienstschnittstellen

dieser wiederum über ATM mit dem DSLAM draußen im Telekom-Häuschen auf der anderen Straßenseite, von da geht es dann via Glasfaser und um einige Ecken herum nach Australien und so weiter und so fort – Sie geben einfach `www.kangaroos-r-us.au` in Ihrem Browser ein. Dass Ihr Heim-PC den entfernten Web-Server findet und mit ihm Daten austauschen kann, verdanken Sie den Kommunikationsprotokollen.

Kommunikationsprotokolle sind dafür da, dass Sie sich nicht mit den Übertragungsprotokollen herumärgern müssen, aber sie können ohne diese natürlich nicht funktionieren. Ziel der Kommunikationsprotokolle ist, die unappetitlichen Details der Übertragungsprotokolle vor Ihnen zu verstecken – etwa so, wie das Gaspedal Ihres Autos dazu dient, die Kennlinie der elektronischen Benzineinspritzung vor Ihnen zu verstecken.

Die Kommunikationsprotokolle, die uns interessieren, sind natürlich IP, TCP und UDP. Dazu kommt noch ICMP als »Infrastrukturprotokoll«, das zur Diagnose, Steuerung und Fehlermeldung dient.

Anwendungsprotokolle realisieren auf der Basis der Kommunikationsprotokolle tatsächliche Dienste wie elektronische Post, Dateiübertragung oder Internet-Telefonie. Wenn Kommunikationsprotokolle dafür gut sind, beliebige Bits und Bytes nach Australien zu schicken und andere von dort zurückzubekommen, dann sorgen Anwendungsprotokolle dafür, dass Sie mit diesen Bits und Bytes tatsächlich etwas anfangen können.

Typische Anwendungsprotokolle, mit denen Sie als Linux-Administrator konfrontiert werden könnten, sind zum Beispiel SMTP, FTP, SSH, DNS, HTTP, POP3 oder IMAP, zum Teil auch mit »sicheren«, also authentisierten und verschlüsselten Ablegern. Alle diese Protokolle werden von Anwendungsprogrammen wie Mail-Clients oder Web-Browsern verwendet und bauen auf Kommunikationsprotokollen wie TCP oder UDP auf.

Protokolldateneinheiten

Die Daten, die über ein Protokoll ausgetauscht werden, nennen wir abstrakt Protokolldateneinheiten – je nach Protokoll können sie spezifischere Namen haben, etwa »Pakete«, »Datagramme«, »Segmente« oder »Frames«.

Schichtenmodell

Aus der Tatsache, dass Kommunikationsprotokolle dafür gedacht sind, die Details der Übertragungsprotokolle zu verstecken, und Anwendungsprotokolle wiederum dazu dienen, die Details der Kommunikationsprotokolle zu verstecken, kann man ein »Schichtenmodell« (Bild 3.1) konstruieren, bei dem die Übertragungsprotokolle die unterste und die Anwendungsprotokolle die oberste Ebene einnehmen. (Daher kommt auch der Begriff »Protokollstapel«.) Jede Schicht auf der Absenderseite empfängt Daten »von oben« und gibt sie »nach unten« weiter; auf der Empfängerseite werden sie »von unten« empfangen und »nach oben« weitergegeben. Konzeptuell sagt man trotzdem, dass zwei Stationen zum Beispiel »über HTTP« kommunizieren, auch wenn die HTTP-Daten in Wirklichkeit über

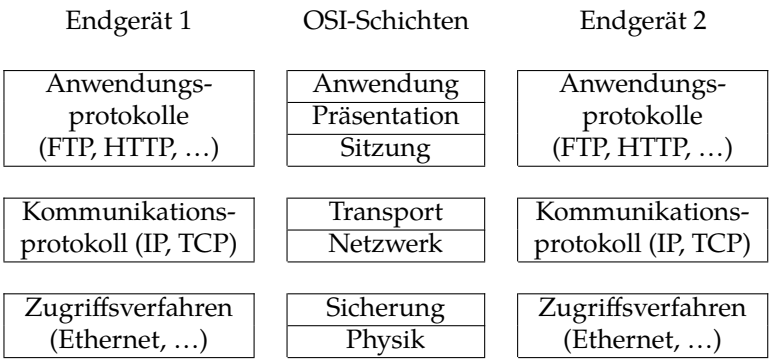


Bild 3.2: ISO/OSI-Referenzmodell

TCP, IP und einen ganzen Zoo von Übertragungsprotokollen von einer Station zur anderen fließen und dort wiederum die IP- und TCP-Schichten passieren müssen, bis sie wieder als HTTP-Daten »sichtbar« werden.

Technisch empfängt auf der Absenderseite auf jeder Ebene das entsprechende Protokoll an seiner Dienstschnittstelle ein »Datenpaket« von der darüberliegenden Schicht und fügt ihm einen »Kopf« mit allen für seine Aufgabe wichtigen Informationen hinzu, bevor es über die Dienstschnittstelle des Protokolls der darunterliegenden Schicht weitergegeben wird. Alles, was an der Dienstschnittstelle übergeben wird, betrachtet die nächstuntere Ebene als Daten; die Kopfdaten des vorherigen Protokolls sind auf der tieferen Ebene nicht von Interesse. Auf der Empfängerseite der Kommunikation durchlaufen die Pakete die gleichen Schichten in umgekehrter Reihenfolge, wobei jede Ebene »ihren« Kopf wieder entfernt und die »Nutzdaten« nach oben weitergibt.

Kopf

Das bekannteste Schichtenmodell ist das »ISO/OSI-Referenzmodell« (Bild 3.2). ISO/OSI (kurz für *International Organisation for Standardisation/Open Systems Interconnection*) war die Basis einer Protokollfamilie, die vom CCITT, der Weltorganisation der Telekommunikationsbehörden und -unternehmen, vorgeschlagen wurde.

ISO/OSI-Referenzmodell

Die ISO/OSI-Netzwerkstandards haben sich nie durchsetzen können – sie waren zu kompliziert und praxisfern, um nützlich zu sein, und die Standarddokumente waren nur schwer zugänglich –, das Referenzmodell mit seinen sieben (!) Schichten ist aber übriggeblieben und wird gerne herangezogen, um die Abläufe bei der Datenübertragung zu erklären.

Viele Protokollstapel lassen sich nicht exakt auf das ISO/OSI-Referenzmodell abbilden. Dies kommt zum einen daher, dass sich nicht jeder Hersteller an die Definitionen des Modells hält, zum anderen aber auch daher, dass einige Protokollstapel schon älter als das OSI-Modell sind. Sie sollten auch nicht den Fehler machen, das ISO/OSI-Referenzmodell mit einer verbindlichen »Norm« für die Struktur von Netzwerksoftware oder gar mit einer Implementierungsanleitung zu verwechseln. Das ISO/OSI-Referenzmodell ist lediglich eine Klarstellung der verschiedenen beteiligten Konzepte und macht es bequemer, über sie zu reden. Trotzdem hier ein kurzer Überblick über die beteiligten Schichten:

- Die Schichten 1 und 2 (physikalische Schicht und Sicherungsschicht) beschreiben, wie die Daten auf das Kabel geschickt werden. Dies beinhaltet neben dem Zugriffsverfahren auch die Codierung der Daten.
- Die Schicht 3 (Netzwerkschicht) definiert die Funktionen, die für die Wegleitung der Daten benötigt werden. Dies umfasst auch die dafür erforderliche Adressierung.
- Der Transport der Applikationsdaten wird in der Schicht 4 (Transportschicht) beschrieben. Hier unterscheidet man zwischen verbindungsorientierten und verbindungslosen Diensten.

- Die Schichten 5, 6 und 7 (Sitzungs-, Präsentations- und Anwendungsschicht) werden in der Praxis nicht so häufig explizit unterschieden (z. B. nicht bei den TCP/IP-Protokollen). Sie beschreiben die systemunabhängige Darstellung von Daten im Netzwerk sowie die Schnittstelle zu den Anwendungsprotokollen.
- Andy Tanenbaum [Tan02] postuliert zusätzlich noch die Schichten 8 und 9 (die finanzielle und die politische Schicht). Während diese Schichten in der Praxis wohlbekannt sind, haben sie ins offizielle ISO/OSI-Referenzmodell noch keinen Eingang gefunden.

Übungen

3.1 [2] Rekapitulieren Sie kurz die Unterschiede zwischen Übertragungs-, Kommunikations- und Anwendungsprotokollen. Nennen Sie Beispiele für die jeweiligen Sorten. (Kennen Sie welche, die nicht aus dem TCP/IP-Umfeld stammen?)

3.2 [1] Was ist der wesentliche Unterschied zwischen den ISO/OSI-Schichten 2 und 3?

3.3 TCP/IP

3.3.1 Überblick

TCP/IP steht für *Transmission Control Protocol/Internet Protocol* und ist die heute am häufigsten eingesetzte Methode für den Datentransfer in Rechnernetzen, seien es nur zwei Rechner in einem lokalen Netz oder aber das ganze Internet. Bei TCP/IP handelt es sich nicht nur um ein einzelnes Protokoll, sondern um eine Fülle unterschiedlicher aufeinander aufbauender Protokolle mit teils sehr unterschiedlichen Aufgaben. Man spricht von einer »Protokollfamilie«.

Die Protokolle der TCP/IP-Protokollfamilie lassen sich zumindest ungefähr in das ISO/OSI-Schichtenmodell aus Bild 3.2 einordnen. Die wichtigsten sind hier kurz aufgelistet:

Netzzugangsschicht Ethernet, IEEE 802.11, PPP (dies sind strenggenommen keine TCP/IP-Protokolle)

Internetschicht IP, ICMP, ARP

Transportschicht TCP, UDP, …

Anwendungsschicht HTTP, DNS, FTP, SSH, NIS, NFS, LDAP, …

Um den Ablauf der Datenübertragung besser zu verstehen und Fehler, die hierbei auftreten, eingrenzen und auffinden zu können, ist es sehr nützlich, die Struktur der wichtigsten Protokolle und den Aufbau der dazugehörigen Protokolldateneinheiten zu kennen. Wir erklären im folgenden kurz die wichtigsten TCP/IP-Protokolle aus der Internet- und Transportschicht.

Übungen

3.3 [2] Welche anderen Protokolle aus der TCP/IP-Protokollfamile fallen Ihnen ein? Zu welchen der vier Schichten gehören sie?

3.3.2 Kommunikation von Ende zu Ende: IP und ICMP

IP IP stellt die Verbindung zwischen zwei Systemen her. Es ist als Protokoll der ISO/OSI-Schicht 3 dafür verantwortlich, dass die Daten durch das Internet den Weg vom Sender zum Empfänger finden. Der Haken an der Sache ist, dass dieser Weg weite Strecken umfassen kann, die aus diversen unabhängigen Abschnitten mit deutlich verschiedener Netzwerktechnik bestehen und die deutlich unterschiedliche Kommunikationsparameter aufweisen. Stellen wir uns vor, ein Benutzer »surft« zu Hause im Internet. Sein Rechner ist über ein analoges Modem und das Telefonnetz per PPP mit einem Einwahlrechner bei einem Provider verbunden, der die Verbindung ins eigentliche Internet herstellt. Die Web-Anfragen werden dann zum Beispiel über Glasfaserleitungen mit ATM um die halbe Welt geschickt, bis sie im Rechenzentrum einer Universität ankommen, wo die Daten per FDDI-basiertem Campusnetz zu einem Institutsrouter fließen, der sie dann an den über Ethernet angeschlossenen Web-Server leitet. Die Web-Seiten nehmen den umgekehrten Weg zurück. Alle diese verschiedenen Teilstrecken verwenden nicht nur unterschiedliche Netzwerktechnik, sondern auch unterschiedliche »lokale« Adressen – während bei der PPP-Übertragung überhaupt keine Adressierung nötig ist (es gibt ja nur zwei Kommunikationspartner), funktioniert ein Ethernet beispielsweise auf der Basis von 48 Bit breiten »MAC«-Adressen.

Provider

Eine der Leistungen von IP besteht darin, einen »globalen« Adressraum zur Verfügung zu stellen, der jedem am Internet beteiligten System eine eindeutige Adresse gibt, über die es identifiziert werden kann. Ferner sorgt es für die Wegleitung (engl. *routing*) von einem System zum anderen ohne Berücksichtigung der tatsächlich verwendeten Netzwerktechnik.

Adressraum

Wegleitung

IP ist ein **verbindungsloses Protokoll**, das heißt, im Gegensatz z. B. zum traditionellen Telefonnetz wird keine feste Verbindung (»Draht«) für die Kommunikation zweier Systeme zur Verfügung gestellt[1], sondern die zu übertragenden Daten werden in Häppchen, sogenannte **Datagramme**, eingeteilt, die unabhängig voneinander adressiert und zugestellt werden. Prinzipiell kann jedes Datagramm einen anderen Weg zum Empfänger nehmen als das vorige; dies macht IP unempfindlich gegen Ausfälle von Leitungen oder Vermittlungsrechnern, solange sich noch irgendein Weg vom Quell- zum Zielsystem finden läßt. IP gibt keine Garantie, dass alle abgeschickten Daten auch tatsächlich beim Zielsystem ankommen, und genausowenig wird garantiert, dass die Daten, die tatsächlich ankommen, in derselben Reihenfolge ankommen, in der sie abgeschickt wurden. Es ist die Aufgabe »höhergelegener« Protokolle, hier für Ordnung zu sorgen, falls die Anwendung das erfordert.

verbindungsloses Protokoll

Datagramme

Stellen Sie sich vor, Sie möchten Ihrer Tante in Australien einen langen Text mit der Post schicken. Wenn Sie das »à la IP« machen wollten, würden Sie den Text auf eine Menge von einzelnen Postkarten schreiben. Die Chancen stehen gut, dass Ihre Postkarten auf dem Weg nach Känguru-Land durcheinander geraten und der dortige Postbote sie nicht genau in derselben Reihenfolge in den Briefkasten Ihrer Tante steckt, in der Sie sie hier eingeworfen haben. Es ist auch durchaus möglich, dass die eine oder andere Postkarte irgendwo liegenbleibt oder ganz verlorengeht.

Warum ist das ein Vorteil? Das traditionelle Telefonnetz mit seinen von Ende zu Ende durchgeschalteten Drahtverbindungen war sehr anfällig gegenüber Störungen – fiel irgendein Leitungsabschnitt aus, brach das Gespräch komplett zusammen und musste neu aufgebaut werden (im Zeitalter der Handvermittlung eine größere Sache). Ergibt sich bei verbindungsloser Übertragung eine Störung oder Unterbrechung, kann das Netz für spätere Datagramme alternative Routen suchen, die die gestörte Stelle umgehen. Verfahren wie TCP erlauben es, zu erkennen, welche Daten durch die Störung verloren gegangen sind, und sorgen dafür, diese nochmals zu versenden.

[1] Auch das Telefonnetz – im Fachjargon POTS (für *plain old telephone system*) genannt – funktioniert längst nicht mehr so.

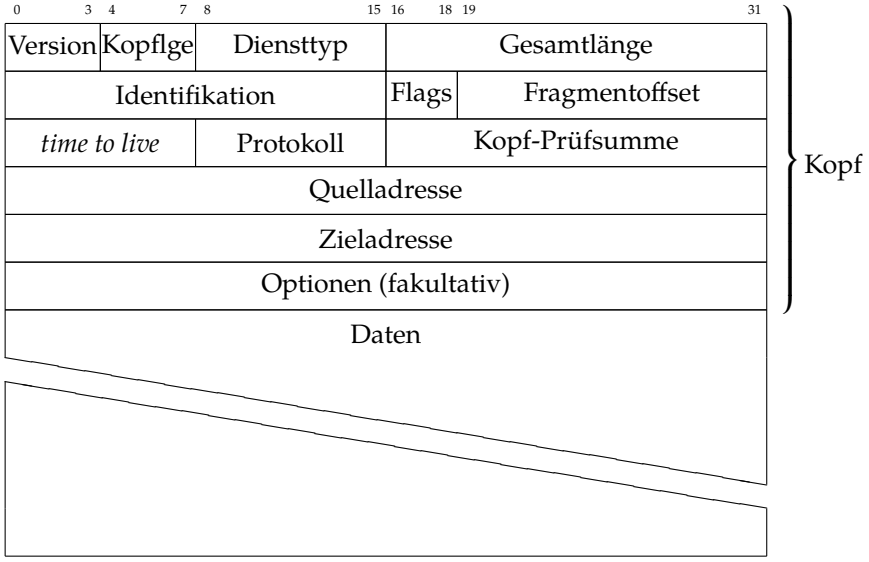


Bild 3.3: Aufbau eines IP-Datagramms. Jede Zeile entspricht 32 Bit.

Fragmentierung

Außerdem kümmert IP sich um **Fragmentierung**. IP-Datagramme dürfen bis zu 65535 Bytes lang sein, aber bei den meisten Transportprotokollen sind nur wesentlich kürzere Protokolldateneinheiten erlaubt – bei Ethernet beispielsweise nur bis zu 1500 Bytes. Längere Datagramme müssen darum »fragmentiert« übertragen werden – am Anfang einer entsprechenden Teilstrecke wird das Datagramm auseinandergenommen, in nummerierte Fragmente aufgeteilt und später wieder zusammengesetzt. IP sorgt dafür, dass nur solche Datagramme als offiziell empfangen gelten, in denen kein Fragment fehlt.

Die offizielle Spezifikation von IP ist [RFC0791]. Lesen müssen Sie das nicht, aber es hilft vielleicht gegen Schlafstörungen.

Bild 3.3 zeigt den Aufbau eines IP-Datagramms. Zumindest zwei Felder sollten wir noch kurz erklären:

- Die *time to live* (oder TTL) gibt die maximale »Lebensdauer« des Datagramms an. Sie wird vom Absender des Datagramms gesetzt und von jeder Station, die das Datagramm auf dem Weg zum Empfänger durchläuft, um 1 vermindert. Wenn die TTL den Wert 0 erreicht, wird das Datagramm verworfen und der Absender benachrichtigt. Das Ziel ist die Vermeidung von »fliegenden Holländern« – Datagrammen, die aufgrund von Fehlern bei der Wegleitung ziellos im Internet umherirren, ohne jemals ihren Bestimmungsort zu erreichen. (Gerne als Standard verwendet werden Werte wie 64, die bei der aktuellen Ausdehnung des Internet absolut jenseits von Gut und Böse sind.)
- Der Diensttyp (engl. *type of service*, TOS) gibt an, welche Dienstgüte für das Datagramm erwünscht ist. Hier können Sie theoretisch neben einer von sieben Vorrangstufen (die ignoriert werden) noch die Attribute »niedrige Verzögerung«, »hoher Durchsatz«, »hohe Zuverlässigkeit« und »niedriger Preis« angeben. Ob das allerdings in irgendeiner Form für einen Unterschied bei der Zustellung sorgt, ist äußerst zweifelhaft, da diese Angaben nicht verbindlich sind und Router sie gerne ignorieren. (Wenn das nicht so wäre, würden wahrscheinlich alle Datagramme *alle* diese wünschenswerten Optionen einschalten.)

ICMP Ein weiteres wichtiges Protokoll ist das *Internet Control Message Protocol* [RFC0792], kurz ICMP genannt (siehe Bild 3.4). Es wird zur Netzverwaltung

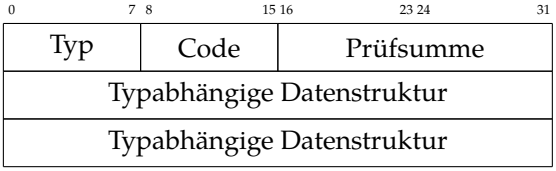


Bild 3.4: Aufbau eines ICMP-Pakets

benutzt und dazu, Probleme mit dem Netz zu melden, etwa wenn eine Verbindung nicht zustande kommt oder ein Teilnetz nicht erreichbar ist. Das sehr bekannte Programm `ping` zum Beispiel verwendet zwei spezielle ICMP-Meldungen (`echo request` und `echo reply`). Das ICMP-Paket wird als »Daten« in einem IP-Datagramm transportiert und enthält abhängig vom Code verschiedene weitere Datenfelder.

IP und Übertragungsprotokolle Damit wir über IP Daten ungeachtet der tatsächlich verwendeten Netzwerktechnik übertragen können, müssen wir von Fall zu Fall definieren, wie IP-Datagramme über das betreffende Netz – sei es Ethernet, PPP über eine analoge Telefonleitung, ATM, WLAN, … – transportiert werden.

In einem Ethernet zum Beispiel sind alle Stationen zumindest konzeptuell an ein gemeinsames Medium angeschlossen – im »klassischen« Ethernet an ein einziges langes Koaxialkabel, das von einer Station zur nächsten läuft, heute zumeist mit Twisted-Pair-Kabeln an einen gemeinsamen Sternverteiler oder Switch. Alles, was eine Station sendet, wird von allen anderen Stationen empfangen, wobei diese sich in der Regel nur diejenigen Protokolldateneinheiten heraussuchen, die wirklich für sie gemeint sind (heutzutage helfen auch die Switches, indem sie den Verkehr »vorsortieren«). Senden zwei Stationen gleichzeitig, kommt es zu einer Kollision, die dadurch behoben wird, dass beide ihren Sendevorgang abbrechen, eine zufällige Zeitspanne warten und es dann wieder versuchen. Ein solches gemeinsames Medium im Ethernet heißt auch »Segment«.

Kollision

Segment

Jede Ethernet-Schnittstelle hat eine eindeutige Adresse, die 48 Bit lange »MAC-Adresse« (kurz für *medium access control* – Medienzugangssteuerung). Protokolldateneinheiten im Ethernet, die sogenannten Frames, können Sie entweder an spezielle andere Stationen im selben Segment verschicken, indem Sie deren MAC-Adresse als Empfänger eintragen – das Frame wird zwar von allen Stationen gesehen, aber von allen außer der adressierten Station ignoriert –, oder als Rundruf (engl. *broadcast*) an *alle* Stationen im Segment senden.

MAC-Adresse

Frames

Ethernet-Netzwerkkarten unterstützen in der Regel auch den sogenannten *promiscuous mode*, in dem sie *alle* Frames – auch die, die sie eigentlich gar nichts angehen – ans System weiterreichen. Dies ist die Basis für interessante Software wie Netzwerkanalyseprogramme und Cracker-Werkzeuge.

Dies macht man sich für die Integration von IP und Ethernet zunutze. Will eine Station (nennen wir sie einmal *A*) mit einer anderen Station (*B*) kommunizieren, deren IP-Adresse sie kennt, deren MAC-Adresse ihr aber nicht bekannt ist, fragt sie zunächst per Ethernet-Broadcast alle angeschlossenen Stationen:

> Station *A*: Wer hat hier die IP-Adresse `203.177.8.4`?
> Station *B*: Ich, und meine MAC-Adresse ist `00:06:5B:D7:30:6F`

Dieser Vorgang folgt dem *Address Resolution Protocol* (ARP, [RFC0826]). Hat Station *A* die MAC-Adresse von Station *B* erhalten, speichert sie diese für eine gewisse Zeit in ihrem »ARP-Cache«, um die Anfrage nicht für jedes Frame wiederholen zu müssen; IP-Datagramme an Stationen, deren IP- und MAC-Adressen im ARP-Cache stehen, können Sie auf Ethernet-Ebene direkt adressieren, indem Sie sie als »Nutzdaten« in Ethernet-Frames einbetten. Auf den ARP-Cache können Sie mit

ARP

ARP-Cache

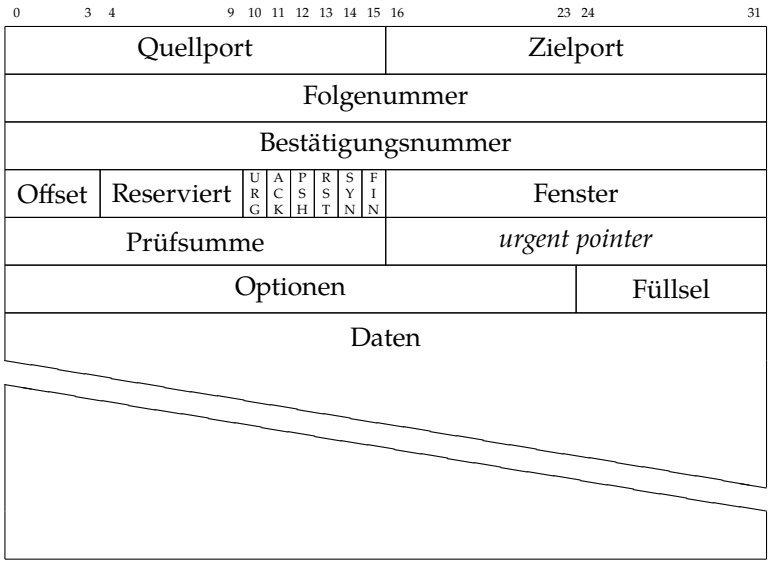


Bild 3.5: Aufbau eines TCP-Segments

dem Kommando `arp` zugreifen – nicht nur lesen, sondern auch Einträge schreiben. Die Ausgabe von `arp` kann beispielsweise so aussehen:

```
# arp
Address              Hwtype  Hwaddress           Flags  Mask  Iface
server.example.org   ether   00:50:DB:63:62:CD   C             eth0
```

Routing

Datagramme an IP-Adressen, die nicht zu Stationen im selben Ethernet-Segment gehören, müssen **geroutet** werden (Abschnitt 3.4.2).

Übungen

3.4 [3] Schätzen Sie die minimale TTL, die notwendig ist, damit Sie von Ihrem Rechner aus alle anderen Stationen auf dem Internet erreichen können. Wie könnten Sie die minimale TTL ermitteln, die zum Erreichen einer gegebenen Station nötig ist? Ist diese Zahl eine Konstante?

3.3.3 Die Basis für Dienste: TCP und UDP

TCP Das »Transmission Control Protocol« (TCP) ist ein zuverlässiges, verbindungsorientiertes Protokoll, das u. a. in [RFC0793] definiert ist. Im Gegensatz zum verbindungslosen IP kennt TCP Operationen zum Verbindungsauf- und -abbau, mit denen zumindest eine »virtuelle« Verbindung zwischen Quell- und Zielsystem geschaltet wird – da TCP-Daten wie alle anderen Daten auch über IP übertragen werden, erfolgt die tatsächliche Datenübertragung nach wie vor verbindungslos und unzuverlässig. TCP erreicht Zuverlässigkeit, indem die Gegenstelle die Ankunft jedes Pakets (im TCP-Jargon »Segment«) bestätigt. Jede der beiden kommunizierenden Stationen versieht ihre Segmente mit Folgenummern (engl. *sequence numbers*), die die Gegenstelle in einem ihrer nächsten Segmente als »angekommen« quittiert. Kommt innerhalb einer gewissen definierten Zeitspanne keine solche Quittung, versucht die sendende Station, das Segment erneut zu schicken, um es vielleicht diesmal bestätigt zu bekommen. Damit die Effizienz darunter nicht zu sehr leidet, wird ein *sliding-window*-Protokoll eingesetzt, so dass eine gewisse Anzahl von Segmenten gleichzeitig unbestätigt bleiben darf. Trotzdem ist TCP deutlich langsamer als IP.

Folgenummer

Eigentlich basieren die Bestätigungen von TCP auf Oktetten (vulgo Bytes),

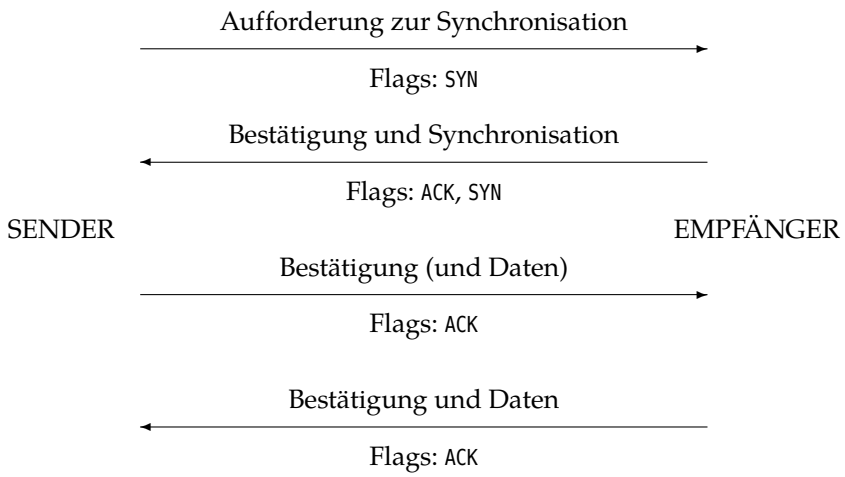


Bild 3.6: Aufbau einer TCP-Verbindung: Der Drei-Wege-Handshake

nicht auf Segmenten – aber für unsere Zwecke ist der Unterschied zunächst akademisch.

Jedes TCP-Segment hat einen mindestens 20 Byte großen Kopf (Bild 3.5) zusätzlich zum IP-Kopf. (Sie erinnern sich: das TCP-Segment *inklusive* TCP-Kopf gilt aus der Sicht von IP, dem Protokoll der darunterliegenden Schicht, als »Daten«.) Anhand einer Prüfsumme können die Daten auf Fehler überprüft werden. Jedes System unterstützt viele unabhängige, gleichzeitige TCP-Verbindungen, zwischen denen anhand von **Portnummern** unterschieden wird.

Portnummern

Die Kombination aus einer IP-Adresse und einer Portnummer zusammen mit der IP-Adresse und Portnummer der Gegenstelle bezeichnet man auch als *socket*. (Derselbe TCP-Port auf einer Station darf gleichzeitig an mehreren TCP-Verbindungen mit unterschiedlichen Gegenstellen – definiert durch IP-Adresse und Portnummer – beteiligt sein.)

Der Aufbau der virtuellen Verbindung erfolgt über den sogenannten **Drei-Wege-Handshake** (engl. *three-way handshake*, siehe Bild 3.6). Im Drei-Wege-Handshake einigen die Kommunikationspartner sich über die zu verwendenden Folgenummern. Hierbei spielen zwei **Flags** im TCP-Kopf, `SYN` und `ACK`, eine entscheidende Rolle. Im ersten Datensegment, das der Absender dem Empfänger schickt, ist das `SYN`-Flag gesetzt und das `ACK`-Flag nicht. Ein solches Segment signalisiert den Verbindungswunsch. Der Empfänger bestätigt das mit einem TCP-Segment, bei dem sowohl das `SYN`- als auch das `ACK`-Flag gesetzt sind. Dieses Segment bestätigt der Absender wiederum mit einem Segment, das das `ACK`-Flag, aber *nicht* das `SYN`-Flag gesetzt hat. An diesem Punkt steht die Verbindung. Darauffolgende TCP-Segmente haben ebenfalls nur noch das `ACK`-Flag gesetzt. – Am Ende der Kommunikation wird die Verbindung über einen Zwei-Wege-Handshake mit dem `FIN`-Flag wieder abgebaut.

Drei-Wege-Handshake

Flags

Am Anfang der Verbindung müssen die beiden Stationen sich einig werden, aber abgebaut werden kann die Verbindung auch unilateral. Das ist sogar dringend nötig, damit Kommandos wie das folgende funktionieren:

```
$ cat bla | ssh blue sort
```

Hier starten wir über die Secure Shell (siehe Kapitel 10) auf dem Rechner `blue` das Kommando `sort` und versorgen es über die Standardeingabe mit

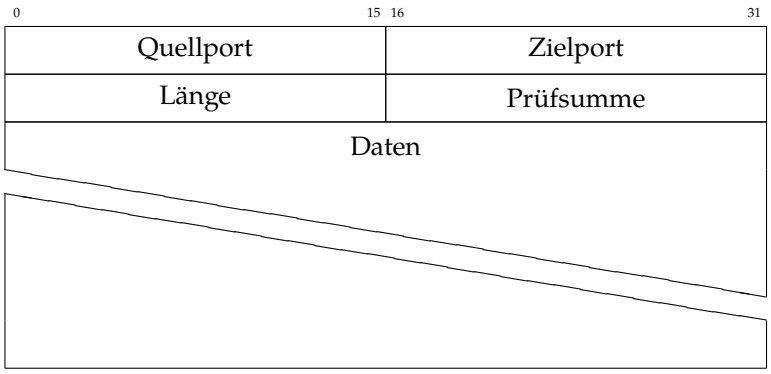


Bild 3.7: Aufbau eines UDP-Datagramms

Daten. (`ssh` liest seine Standardeingabe auf dem lokalen Rechner, leitet sie über das Netz an den entfernten Rechner weiter und übergibt sie dort dem Programm `sort` auf dessen Standardeingabe.) `sort` funktioniert aber so, dass es seine Standardeingabe bis zum Ende liest, dann erst die gelesenen Daten sortiert und sie auf die Standardausgabe schreibt, die dann per `ssh` wieder auf den lokalen Rechner (und den Bildschirm) geleitet wird. – Das Problem besteht nun darin, dass die `ssh` dem entfernt laufenden `sort` signaliseren können muss, dass die Standardeingabe fertig gelesen wurde und es mit dem Sortieren und der Ausgabe anfangen kann. Das geschieht dadurch, dass die Verbindung »zum« entfernten Rechner abgebaut wird. Derjenige Teil der Verbindung, der »vom« entfernten Rechner liest, bleibt jedoch erhalten und kann die Ausgabe von `sort` zurücktransportieren – würde ein Verbindungsabbau automatisch für beide Richtungen gelten, könnte diese Anwendung nicht funktionieren.

Nach einem unilateralen Verbindungsabbau fließen natürlich nach wie vor Daten in beide Richtungen zwischen den Stationen, denn die Station, die die Verbindung abgebaut hat, muss die Daten, die sie über den noch stehenden Teil der Verbindung empfängt, ja auch bestätigen. Nur Nutzdaten kann sie nicht mehr über die Verbindung schicken.

UDP Im Gegensatz zu TCP ist das »User Datagram Protocol« (UDP) [RFC0768] ein verbindungsloses und unzuverlässiges Protokoll. Tatsächlich ist es nicht viel mehr als »IP mit Ports«, denn wie bei TCP können auf einer Station maximal 65535 Kommunikationsendpunkte unterschieden werden (UDP und TCP können dieselbe Portnummer gleichzeitig für unterschiedliche Zwecke verwenden). Bei UDP entfällt der Verbindungsaufbau von TCP genau wie die Bestätigungen, so dass das Protokoll viel »schneller« ist – allerdings um den Preis, dass wie bei IP Daten verloren gehen oder durcheinander geraten können.

UDP verwendet man da, wo entweder nur wenige Daten übertragen werden müssen, so dass die Kosten des Verbindungsaufbaus bei TCP im Vergleich sehr hoch sind – Stichwort DNS –, oder dort, wo es nicht auf jedes Bit ankommt, aber Zeitverzögerungen unerwünscht sind. Bei Internet-Telefonie oder Videoübertragung machen verlorene Datagramme sich höchstens durch Knacklaute oder Schneegriesel im Bild bemerkbar; eine längere Kunstpause, wie sie bei TCP durchaus mal vorkommen kann, wäre im Vergleich viel störender.

Ports **Ports** TCP und UDP unterstützen das Konzept von Ports, über die ein System mehr als eine Netzwerkverbindung gleichzeitig verwalten kann (gut, bei UDP gibt es keine »Verbindungen«, aber trotzdem …). Getrennt für TCP und UDP gibt es

```
# Network services, Internet style

echo       7/tcp
echo       7/udp
discard    9/tcp    sink null
discard    9/udp    sink null
systat     11/tcp   users
daytime    13/tcp
daytime    13/udp
netstat    15/tcp
qotd       17/tcp   quote
chargen    19/tcp   ttytst source
chargen    19/udp   ttytst source
ftp-data   20/tcp
ftp        21/tcp
fsp        21/udp   fspd
ssh        22/tcp                  # SSH Remote Login Protocol
ssh        22/udp                  # SSH Remote Login Protocol
telnet     23/tcp
smtp       25/tcp   mail
<<<<<
```

Bild 3.8: Die Datei `/etc/services` (Auszug)

jeweils 65536 Ports, die allerdings nicht alle sinnvoll benutzt werden können: Die Portnummer 0 ist ein Signal an die TCP/IP-Implementierung des Systems, einen ansonsten unbenutzten Port auszusuchen.

Die meisten Ports stehen den Anwendern des Systems frei zur Verfügung, aber diverse Ports sind fest bestimmten Diensten zugeordnet. Man spricht von den ***well-known ports*** und den ***registered ports***. So ist zum Beispiel festgelegt, dass der TCP-Port 25 auf einem System für dessen Mail-Server reserviert ist, der dort auf Verbindungen gemäß dem *Simple Mail Transfer Protocol* (SMTP) wartet. Entsprechend ist der TCP-Port 21 dem *File Transfer Protocol*-Server (FTP) vorbehalten und so weiter. Diese Zuordnungen werden in regelmäßigen Abständen von der IANA veröffentlicht und sind zum Beispiel unter `http://www.iana.org/assignments/port-numbers` zu finden.

well-known ports

registered ports

Die *well-known ports* sind laut IANA die Ports von 0 bis 1023, die *registered ports* diejenigen von 1024 bis 49151. Wenn Sie ein Programm in Umlauf bringen wollen, das einen neuen Dienst anbietet, sollten Sie sich von der IANA eine oder mehrere Portnummern zusprechen lassen.

Die übrigen Ports – von 49152 bis 65535 – heißen im IANA-Jargon *dynamic and/or private ports*. Diese kommen für die Client-Seite von Verbindungen in Frage (es ist unwahrscheinlich, dass Ihr Rechner über 16.000 Verbindungen zu TCP-Servern gleichzeitig halten muss) oder für die Implementierung »privater« Server.

Die IANA neigt dazu, eine für ein TCP-basiertes Protokoll reservierte Portnummer auch für UDP zu reservieren, selbst wenn das betreffende TCP-Protokoll mit UDP keinen Sinn ergibt, und umgekehrt. Port 80 ist zum Beispiel sowohl als TCP- wie auch als UDP-Port für HTTP reserviert, auch wenn UDP-basiertes HTTP kein aktuell interessantes Thema ist. Man läßt sich so Ellbogenfreiheit für künftige Erweiterungen.

Auf einem Linux-System steht eine Zuordnungstabelle in der Datei `/etc/services` (Bild 3.8). Diese Zuordnung wird zum Beispiel vom Internet-Daemon

Tabelle 3.1: Gängige Anwendungsprotokolle auf TCP/IP-Basis

Port	K-Prot	Name	Bedeutung
20	TCP	FTP	Dateiübertragung (Datenverbindungen)
21	TCP	FTP	Dateiübertragung (Steuerverbindungen)
22	TCP	SSH	Sicheres (authentisiertes und verschlüsseltes) Anmelden auf entfernten Rechnern; sichere Dateiübertragung
23	TCP	TELNET	Anmelden auf entfernten Rechnern (unsicher und veraltet)
25	TCP	SMTP	E-Mail-Übertragung
53	UDP/TCP	DNS	Namens- und Adressenauflösung und verwandte Verzeichnisdienste
80	TCP	HTTP	Ressourcenzugriff im World Wide Web
110	TCP	POP3	Zugriff auf entfernte E-Mail-Postfächer
123	UDP/TCP	NTP	Network Time Protocol (Zeitsynchronisation)
137	UDP	NETBIOS	NetBIOS-Namensdienst
138	UDP	NETBIOS	NetBIOS-Datagrammdienst
139	TCP	NETBIOS	NetBIOS-Sitzungsdienst
143	TCP	IMAP	Zugriff auf Mail auf einem entfernten Server
161	UDP	SNMP	Netzwerkverwaltung
162	UDP	SNMP	Traps für SNMP
389	TCP	LDAP	Verzeichnisdienst
443	TCP	HTTPS	HTTP über SSL (authentisiert/verschlüsselt)
465	TCP	SSMTP	SMTP über SSL (veraltet – nicht benutzen!)*
514	UDP	Syslog	Protokolldienst
636	TCP	LDAPS	LDAP über SSL (authentisiert/verschlüsselt)*
993	TCP	IMAPS	IMAP über SSL (authentisiert/verschlüsselt)*
995	TCP	POP3S	POP3 über SSL (authentisiert/verschlüsselt)*

* Diese Dienste können auch über Verbindungen abgewickelt werden, die zunächst unverschlüsselt aufgebaut und dann zu authentisierten und verschlüsselten Verbindungen »aufgewertet« werden.

(`inetd` oder `xinetd`) oder von der C-Bibliotheksfunktion `getservbyname()` verwendet, um zu einem gegebenen Dienstnamen den passenden Port zu finden.

Sie können `/etc/services` selbst ändern, etwa um Ihre selbsterfundenen Dienste zu unterstützen. Achten Sie aber auf Aktualisierungen der Datei durch Ihre Distribution.

reservierte Ports

Die Ports 0 bis 1023 sind auf unixartigen Systemen reserviert, das heißt, nur `root` darf sie öffnen. Dies ist eine Sicherheitsvorkehrung dagegen, dass beliebige Benutzer zum Beispiel einen eigenen Webserver auf einem ansonsten unbenutzten Port 80 starten und damit offiziell wirken.

3.3.4 Die wichtigsten Anwendungsprotokolle

Im vorigen Abschnitt haben wir das Konzept eines »Diensts« angesprochen. Während Kommunikationsprotokolle wie TCP und UDP sich damit befassen, Daten von einer Station zur anderen zu befördern, basieren »Dienste« in der Regel auf Anwendungsprotokollen, die den per Kommunikationsprotokoll ausgetauschten Daten eine Bedeutung zuordnen. Wenn Sie zum Beispiel eine E-Mail per SMTP verschicken, nimmt Ihr Rechner mit dem entfernten SMTP-Server Kontakt auf (über TCP auf Port 25), identifiziert sich, schickt Ihre Adresse sowie die des oder der Adressaten und danach die eigentliche Mail – jeweils nach Aufforderung durch den entfernten Server. Die Details dieser Konversation regelt das Anwendungsprotokoll SMTP.

»Dienste« und »Protokolle« sind nicht ganz dasselbe. Ein »Dienst« ist etwas, wofür Sie den Computer verwenden möchten, etwa E-Mail, Web-Zugriff

oder Drucken auf einem entfernten Druckserver. Für viele Dienste gibt es im Internet »kanonische« Protokolle, die sich einfach anbieten – für E-Mail gibt es zum Beispiel kaum Alternativen zu SMTP –, aber manche Dienste verwenden auch dasselbe unterliegende Protokoll wie andere. Web-Zugriff beispielsweise erfolgt in der Regel über HTTP und der Zugriff auf einen entfernten Druckserver über das »Internet Printing Protocol« (IPP). Wenn Sie genau nachschauen, werden Sie aber feststellen, dass IPP, so wie es heute verwendet wird, auch nichts anderes ist als glorifiziertes HTTP. Der einzige Unterschied ist, dass HTTP den TCP-Port 80 verwendet und IPP den TCP-Port 631.

Tabelle 3.1 zeigt eine Zusammenfassung einiger wichtiger Anwendungsprotokolle. Einige davon werden uns später in dieser Unterlage noch einmal begegnen; mit weiteren befassen sich andere Linup-Front-Schulungsunterlagen.

Schlechte Nachrichten für LPIC-1-Kandidaten: Das LPI möchte, dass Sie die Portnummern aus Tabelle 3.1 und die jeweiligen dazugehörigen Dienste auswendig wissen (LPI-Lernziel 109.1). Viel Spaß beim Büffeln.

3.4 Adressen, Wegleitung und Subnetting

3.4.1 Grundlagen

Jede Netzwerk-Schnittstelle eines Systems in einem TCP/IP-Netz hat mindestens eine IP-Adresse. Als »Schnittstelle« wird in diesem Fall der Teil eines Systems bezeichnet, der in der Lage ist, IP-Datagramme zu senden und zu empfangen. Ein einzelnes System kann mehr als eine solche Schnittstelle haben und hat dann meist auch mehr als eine IP-Adresse. Mit

```
$ /sbin/ifconfig
```

oder

```
$ /sbin/ip addr show
```

können Sie die konfigurierten Schnittstellen bzw. Netzwerkgeräte auflisten.

IP-Adressen

IP-Adressen sind 32 Bit lang und werden normalerweise als *dotted quads* notiert – man betrachtet sie als Folge von vier 8-Bit-Zahlen, die man dezimal als Werte zwischen 0 und 255 hinschreibt, etwa als »`203.177.8.4`«[2]. Jede IP-Adresse wird weltweit eindeutig vergeben und bezeichnet eine Station in einem bestimmten Teilnetz des Internet. Dazu werden IP-Adressen in einen Netzwerk- und einen Stationsanteil aufgeteilt. Der Netzwerk- und Stationsanteil ist variabel und kann der Anzahl der in einem Netz benötigten Stationsadressen angepasst werden. Wenn der Stationsanteil n Bit beträgt, bleiben für den Netzwerkanteil $32 - n$ Bit. Die Verteilung dokumentiert die **Netzmaske**, die für jedes Bit der IP-Adresse, das zum Netzwerkanteil gehört, eine binäre 1 und für jedes Bit des Stationsanteils eine binäre 0 enthält. Die Netzmaske wird entweder als *dotted quad* oder – heutzutage oft – einfach als Anzahl der Einsen notiert. »`203.177.8.4/24`« wäre also eine Adresse in einem Netz mit der Netzmaske »`255.255.255.0`«.

Netzmaske

Nehmen wir als Beispiel ein Netzwerk mit 28 Geräten an. Die nächsthöhere Potenz von 2 ist $32 = 2^5$. Das bedeutet, dass für die Numerierung der verschiedenen Systeme 5 Bits gebraucht werden. Die restlichen 27 Bits (32 – 5) identifizieren das Netzwerk und sind bei jedem System in diesem Netzwerk gleich. Die Netzmaske ist `255.255.255.224`, denn im letzten *quad* sind die obersten 3 Bits – die mit den Werten 128, 64 und 32, zusammen 224 –, gesetzt.

[2] Es ist übrigens durchaus legitim und wird von den meisten Programmen verstanden, eine IP-Adresse als ausmultiplizierte Dezimalzahl anzugeben – in unserem Beispiel statt `203.177.8.4` etwa 3417376772. Dies ist die Basis für »Trick-URLs« der Form `http://www.microsoft.com@3417376722/bla.html`.

Tabelle 3.2: Beispiel für Adressenvergabe

Bedeutung	IP-Adresse binär	IP-Adresse dezimal
Netzmaske	11111111 11111111 11111111 11100000	255.255.255.224
Netzwerkadresse	11001011 10110001 00001000 00000000	203.177.8.0
Stationsadressen	11001011 10110001 00001000 00000001	203.177.8.1
⋮	⋮	⋮
	11001011 10110001 00001000 00011110	203.177.8.30
Broadcast-Adresse	11001011 10110001 00001000 00011111	203.177.8.31

Netzwerkadresse
Broadcast-Adresse

Die erste und die letzte IP-Adresse in einem Netzwerk werden verabredungsgemäß für spezielle Zwecke reserviert: Die erste Adresse (Stationsanteil nur binäre Nullen) ist die **Netzwerkadresse**, die letzte Nummer (Stationsanteil nur binäre Einsen) die **Broadcast-Adresse**. Im obigen Beispiel wäre also `203.177.8.0` die Netzwerkadresse und `203.177.8.31` die Broadcast-Adresse. Für die Stationen stehen dann die Nummern von 1 bis 30 zur Verfügung (Tabelle 3.2).

Die Adresse `255.255.255.255` ist eine Broadcast-Adresse, aber nicht für das gesamte Internet, sondern für das lokale Netzsegment (also zum Beispiel alle Stationen auf demselben Ethernet). Diese Adresse wird benutzt, wenn keine genauere Adresse bekannt ist, etwa wenn eine Station sich über DHCP eine IP-Adresse und eine Netzmaske holen möchte.

3.4.2 Wegleitung

Wegleitung (engl. *routing*) dient dazu, IP-Datagramme, die nicht direkt im lokalen Netz zugestellt werden können, an die richtige Adresse zu schicken[3]. Tatsächlich können Sie argumentieren, dass Wegleitung die zentrale Eigenschaft ist, die TCP/IP von »Spielzeugprotokollen« wie NetBEUI und Appletalk unterscheidet und damit das Internet, so wie wir es kennen, erst möglich gemacht hat.

Wegleitung greift da, wo der Empfänger eines IP-Datagramms nicht im selben Netz zu finden ist wie der Absender. Feststellen kann die sendende Station das (natürlich) anhand der IP-Adresse der gewünschten Zielstation, indem sie den Teil der Zieladresse betrachtet, der von ihrer eigenen Netzmaske »abgedeckt« wird, und überprüft, ob er mit ihrer eigenen Netzwerkadresse übereinstimmt. Wenn das der Fall ist, ist der Empfänger »lokal« und kann direkt erreicht werden (Abschnitt 3.3.2 auf Seite 53).

Routing-Tabelle

Kann der Empfänger nicht direkt erreicht werden, konsultiert die Station (jedenfalls, wenn sie ein Linux-Rechner ist) eine Routing-Tabelle, die zumindest ein *default gateway* ausweisen sollte, also eine Station, die sich um die Weiterleitung nicht direkt zustellbarer Datagramme kümmert. (Diese Station muss in aller Regel selber direkt erreichbar sein.) Eine solche Station heißt »Router« und ist entweder selbst ein Computer oder ein spezielles für diese Funktion ausgelegtes Gerät.

Der Router geht grundsätzlich ganz analog vor wie eben beschrieben: Er verfügt über verschiedene Netzwerkschnittstellen, die jeweils eine Adresse und eine Netzmaske zugeordnet bekommen haben, und kann Datagramme unmittelbar an Stationen zustellen, die sich über die Netzmasken seiner Netzwerkschnittstellen als in einem »seiner« Netze befindlich identifizieren lassen. Für alles Weitere werden wiederum direkt erreichbare Stationen herangezogen, die als Router fungieren und so fort.

[3] Vorausgesehen wurde das bereits im Alten Testament, und zwar aus der Sicht des Datagramms: »Er führet mich auf rechter Straße um seines Namens willen.« (Ps 23:3) Denn im Internet gibt es für Sie kaum eine bessere Methode, das Ansehen Ihres Namens zu ramponieren, als eine kapitale Routing-Fehlkonfiguration …

Tabelle 3.3: Traditionelle IP-Netzklassen

Klasse	Netzanteil	Anzahl der Netze	Stationen pro Netz	Adressen
Class A	8 Bit	128 – 126 nutzbar	16.777.214 ($2^{24} - 2$)	`0.0.0.0` – `127.255.255.255`
Class B	16 Bit	16.384 (2^{14})	65.534 ($2^{16} - 2$)	`128.0.0.0` – `191.255.255.255`
Class C	24 Bit	2.097.152 (2^{21})	254 ($2^8 - 2$)	`192.0.0.0` – `223.255.255.255`
Class D	-	-	-	`224.0.0.0` – `239.255.255.255`
Class E	-	-	-	`240.0.0.0` – `254.255.255.255`

Im wirklichen Leben können Routingtabellen um einiges komplexer sein. Es ist zum Beispiel möglich, Datagramme, die an bestimmte Stationen oder Netze gerichtet sind, gezielt über Router zu leiten, die nicht das *default gateway* sind.

Eine wichtige Beobachtung ist, dass eine Station (PC oder Router) normalerweise nur über den direkt nächsten Schritt der Wegleitung (man sagt auch »Hop«) entscheidet, anstatt den kompletten Weg vom ursprünglichen Absender des Datagramms bis zum Empfänger vorzugeben. Das heißt, es liegt in der Hand jedes Routers zwischen Absender und Empfänger, denjenigen nächsten Hop auszuwählen, den er für am sinnvollsten hält. Gut konfigurierte Router gleichen sich mit ihren »Nachbarn« ab und können Informationen über die Netzauslastung und möglicherweise bekannte Blockagen anderswo im Netz in ihre Wegleitungsentscheidungen einfließen lassen. Für unsere Zwecke führt eine detaillierte Diskussion dieser Thematik allerdings zu weit.

Tatsächlich ist es auch möglich, dass ein Datagramm den kompletten Weg vorgibt, den es zu seinem Ziel durchlaufen möchte. Das heißt dann *source routing*, ist eigentlich verpönt und wird heutzutage von großen Teilen der Netzinfrastruktur völlig missachtet, da es einerseits mit der Idee der dynamischen Lastverteilung kollidiert und andererseits Vehikel für Sicherheitsprobleme sein kann.

3.4.3 IP-Netzklassen

Die Menge der IP-Adressen von `0.0.0.0` bis `255.255.255.255` wurde traditionell in mehrere **Netzklassen** eingeteilt, die als »Class A«, »Class B« und »Class C« bezeichnet werden. Netzklassen

Es gibt auch noch »Class D« (Multicast-Adressen) und »Class E« (für experimentelle Zwecke), diese sind jedoch für die Vergabe von IP-Adressen für Endgeräte von geringem Interesse.

Die Klassen A bis C unterscheiden sich durch ihre Netzmaske, unter dem Strich also in der Anzahl der pro Klasse möglichen Netze und der Anzahl der in einem solchen Netz möglichen Stationen. Während eine Class-A-Adresse einen Netzanteil von 8 Bit hat, hat eine Class-B-Adresse einen von 16 Bit und eine Class-C-Adresse einen von 24 Bit. Jeder der Netzklassen war ein fester Bereich der möglichen IP-Adressen zugeordnet (Tabelle 3.3)

Aufgrund der zunehmenden Verknappung von IP-Adressen wurde in den 1990er Jahren die Aufteilung der IP-Adressen in die drei Adressklassen aufgegeben. Inzwischen verwendet man klassenlose Wegleitung (*classless inter-domain routing*, CIDR) nach [RFC1519]. Während die Grenze zwischen Netzwerk- und Stationsadresse bei der »alten« Aufteilung nur an drei verschiedenen Positionen liegen konnte, ist es gemäß CIDR möglich, beliebige Netzmasken zu vergeben und so einerseits die Größe der einem Anwender (meist einem Provider) zur Verfügung gestellten Adressbereichs feiner zu steuern und andererseits die »Explosion« der Wegleitungstabellen zu vermeiden. Eine Installation mit sechzehn klassenlose Wegleitung

Tabelle 3.4: Beispiel für Subnetting

Bedeutung	IP-Adresse binär	dezimal
Netzmaske	11111111 11111111 11111111 11110000	255.255.255.240
Netzwerkadresse (1)	11001011 10110001 00001000 00000000	203.177.8.0
Stationsadressen (1)	11001011 10110001 00001000 00000001	203.177.8.1
⋮	⋮	⋮
	11001011 10110001 00001000 00001110	203.177.8.14
Broadcast-Adresse (1)	11001011 10110001 00001000 00001111	203.177.8.15
Netzwerkadresse (2)	11001011 10110001 00001000 00010000	203.177.8.16
Stationsadressen (2)	11001011 10110001 00001000 00010001	203.177.8.17
⋮	⋮	⋮
	11001011 10110001 00001000 00011110	203.177.8.30
Broadcast-Adresse (2)	11001011 10110001 00001000 00011111	203.177.8.31

aneinandergrenzenden »Class-C«-Netzen (Netzmaske »/24«) kann so aus der Sicht der Wegleitung als ein Netz mit der Netzmaske »/20« angesehen werden – eine grosse Vereinfachung, da die Tabellen in Routern wesentlich kompakter ausfallen können. Im Internet werden heute normalerweise keine Adressen direkt geroutet, deren Netzwerkanteil länger als 19 Bit ist; Sie müssen sich in der Regel eines Providers bedienen, der den ganzen Adressenblock verwaltet und die IP-Datagramme dann intern weiterleitet.

3.4.4 Subnetting

Oftmals ist die Einteilung in ein großes Netzwerk zu ungenau oder nicht sinnvoll. Deswegen teilen Betreiber ihre Netzwerke oft in mehrere kleinere Netzwerke auf. Das geschieht, indem dem festen Netzwerkanteil der IP-Adresse noch ein weiterer fester Anteil mitgegeben wird, der das unterteilte Netz, das »Subnet«, spezifiziert.

Subnetting

Subnetting könnte im Beispiel von oben möglicherweise so aussehen: Sie möchten statt eines »großen« Netzes mit 32 Adressen (für 30 Stationen) beispielsweise zwei »kleine« Netze mit je 16 Adressen (für jeweils bis zu 14 Stationen) betreiben, etwa um aus Sicherheitsgründen getrennte Ethernet-Stränge einsetzen zu können. Hierzu verlängern Sie die Netzmaske um 1 Bit; die Netzwerk-, Stations- und Broadcast-Adressen ergeben sich sinngemäß wie oben (Tabelle 3.4).

Ungleich große Subnetze

Es ist nicht notwendig, dass alle Subnetze gleich groß sind. Das Netz 203.177.8.0/24 ließe sich zum Beispiel ohne weiteres in ein Subnetz mit 126 Stationsadressen (etwa 203.177.8.0/25 mit den Stationsadressen 203.177.8.1 bis 203.177.8.126 und der Broadcast-Adresse 203.177.8.127) und zwei Subnetze mit je 62 Stationsadressen (etwa 203.177.8.128/26 und 203.177.8.192/26 mit den respektiven Stationsadressen 203.177.8.129 bis 203.177.8.190 und 203.177.8.193 bis 203.177.8.254 sowie den Broadcast-Adressen 203.177.8.191 und 203.177.8.255) aufteilen.

Kleinstmögliches Netz

Das kleinstmögliche IP-Netz hat einen 30 Bit langen Netzwerkanteil und einen 2 Bit langen Stationsanteil. Dies sind insgesamt vier Adressen, von denen eine Netzwerk- und eine Broadcast-Adresse abgehen, es bleiben also noch zwei Adressen für Stationen. Sie finden diese Konstellation mitunter bei Punkt-zu-Punkt-Verbindungen etwa über Modem oder ISDN.

3.4.5 Private IP-Adressen

Weltweite Vergabe von IP-Adressen

IP-Adressen sind weltweit eindeutig und müssen deswegen zentral vergeben werden. Sie können Ihre IP-Adresse also nicht beliebig wählen, sondern müssen sie beantragen – typischerweise bei Ihrem Provider, der seinerseits einen Block von

Tabelle 3.5: Private IP-Adressbereiche nach RFC 1918

Adressraum	von	bis
Class A	10.0.0.0	10.255.255.255
Class B	172.16.0.0	172.31.255.255
Class C	192.168.0.0	192.168.255.255

IP-Adressen von einer nationalen oder internationalen Organisation zugeordnet bekommen hat (Abschnitt 3.1.2). Die Anzahl der international möglichen Netzwerkadressen ist natürlich limitiert.

Anfang Februar 2011 teilte die IANA den fünf regionalen Registries die letzten fünf verfügbaren /8-Adressbereiche zu. Wahrscheinlich werden dem APNIC (*Asia Pacific Network Information Centre*) zuerst die IP-Adressen ausgehen, vermutlich Mitte 2011. Danach hilft dann nur noch Betteln oder IPv6.

Für Netzwerke, die nicht direkt an das Internet angeschlossen sind, sind besondere Adressbereiche, die privaten IP-Adressen gemäß [RFC1918], vorgesehen, die im Internet nicht geroutet werden (Tabelle 3.5). Diese Adressen können Sie ungeniert in Ihren lokalen Netzen verwenden – inklusive Subnetting und allen Finessen.

Private IP-Adressen

3.4.6 Masquerading und Portweiterleitung

IP-Adressen sind heute eine knappe Ressource, und das wird auch so bleiben, bis wir alle auf IPv6 (Abschnitt 3.5) umgestiegen sind. Es ist also höchst wahrscheinlich, dass Sie nur eine einzige »offizielle« (also Nicht-RFC-1918-)IP-Adresse zur Verfügung haben, um Ihr ganzes Netz ans Internet anzubinden – bei Heimnetzen oder solchen in kleinen Firmen ist das sogar die Regel. Die Abhilfe (euphemistisch für »lahme Krücke«) dagegen ist einerseits »Masquerading«, andererseits »Portweiterleitung«. Beiden Verfahren liegt zugrunde, dass als einziges Ihr Router mit einer öffentlichen Adresse am Internet teilnimmt. Alle anderen Stationen in Ihrem Netz haben Adressen nach [RFC1918]. Masquerading bedeutet, dass Ihr Router bei Datagrammen, die Stationen in Ihrem Netz nach »draußen« schicken, die Absender-IP-Adresse der betreffenden Station durch seine eigene ersetzt und die dazugehörigen Antwortdatagramme an den eigentlichen Absender weiterleitet. Sowohl die Stationen in Ihrem Netz als auch »das Internet« bekommen davon nichts mit – die einen denken, sie reden direkt mit dem Internet, während das andere nur die (offizielle) IP-Adresse Ihres Routers zu sehen bekommt. Umgekehrt ermöglicht Portweiterleitung, dass Rechner aus dem Internet Dienste wie zum Beispiel DNS, Mail oder HTTP über die jeweiligen Ports auf dem *Router* ansprechen, dieser die entsprechenden Datagramme aber an einen Rechner im internen Netz weiterleitet, der den tatsächlichen Dienst erbringt.

Masquerading

Portweiterleitung

Sie sollten der Versuchung widerstehen, Ihren Router gleichzeitig zum Web-, Mail- oder DNS-Server zu machen; die Gefahr, dass ein Angreifer über eines der großen dafür nötigen Serverprogramme Ihren Router kompromittiert und damit im schlimmsten Fall freien Zugriff auf Ihr ganzes lokales Netz bekommt, ist viel zu groß.

Portweiterleitung und Masquerading sind zwei Ausprägungen eines Konzepts, das allgemein als NAT (engl. *network address translation*, Netzadress-Umsetzung) bezeichnet wird. Im Speziellen sprechen wir bei Masquerading auch von *Source NAT*, da die Absenderadresse von ausgehenden Datagrammen modifiziert wird[4], während Portweiterleitung einen Fall von *Destination NAT* darstellt – hier wird die Zieladresse von an uns gerichteten Datagrammen verändert.

NAT

[4] Dass wir auch die Zieladresse der als Antwort eingehenden Datagramme umschreiben müssen, ignorieren wir hier der Bequemlichkeit halber.

Übungen

3.5 [!1] Können die folgenden IP-Adressen mit der aufgeführten Netzmaske als Stationsadressen in dem betreffenden IP-Netz verwendet werden? Wenn nein, warum nicht?

	IP-Adresse	Netzmaske
a)	`172.55.10.3`	`255.255.255.252`
b)	`138.44.33.12`	`255.255.240.0`
c)	`10.84.13.160`	`255.255.255.224`

3.6 [2] Welche Gründe könnten Sie dafür haben, einen Adressbereich, den Sie von einem Provider bekommen, in Subnetze aufzuteilen?

3.7 [!2] Das Netz mit der IP-Adresse `145.2.0.0` und der Netzmaske `255.255.0.0` wurde mit der Subnetzmaske `255.255.240.0` in folgende Subnetze aufgeteilt:

- `145.2.128.0`
- `145.2.64.0`
- `145.2.192.0`
- `145.2.32.0`
- `145.2.160.0`

Welche weiteren Subnetze sind noch möglich? In welchem Subnetz befindet sich die Station `145.2.195.13`?

3.5 IPv6

3.5.1 Überblick

IPv4

Die heute am weitesten verbreitete Fassung von IP ist die Version 4, kurz »IPv4« genannt. Durch das rasche Wachstum des Internets stößt diese Version inzwischen an ihre Grenzen – Hauptprobleme sind die zunehmende Adressenverknappung, der Wildwuchs bei der Adressvergabe und das daraus resultierende hochkomplexe Routing sowie die kaum vorhandene Unterstützung von Sicher-

IPv6

heitsmechanismen und Methoden zur Dienstgütesteuerung. IPv6 soll hier Abhilfe schaffen.

Eigenschaften

Die wichtigsten Eigenschaften von IPv6 sind:

- Die Adresslänge wird von 32 auf 128 Bit vergrößert, wodurch man auf $3{,}4 \cdot 10^{38}$ Adressen kommt. Dies würde ausreichen, um jeder der heute lebenden 6,5 Milliarden Personen rund 50 Quadrillionen (eine Zahl mit 27 Nullen) IPv6-Adressen zuzuordnen. Das sollte für die vorhersehbare Zukunft genügen.
- IPv6-Rechner können sich automatisch Konfigurationsparameter von einem Router holen, wenn sie an ein Netz angeschlossen werden. Falls nötig, gibt es immer noch ein DHCPv6-Protokoll.
- Ein IP-Kopf enthält nur noch 7 Felder, damit Router die Pakete schneller verarbeiten können. Bei Bedarf können mehrere Köpfe für ein Datagramm verwendet werden.
- Erweiterte Unterstützung von Optionen und Erweiterungen, was auch dazu beiträgt, dass Router Datagramme schneller verarbeiten können.
- Bessere Übertragung von Audio- und Videodaten sowie bessere Unterstützung von Echtzeitanwendungen.

- Erhöhte Sicherheit durch abgesicherte Datenübertragung sowie Methoden zur Authentisierung und Integritätssicherung.
- Erweiterbarkeit, um die Zukunftsfähigkeit des Protokolls sicherzustellen. Es wurde nicht versucht, alle Möglichkeiten abzudecken, denn die Zukunft bringt Neuerungen, die heute nicht absehbar sind. Stattdessen ist das Protokoll offen für die rückwärtskompatible Integration weiterer Funktionen.

Während die Standardisierung von IPv6 schon seit einer ganzen Weile abgeschlossen ist, hapert es an der allgemeinen Umsetzung durchaus noch. Es sind vor allem die Dienstanbieter, die sich zieren. Linux unterstützt IPv6 schon jetzt, so dass die Umstellung einer Linux-Infrastruktur auf den neuen Standard kein Problem darstellen wird. Sie können auch IPv6-Pakete testhalber über IPv4 transportieren, indem Sie sie in IPv4-Pakete einpacken (»Tunneling«). Damit könnte eine Firma zum Beispiel ihre internen Netze auf IPv6 aufbauen lassen und sogar mehrere Standorte über ein »virtuelles« IPv6-Netz im traditionellen IPv4-Netz verbinden. Umsetzung

Wir sollten außerdem hervorheben, dass IPv6 nur einen gezielten Ersatz für das bisherige IPv4 darstellt. Die allermeisten Protokolle, die auf IP aufsetzen – beginnend mit TCP und UDP – bleiben unverändert. Lediglich auf der »Infrastrukturebene« werden einige Protokolle überflüssig oder durch IPv6-basierte Versionen ersetzt.

3.5.2 IPv6-Adressierung

IPv6 erlaubt 2^{128} verschiedene Adressen – eine unvorstellbar große Zahl. Im Grunde könnte jedes Sandkorn auf der Erde über etliche Adressen verfügen, aber das ist gar nicht Ziel der Sache: Der große Adressraum gestattet eine deutlich flexiblere Adressvergabe für die unterschiedlichsten Zwecke sowie viel einfachereres Routing.

IPv6-Adressen werden nicht wie IPv4-Adressen dezimal notiert, sondern hexadezimal (also zur Basis 16). Dabei werden immer vier Ziffern zusammengefasst und diese Gruppen durch Doppelpunkte getrennt. Eine IPv6-Adresse könnte also aussehen wie Schreibweise

```
fe80:0000:0000:0000:025a:b6ff:fe9c:406a
```

Führende Nullen in einer Gruppe dürfen wegfallen, und (maximal) eine Folge von Null-Blocks darf durch zwei Doppelpunkte ersetzt werden. Eine verkürzte Schreibweise für die Adresse aus dem vorigen Beispiel wäre also

```
fe80::25a:b6ff:fe9c:406a
```

Der IPv4-Loopback-Adresse `127.0.0.1` entspricht die IPv6-Adresse `::1` – eine Abkürzung von

```
0000:0000:0000:0000:0000:0000:0000:0001
```

»Broadcast-Adressen« à la `192.168.1.255` gibt es bei IPv6 nicht (hierzu später mehr).

IPv6-Adressen können in einen 64 Bit breiten Netz- und einen 64 Bit breiten Stationsanteil aufgeteilt werden. Das heißt, jedes IPv6-Subnetz verfügt über 2^{64} Adressen, also 2^{32}-mal so viele wie das komplette IPv4-Internet! Subnetting mit variablen Präfixlängen, so wie es in IPv4 gemacht wird (Abschnitt 3.4.4) ist bei IPv6 nicht wirklich vorgesehen. Es wird aber davon ausgegangen, dass Ihr Provider Ihnen zum Beispiel ein »`/56`«-Adresspräfix zur Verfügung stellt, so dass Sie über 256 Subnetze mit je 2^{64} Adressen verfügen können, was Ihren Stil nicht allzusehr einschränken sollte. (Netzwerkpräfixe können Sie angeben, indem Sie an die Adresse einen Schrägstrich und die Präfixlänge als Dezimalzahl anhängen – eine Adresse wie `fe80::/16` beschreibt also das Netz, wo Adressen mit `fe80` anfangen und dann beliebig weitergehen.) Subnetting

Grundsätzlich gibt es drei Arten von IPv6-Adressen: Arten von IPv6-Adressen

- »Unicast«-Adressen beziehen sich auf eine bestimmte Netzwerkschnittstelle (ein Rechner kann mehrere Netzwerkschnittstellen haben, die jeweils mit eigenen Adressen ausgestattet sind).
- »Anycast«-Adressen beziehen sich auf eine Gruppe von Netzwerkschnittstellen. Typischerweise gehören diese zu verschiedenen Stationen und die jeweils »nächste« antwortet tatsächlich. Sie können zum Beispiel alle Router in einem IPv6-Netz adressieren, indem Sie die Adresse verwenden, die sich ergibt, wenn Sie an das (64-Bit-)Adresspräfix des Netzes einen Stationsteil anhängen, der nur aus Nullen besteht.
- »Multicast«-Adressen werden benutzt, um dieselben Pakete an mehrere Schnittstellen zuzustellen. Wie erwähnt, verwendet IPv6 keinen Broadcast; Broadcast ist ein Sonderfall von Multicast. Die Adresse `ff02::1` spricht zum Beispiel alle Rechner im lokalen Netz an.

Sichtbarkeitsbereiche

Ferner kann man verschiedene Sichtbarkeitsbereiche (*scopes*) unterscheiden:

- Globale (*global*) Sichtbarkeit gilt für Adressen, die im gesamten (IPv6-)Internet geroutet werden.
- Verbindungslokale (*link-local*) Sichtbarkeit gilt für Adressen, die nicht geroutet werden und nur im selben Netz verwendbar sind. Solche Adressen werden meist für interne IPv6-Verwaltungszwecke benutzt. Verbindungslokale Adressen sind immer im Netz `fe80::/64`; die übrigen 64 Bit ergeben sich im einfachsten Fall aus der MAC-Adresse der Schnittstelle.
- Standortlokale (*site-local*) Sichtbarkeit gilt für Adressen, die nur innerhalb eines »Standorts« geroutet werden. Niemand weiß genau, was das heißen soll, und standortlokale Adressen sind inzwischen (wieder) verpönt. Standortlokale Adressen haben das Präfix `fec0::/10`.
- Eindeutig lokale (*unique-local*) Adressen ähneln standortlokalen Adressen und erfüllen mehr oder weniger den Zweck, den die RFC-1918-Adressen (`192.168.`*x.y* & Co.) bei IPv4 hatten – wobei IPv6 es einfach macht, »richtige«, also global sichtbare Adressen zu vergeben, so dass Sie nicht auf eindeutig lokale Adressen zurückgreifen müssen, nur damit Ihre Stationen überhaupt Adressen bekommen können. Es gibt also keinen wichtigen Grund, eindeutig lokale Adressen zu benutzen, außer vielleicht als Rückzugsposition, falls irgendetwas mit Ihrem »echten« Präfix nicht stimmt. Eindeutig lokale Adressen haben das Präfix `fd00::/8`; die nächsten 40 Bit für ein `/48`-Netz dürfen Sie sich selbst aussuchen (aber nehmen Sie nicht `fd00::/48`).

mehrere Adressen

Eine sehr wichtige Beobachtung ist, dass in IPv6 jede Netzwerkschnittstelle mehrere Adressen haben kann. Sie bekommt automatisch eine verbindungslokale Adresse, aber kann ohne Weiteres auch noch mehrere eindeutig lokale oder global sichtbare Adressen haben. Alle diese Adressen sind gleichwertig.

ipv6calc

Ein nützliches Kommando für den geplagten IPv6-Administrator ist `ipv6calc`, das den Umgang mit IPv6-Adressen erleichtert. Zum Beispiel kann es Informationen über eine Adresse ausgeben:

```
$ ipv6calc --showinfo fe80::224:feff:fee4:1aa1
No input type specified, try autodetection... found type: ipv6addr
No output type specified, try autodetection... found type: ipv6addr
Address type: unicast, link-local
Error getting registry string for IPv6 address:▷
 ◁ reserved(RFC4291#2.5.6)
Interface identifier: 0224:feff:fee4:1aa1
EUI-48/MAC address: 00:24:fe:e4:1a:a1
MAC is a global unique one
MAC is an unicast one
OUI is: AVM GmbH
```

Bei der Adresse in unserem Beispiel handelt es sich also um eine verbindungslokale Unicast-Adresse, deren Stationsteil auf ein Gerät hindeutet, das von der AVM GmbH hergestellt wurde (tatsächlich eine FRITZ!Box).

`ipv6calc` erlaubt auch die Umrechnung von Adressen von einem Format in ein anderes. Sie können damit zum Beispiel die Methode nachstellen, mit der aus einer MAC-Adresse der Stationsteil einer IPv6-Adresse (auch „EUI-64“ genannt) erzeugt wird:

```
$ ipv6calc --in mac --out eui64 00:24:fe:e4:1a:a1
No action type specified, try autodetection... found type: geneui64
0224:feff:fee4:1aa1
```

Kommandos in diesem Kapitel

arp Erlaubt Zugriff auf den ARP-Cache (Abbildung von IP- auf MAC-Adressen) `arp`(8) 53

inetd Internet-Superserver, überwacht Ports und startet ggf. Dienste `inetd`(8) 57

ipv6calc Hilfsprogramm für Berechnungen mit IPv6-Adressen `ipv6calc`(8) 66

xinetd Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste `xinetd`(8) 57

Zusammenfassung

- Das Internet wurzelt in den ARPAnet-Anfängen der 1960er Jahre, wurde in den frühen 1980er Jahren auf seine heutige technische Basis gestellt und nahm in den 1980er und 1990er Jahren einen ungeahnten Aufschwung.
- Das ISO/OSI-Referenzmodell dient zur Begriffsbildung über die Struktur von Rechnerkommunikation.
- TCP/IP ist die heute populärste Protokollfamilie für den Datentransfer in Rechnernetzen.
- ICMP dient zur Netzverwaltung und Problemweitermeldung.
- TCP bietet auf der Basis von IP einen verbindungsorientierten und zuverlässigen Transportdienst.
- UDP ist wie IP verbindungslos und unzuverlässig, aber viel einfacher und schneller als TCP.
- TCP und UDP verwenden Portnummern, um zwischen verschiedenen Verbindungen auf demselben Rechner zu unterscheiden.
- Verschiedene TCP/IP-Dienste haben feste Portnummern zugeordnet. Diese Zuordnung ist der Datei `/etc/services` zu entnehmen.
- IP-Adressen identifizieren Stationen weltweit. Sie sind 32 Bit lang und bestehen aus einem Netzwerk- und einem Stationsteil, zwischen denen über die Netzmaske unterschieden wird.
- Früher wurden die verfügbaren IP-Adressen in Klassen eingeteilt. Heute verwendet man klassenlose Wegleitung mit Netzmasken variabler Länge.
- IP-Netze können weiter in Subnetze unterteilt werden, indem man die Netzmaske anpasst.
- Für die Verwendung in lokalen Netzen sind einige IP-Adressbereiche reserviert, die von Providern nicht geroutet werden.
- IPv6 hebt diverse Einschränkungen des heute üblichen IPv4 auf, befindet sich aber noch nicht in weitem Gebrauch.

Literaturverzeichnis

IPv6-HOWTO Peter Bieringer. »Linux IPv6 HOWTO«, Oktober 2005. http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/

RFC0768 J. Postel. »User Datagram Protocol«, August 1980. http://www.ietf.org/rfc/rfc0768.txt

RFC0791 Information Sciences Institute. »Internet Protocol«, September 1981. http://www.ietf.org/rfc/rfc0791.txt

RFC0792 J. Postel. »Internet Control Message Protocol«, September 1981. http://www.ietf.org/rfc/rfc0792.txt

RFC0793 Information Sciences Institute. »Transmission Control Protocol«, September 1981. http://www.ietf.org/rfc/rfc0793.txt

RFC0826 David C. Plummer. »An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Addresses for Transmission on Ethernet Hardware«, November 1982. http://www.ietf.org/rfc/rfc0826.txt

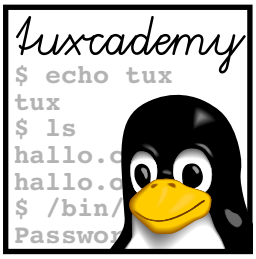
RFC1519 V. Fuller, T. Li, J. Yu, et al. »Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy«, September 1993. http://www.ietf.org/rfc/rfc1519.txt

RFC1918 Y. Rekhter, B. Moskowitz, D. Karrenberg, et al. »Address Allocation for Private Internets«, Februar 1996. http://www.ietf.org/rfc/rfc1918.txt

RFC4291 R. Hinden, S. Deering. »IP Version 6 Addressing Architecture«, Februar 2006. http://www.ietf.org/rfc/rfc4291.txt

Ste94 W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional Computing Series. Boston etc.: Addison-Wesley, 1994.

Tan02 Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, 2002, 3. Auflage. Unbedingt auf englisch lesen, die deutsche Übersetzung ist eine Katastrophe.



4

Linux-Netzkonfiguration

Inhalt

4.1 Netzschnittstellen 70
4.1.1 Hardware und Treiber 70
4.1.2 Netzwerkkarten konfigurieren mit `ifconfig` 71
4.1.3 Wegleitung konfigurieren mit `route` 72
4.1.4 Netzkonfiguration mit `ip` 75
4.2 Dauerhafte Netzkonfiguration 76
4.3 DHCP 78
4.4 IPv6-Konfiguration 80
4.5 Namensauflösung und DNS 81

Lernziele

- Die Netzkonfigurationsmechanismen der wichtigsten Distributionen kennen
- Netzschnittstellen konfigurieren können
- Statische Routen einrichten können
- Linux als DHCP- und DNS-Client konfigurieren können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 3)

4.1 Netzschnittstellen

4.1.1 Hardware und Treiber

Je nach verwendeter Technik und Zugangsverfahren sprechen Linux-Rechner das Netz über Modems, ISDN-Karten, Ethernet- oder WLAN-Adapter und ähnliches an. Die folgenden Abschnitte beschäftigen esich hauptsächlich mit der Einrichtung von Ethernetkarten.

Eine Netzwerkkarte wird unter Linux wie andere Hardware auch vom Kernel angesteuert – heute normalerweise über modulare Treiber, die bei Bedarf dynamisch geladen werden. Anders als zum Beispiel Festplattenpartitionen oder Drucker erscheinen Netzwerkkarten aber nicht als Gerätedateien in `/dev`, sondern werden über Schnittstellen (engl. *interfaces*) angesprochen. Diese Schnittstellen sind »virtuell« in dem Sinne, dass der Kernel sie nach dem Einbinden des passenden Treibers direkt bereitstellt und eine Netzwerkkarte durchaus über mehr als eine (fast) unabhängige Schnittstelle angesprochen werden kann. Die Schnittstellen haben Namen; ein typischer Name für eine Ethernet-Karte wäre zum Beispiel `eth0`.

Schnittstellen

Netzwerkkarten werden heutzutage beim Start des Systems vom Kernel erkannt, der anhand der PCI-ID das richtige Treibermodul identifizieren kann. Es obliegt der `udev`-Infrastruktur, der Netzwerkkarte einen Namen zu geben und den Treiber tatsächlich zu laden.

Ein Fallstrick, den moderne Linux-Distributionen hier aufbauen, ist, dass der Schnittstellenname einer Netzwerkkarte an deren MAC-Adresse gekoppelt wird. (Jede Netzwerkkarte hat eine weltweit eindeutige MAC-Adresse, die vom Hersteller gesetzt wird.) Wenn Sie in einem Rechner also die Netzwerkkarte austauschen, ohne die Informationen zurückzusetzen, die `udev` sich über die Netzwerkkarten merkt, die es schon mal gesehen hat, dann stehen die Chancen gut, dass Ihre neue Karte den Namen `eth1` bekommt und die Konfiguration, die von der Existenz von `eth0` ausgeht, nicht beachtet wird.

Ein typischer Platz, wo solche Informationen landen können, ist das Verzeichnis `/etc/udev/rules.d`. In einer Datei wie `70-persistent-net.rules` können sich Zeilen finden wie

```
SUBSYSTEM=="net", DRIVERS=="?*",▷
 ◁ ATTRS{address}=="00:13:77:01:e5:4a", NAME="eth0"
```

die der Karte mit der MAC-Adresse `00:13:77:01:e5:4a` den Namen `eth0` zuordnet. Sie können die MAC-Adresse mit der Hand korrigieren oder die Zeile komplett löschen und darauf warten, dass `udev` beim nächsten Systemstart die Einträge an die veränderte Realität anpasst.

Machen Sie sich keinen Kopf, wenn Sie Linux in einer virtuellen Maschine laufen lassen und die Datei `70-persistent-net.rules` nicht finden können. Für die meisten »virtuellen« Netzwerkschnittstellen wird sie nämlich gar nicht angelegt.

Früher (in der Zeit vor `udev`) oblag es den Installationsprozeduren der Distributionen, die richtigen Treiber für Netzwerkkarten zu finden und dem System bekannt zu machen. Typischerweise ging das über `/etc/modules.conf`, wo Einträge wie

```
alias eth0 3c59x
```

untergebracht wurden – dies war ein Signal an den Kernel, beim ersten Zugriff auf die Schnittstelle `eth0` das Treibermodul `3c59x.o` zu laden. Aber das ist vorbei …


Der Linux-Kernel ist natürlich nicht notwendigerweise modular ausgelegt, auch wenn die Standardkernels der meisten Distributionen ohne Module nicht auskommen könnten. Wenn Sie Ihren eigenen Kernel übersetzen (siehe hierzu etwa *Linux-Systemanpassungen*), dann können Sie die Treiber für Ihre Netzwerkkarten auch fest einbauen.


Für besondere Anforderungen, typischerweise bei Rechnern mit erhöhtem Sicherheitsbedürfnis wie paketfilternden Routern oder Servern, die dem Internet ausgesetzt sind, können Sie die Infrastruktur für ladbare Module auch komplett aus dem Kernel verbannen. Das macht es Crackern schwerer (aber leider nicht unmöglich), sich unbemerkt auf dem System einzunisten.

4.1.2 Netzwerkkarten konfigurieren mit `ifconfig`

Bevor Sie eine Schnittstelle zum Zugriff auf das Netz verwenden können, müssen Sie ihr eine IP-Adresse, eine Netzmaske und so weiter zuweisen. Manuell geht das traditionellerweise mit dem Kommando `ifconfig`:

```
# ifconfig eth0 192.168.0.75 up
# ifconfig eth0
eth0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:73
     inet addr:192.168.0.75 Bcast:192.168.0.255 Mask:255.255.255.0
     inet6 addr: fe80::2a0:24ff:fe56:e373/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:6 errors:0 dropped:0 overruns:0 carrier:6
     collisions:0 txqueuelen:100
     RX bytes:0 (0.0 b)  TX bytes:460 (460.0 b)
     Interrupt:5 Base address:0xd800
```

Nach der Zuweisung einer IP-Adresse können Sie durch Aufruf des gleichen Kommandos ohne Angabe einer IP-Adresse den Status der Schnittstelle auslesen. Hier werden nicht nur die aktuelle IP-Adresse, sondern auch der Hardwaretyp, die MAC- (oder »Hardware-«)Adresse, die Broadcast-Adresse, die Netzmaske, die IPv6-Adresse und viele weitere Daten angezeigt. Am Beispiel lässt sich erkennen, dass Werte wie Netzmaske und Broadcast-Adresse auch ohne explizite Zuweisung vom Kernel auf Standardwerte (hier gemäß dem ersten Oktett der vergebenen IP-Adresse die für ein Class-C-Netz) gesetzt werden. Sollten die gewünschten Werte vom Standard abweichen, müssen Sie sie explizit angeben:

```
# ifconfig eth0 192.168.0.75 netmask 255.255.255.192 \
>      broadcast 192.168.0.64
# ifconfig eth0
eth0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:73
     inet addr:192.168.0.75 Bcast:192.168.0.64 Mask:255.255.255.192
     inet6 addr: fe80::2a0:24ff:fe56:e373/64 Scope:Link
◁◁◁◁◁
```


Mit den Parametern `up` und `down` können Sie mit `ifconfig` einzelne Interfaces gezielt hoch- bzw. herunterfahren.


Das Loopback-Interface hat nach Konvention die IP-Adresse `127.0.0.1` und wird automatisch konfiguriert. Sollte das aus irgendwelchen Gründen einmal nicht klappen oder die Konfiguration verlorengehen, dann können Sie das über

Loopback-Interface

```
# ifconfig lo 127.0.0.1 up
```

nachholen.

Aliasnamen

Zu Testzwecken oder für besondere Anforderungen kann es sinnvoll sein, einer Schnittstelle einen Aliasnamen mit einer abweichenden IP-Adresse, Netzmaske usw. zu geben. Das ist mit `ifconfig` kein Problem:

```
# ifconfig eth0:0 192.168.0.111
# ifconfig eth0:0
eth0:0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:72
     inet addr:192.168.0.111 Bcast:192.168.0.255 Mask:255.255.255.0
     UP BROADCAST MULTICAST  MTU:1500  Metric:1
     Interrupt:5 Base address:0xd800
```

Der Aliasname wird aus dem Interfacenamen gebildet, indem getrennt durch einen Doppelpunkt eine Namenserweiterung angehängt wird. Wie diese Namenserweiterung aussieht, ist gleichgültig (es spräche nichts gegen `eth0:Mr.X`), nach Konvention numeriert man die Aliasnamen aber fortlaufend durch: `eth0:0`, `eth0:1`, …

Übungen

4.1 [1] Welches Kernel-Modul gehört zu Ihrer Netzwerkkarte? Ist es geladen?

4.2 [!1] Kontrollieren Sie, ob Ihre Netzwerkkarte läuft und welche IP-Adresse ihr zugeordnet ist.

4.3 [!2] Weisen Sie Ihrer Netzwerkkarte eine neue IP-Adresse (ggf. nach Angaben Ihres Trainers) zu. Kontrollieren Sie, ob Sie die anderen Rechner im Netz noch erreichen können.

4.1.3 Wegleitung konfigurieren mit `route`

Jeder Rechner in einem TCP/IP-Netzwerk benötigt Wegleitung, denn selbst die einfachste Station besitzt mindestens zwei Netzschnittstellen – das Loopback-Interface und die Schnittstelle zum restlichen Netz, also eine Ethernet- oder WLAN-Karte oder eine Verbindung zum Internet. Die Routen für das Loopback-Interface und für die Netze, in denen Netzwerkkarten sich direkt befinden, werden bei aktuellen Linux-Kernels bei der Initialisierung der Netzwerkkarten automatisch gesetzt. Andere Routen – insbesondere die »Default-Route«, die angibt, wo Datagramme hingeschickt werden, für die es keine bessere Antwort gibt – müssen explizit konfiguriert werden.

Grundsätzlich unterscheiden wir zwischen *statischer* und *dynamischer* Wegleitung. Bei ersterer werden die Routen manuell gesetzt und danach nicht oder nur selten geändert. Bei letzterer unterhält das System sich mit anderen Routern in seiner Umgebung und passt seine Routen den aktuellen Gegebenheiten im Netz an. Dynamische Wegleitung erfordert die Installation und Konfiguration von »Routing-Daemons« wie `gated` oder `routed` und wird hier nicht weiter behandelt. Wir beschäftigen uns im Rest dieses Abschnitts ausschließlich mit statischer Wegleitung.

Routing-Tabelle

Der Kernel unterhält eine Routing-Tabelle, die die aktuelle Konfiguration der Wegleitung zusammenfasst. Sie enthält Regeln (die Routen), die beschreiben, wohin welche Datagramme geschickt werden müssen. Maßgeblich dafür ist deren Zieladresse. Sie können die Routing-Tabelle mit dem Kommando `route` abrufen:

```
# ifconfig eth0 192.168.0.75
# route
Kernel IP routing table
Destination  Gateway Genmask       Flags Metric Ref Use Iface
192.168.0.0  *       255.255.255.0 U     0      0   0   eth0
```

Die Spalten der Tabelle haben folgende Bedeutung:

- Die erste Spalte enthält die Zieladresse. Als Adresse kommen Netz- und Stationsadressen sowie der Eintrag (`default`) für die sogenannte Default-Route in Frage. Die Default-Route legt das Ziel für alle Datagramme fest, für die keine der übrigen Routen gilt. Default-Route
- Die zweite Spalte definiert als Ziel für die Datagramme einen Router, an den die Pakete weitergegeben werden. Gültige Einträge an dieser Stelle sind Stationsadressen sowie der Eintrag »*« wenn die Pakete nicht an einen anderen Rechner gehen sollen.
- Die dritte Spalte enthält die zur Zieladresse passende Netzmaske. Handelt es sich bei der Zieladresse um eine Station, dann steht hier die Netzmaske `255.255.255.255`. Die Default-Route hat die Netzmaske `0.0.0.0`.
- Die vierte Spalte enthält Flags, die die Route näher beschreiben. Folgende Werte können u. a. vorkommen:

  `U` die Route ist aktiv (up)

  `G` die Route ist eine »Gateway-Route«, das heisst, als Ziel ist ein Router (und kein direkt angeschlossenes Netz wie mit "*") angegeben.

  `H` die Route ist eine »Host-Route«, das heisst, die Zieladdresse bezeichnet einen einzelnen Rechner. `G` und `H` schliessen sich natürlich nicht aus und tauchen manchmal zusammen auf.
- Die fünfte und sechste Spalte enthalten Angaben, die bei dynamischem Routing eine Rolle spielen: Die »Metrik« in der fünften Spalte gibt die Anzahl der »Hops« zum Ziel an; sie wird vom Linux-Kern nicht ausgewertet, sondern ist vor allem für Programme wie den `gated` interessant. Der Wert in der sechsten Spalte wird in Linux nicht verwendet.
- Die siebte Spalte gibt an, wie oft die Route schon verwendet wurde.
- Die achte Spalte schließlich enthält optional die Schnittstelle, über die die Datagramme weitergeleitet werden sollen. Das kommt insbesondere bei Routern vor, die mehrere Interfaces besitzen, etwa Ethernet-Schnittstellen in verschiedenen Netzsegmenten oder eine Ethernet-Schnittstelle und eine Schnittstelle zum ISDN.

Am Beispiel wird deutlich, dass der Kernel beim Setzen der IP-Adresse mit `ifconfig` nicht nur eigenständig Netzmaske und Broadcast-Adresse setzt, sondern auch mindestens eine Route – diejenige nämlich, die alle Datagramme, deren Zieladressen im direkt an die Schnittstelle angeschlossenen Netz liegen, auf dieses Netz leitet.

Ein komplexeres Beispiel für eine Routing-Tabelle könnte so aussehen:

```
# route
Kernel IP Routentabelle
Ziel        Router      Genmask         Flags Metric Ref Use Iface
192.168.0.0 *           255.255.255.0   U     0      0   0   eth0
192.168.2.0 *           255.255.255.0   U     0      0   0   eth1
10.10.3.0   192.168.0.1 255.255.255.0   UG    0      0   0   eth0
112.22.3.4  *           255.255.255.255 UH    0      0   0   ppp0
default     112.22.3.4  0.0.0.0         UG    0      0   0   ppp0
```

Beim Beispielrechner handelt es sich offenbar um einen Router mit drei Schnittstellen. Die ersten drei Routen sind Netzwerkrouten, die Datagramme laufen entsprechend ihres Zielnetzes entweder über `eth0`, `eth1` oder den Router `192.168.0.1` (der über die erste Route zu erreichen ist). Die vierte Route ist eine »Host-Route«, die eine Punkt-zu-Punkt-Verbindung zum Providerrechner über das Modem `ppp0`

aufbaut. Die fünfte Route ist die entsprechende Default-Route, die alle Datagramme, die nicht in die lokalen Netze `192.168.0.0/24`, `192.168.2.0/24` oder `10.10.3.0/24` gerichtet sind, über das Modem in die weite Welt weiterleitet.

Das Kommando `route` dient nicht nur zum Abrufen, sondern auch zur Manipulation der Routing-Tabelle. Für das oben gezeigte Beispiel (drei lokale Ethernet-Segmente und die PPP-Verbindung) würde die Routing-Tabelle etwa folgendermassen aufgebaut:

```
# route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 dev eth1
# route add -net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
# route add -host 112.22.3.4 dev ppp0
# route add default dev ppp0
```

Die ersten beiden Zeilen im Beispiel sind eigentlich nicht nötig, da die entsprechenden Routen automatisch in Kraft gesetzt werden, wenn die Schnittstellen ihre Adressen bekommen.

Allgemeiner gesagt hat `route` zum Setzen und Entfernen von Routen folgende Syntax:

```
route add [-net|-host] ⟨Ziel⟩ [netmask ⟨Netzmaske⟩]▷
 ◁      [gw ⟨Gateway⟩] [[dev] ⟨Interface⟩]
route del [-net|-host] ⟨Ziel⟩ [netmask ⟨Netzmaske⟩]▷
 ◁      [gw ⟨Gateway⟩] [[dev] ⟨Interface⟩]
```

Zum Hinzufügen einer Route müssen Sie den entsprechenden Parameter (`add`) setzen; danach geben Sie an, ob es sich um eine Stations- oder Netzroute handelt (`-host` oder `-net`). Danach wird das Ziel definiert. Handelt es sich um eine Netzroute, muss immer eine Netzmaske (`netmask` ⟨*Netzmaske*⟩) angegeben werden; Sie können die Netzmaske auch im CIDR-Stil an die Zieladresse anhängen. Für jede Route muss entweder ein Router (⟨*Gateway*⟩) oder eine Schnittstelle als »nächste Station« angegeben werden.

Routen löschen

Löschen könnten Sie die Routen zum Beispiel so:

```
# route del -net 192.168.0.0 netmask 255.255.255.0
# route del -net 192.168.2.0 netmask 255.255.255.0
# route del -net 10.0.3.0 netmask 255.255.255.0
# route del -host 112.22.3.4
# route del default
```

Zum Löschen einer Route müssen Sie die gleichen Angaben machen wie zum Hinzufügen einer Route. Lediglich die Angabe des Gateways bzw. des Interfaces können Sie weglassen. Bei doppelt vorkommenden Zielen, etwa dasselbe Zielnetz über zwei verschiedene Schnittstellen, wird zuerst die jüngste (zuletzt gesetzte) Route gelöscht.

IP-Forwarding

Soll ein Rechner wie im Beispiel als Gateway zwischen zwei Netzen dienen, so sollte der Kernel IP-Pakete, die nicht für den Rechner selbst bestimmt sind, entsprechend der Routing-Tabelle weitergeben. Dieses sogenannte **IP-Forwarding** ist standardmäßig ausgeschaltet. Die Einstellung und Anzeige des IP-Forwarding findet über die (Pseudo!-)Datei `/proc/sys/net/ipv4/ip_forward` statt. In ihr »steht« nur ein Zeichen – entweder eine Null (abgeschaltet) oder eine Eins (aktiviert). »Geschrieben« wird die Datei üblicherweise mit `echo`:

```
# cat /proc/sys/net/ipv4/ip_forward
0
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
# cat /proc/sys/net/ipv4/ip_forward
1
```

Achtung: Diese Einstellung geht, wie die bisher besprochenen Einstellungen per Kommando, verloren, wenn der Rechner neu gestartet wird. (Die Distributionen haben Mittel und Wege, diese Einstellung permanent abzulegen; bei Debian GNU/Linux etwa über einen Eintrag der Form »ip_forward=yes« in der Datei /etc/network/options, bei den Novell/SUSE-Distributionen durch »IP_FORWARD="yes"« in /etc/sysconfig/sysctl. Bei den RedHat-Distributionen können Sie eine Zeile der Form

```
net.ipv4.ip_forward = 1
```

in die Datei /etc/sysctl.conf eintragen.)

4.1.4 Netzkonfiguration mit ip

Mit dem Kommando ip können sowohl die Schnittstelle als auch Routen konfiguriert werden. Das Kommando soll die eben beschriebenen Kommandos über kurz oder lang ablösen. Die Syntax lautet allgemein wie folgt:

```
ip [⟨Option⟩] ⟨Objekt⟩ [⟨Kommando⟩ [⟨Parameter⟩]]
```

Als ⟨*Objekt*⟩ kommen unter anderem link (Parameter der Schnittstelle), addr (IP-Adresse und andere Adressen der Schnittstelle) und route (Auslesen, Setzen und Löschen von Routen) in Frage. Für jedes Objekt stehen spezifische Kommandos zur Verfügung.

Wird kein Kommando angegeben, werden die momentanen Einstellungen entsprechend des Kommandos list bzw. show angezeigt. Weitere typische Kommandos sind set für das Objekt link sowie add und del für die Objekte addr und route.

Die meisten Kommandos erfordern noch weitere Parameter, schließlich müssen Sie, möchten Sie mit »ip addr add« eine IP-Adresse zuweisen, diese IP-Adresse auch angeben.

Die entsprechende Syntax können Sie dem Befehl ip mit dem Kommando help entlocken. So zeigt »ip help« alle möglichen Objekte an und »ip link help« alle zum Objekt link gehörenden Parameter inklusive Syntax. Leider ist die Syntax nicht immer ganz einfach zu durchschauen.

Wenn Sie sich mit Cisco-Routern auskennen, werden Sie gewisse Ähnlichkeiten zum Cisco-ip-Kommando bemerkt haben. Diese Ähnlichkeiten sind beabsichtigt.

Als Beispiel: Möchten Sie einer Netzwerkkarte eine IP-Adresse zuweisen, können Sie dafür folgendes Kommando verwenden:

```
# ip addr add local 192.168.2.1/24 dev eth0 brd +
```

Anders als bei ifconfig *müssen* hier Netzmaske und Broadcastadresse mit angegeben werden, letztere auch indirekt mit brd +. Der Parameter local wird als Parameter zur Angabe einer IP-Adresse verwendet, da es sich für »ip addr add« hierbei aber um den Default-Parameter handelt, kann er auch weggelassen werden. Die Default-Parameter können Sie der Handbuchseite ip(8) entnehmen.

Achtung: Im Unterschied zu ifconfig ist die Schnittstelle nach der Zuweisung einer IP-Adresse noch nicht aktiv. Die Aktivierung erfolgt gesondert:

```
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo-fast qlen 100
        link/ether 00:a0:24:56:e3:72 brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0
```

```
# ip link set up dev eth0
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo-fast qlen 100
        link/ether 00:a0:24:56:e3:72 brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0
        inet6 fe80::2a0:24ff:fe56:e372/64 scope link
```

Auch mit `ip` können Sie Aliasnamen für Schnittstellen vergeben:

```
# ip addr add 192.168.0.222/24 dev eth0 brd + label eth0:0
```

Es empfiehlt sich, die Syntax von `ip` kennenzulernen, nicht nur, weil dieses Programm der zukünftige Standard ist, sondern weil es zum Teil auch anschaulicher zu verwenden ist als die Alternativen: Das Setzen und Löschen von Routen ist beispielsweise einfacher als mit `route`:

```
# ip route add 192.168.2.1 via 192.168.0.254
# ip route del 192.168.2.1
```

4.2 Dauerhafte Netzkonfiguration

Eins ist sicher: Wenn Sie einmal die richtige Netzkonfiguration für Ihr System herausbekommen haben, werden Sie diese nicht immer wieder von neuem einstellen wollen. Leider vergisst der Linux-Kernel sie aber beim Herunterfahren.

Die verschiedenen Linux-Distributionen haben dieses Problem auf unterschiedliche Weise gelöst:

Bei Debian GNU/Linux und den davon abgeleiteten Distributionen steht die Netzkonfiguration in der Datei `/etc/network/interfaces`. Diese Datei ist mehr oder weniger selbsterklärend:

```
# cat /etc/network/interfaces
auto lo eth0

iface lo inet loopback

iface eth0 inet static                                    oder »... inet dhcp«
  address 192.168.0.2
  netmask 255.255.255.0
  network 192.168.0.0
  broadcast 192.168.0.255
  up route add -net 10.10.3.0/24 gw 192.168.0.1
  down route del -net 10.10.3.0/24 gw 192.168.0.1
```

In der Datei gibt es einen Eintrag für jede Schnittstelle. Die Schnittstellen können mit den Kommandos `ifup` und `ifdown` einzeln oder (mit der Option `-a`) kollektiv aktiviert bzw. deaktiviert werden; beim Systemstart kümmert sich das Skript `/etc/init.d/networking` um die Initialisierung der Schnittstellen. (Alternativ macht das auch `udev`, nur müssen die betreffenden Schnittstellen dann in einer Zeile wie »`allow-hotplug eth0`« aufgeführt werden. Interessant ist das für Netzwerkkarten, die nicht immer zur Verfügung stehen, etwa USB-basierte Ethernet- oder UMTS-Adapter.) – Zeilen mit `up` am Anfang enthalten Kommandos, die beim Start der Schnittstelle ausgeführt werden (in der Reihenfolge, wie sie in der Datei stehen), Zeilen mit `down` am Anfang entsprechend Kommandos für den Stopp. Mehr Beispiele für die fremdartigen und wundervollen Dinge, die mit dem Debian-Netzkonfigurationsmechanismus möglich sind, finden Sie in `interfaces`(5) und der Datei `/usr/share/doc/ifupdown/examples/network-interfaces.gz`.

Der YaST als zentrales Konfigurationswerkzeug der Novell/SUSE-Distributionen enthält selbstverständlich auch Module zur Netzkonfiguration (Netzwerkgeräte/Netzwerkkarte). Einstellungen, die mit dem YaST vorgenommen werden, legt dieser häufig in Form von Variablen in Dateien unterhalb von `/etc/sysconfig` ab, wo Init-Skripte oder das Programm `SuSEconfig` sie auslesen. Die Netzkonfiguration im besonderen wird (und das können Sie auch manuell) im Verzeichnis `/etc/sysconfig/network` abgelegt. Hier befindet sich für jede Schnittstelle eine Datei namens `ifcfg-`⟨*Schnittstelle*⟩ (also zum Beispiel `ifcfg-eth0`), die die Einstellungen für die betreffende Schnittstelle enthält. Das kann ungefähr so aussehen:

```
BOOTPROTO='static'                                  oder dhcp (unter anderem)
BROADCAST='192.168.0.255'
ETHTOOL_OPTIONS=''
IPADDR='192.168.0.2'
MTU=''
NAME='79c970 [PCnet32 LANCE]'                       Name im YaST
                                                    (VMware läßt grüßen)
NETMASK='255.255.255.0'                             Oder PREFIXLEN=24
NETWORK='192.168.0.0'
REMOTE_IPADDR=''                                    Gegenstelle bei PPP
STARTMODE='auto'                                    oder manual, hotplug, ...
USERCONTROL='no'
```

(Eine ausführliche Erklärung steht in `ifcfg`(5).) Allgemeine Einstellungen für die Netzkonfiguration stehen in `/etc/sysconfig/network/config`. – Auch die SUSE-Distributionen unterstützen Kommandos namens `ifup` und `ifdown`, die allerdings subtil anders funktionieren als die von Debian GNU/Linux. Zumindest die grundlegenden Aufrufe wie »`ifup eth0`« sind gleich, aber schon »`ifup -a`« geht nicht – um alle Schnittstellen zu starten oder anzuhalten, müssen Sie »`rcnetwork start`« oder »`rcnetwork stop`« sagen. (Zum Trost funktioniert auch »`rcnetwork start eth0`«.) `rcnetwork` ist SUSE-typisch natürlich nichts anderes als ein symbolisches Link auf das Init-Skript `/etc/init.d/network`.

Routen können Sie bei den Novell/SUSE-Distributionen über die Datei `/etc/sysconfig/network/routes` konfigurieren. Der Inhalt der Datei (hier passend zum oben verwendeten Beispiel) ähnelt der Darstellung des Befehls `route`:

```
# cat /etc/sysconfig/network/routes
10.10.3.0       192.168.0.1     255.255.255.0   eth0
112.22.3.4      0.0.0.0         255.255.255.255 ppp0
default         112.22.3.4      -               -
```

Soll kein Gateway verwendet werden, lautet der Eintrag »`0.0.0.0`«, nicht gesetzte Netzmasken oder Schnittstellennamen werden durch ein »`-`« dargestellt. Auch die Routen werden durch Aufruf von »`rcnetwork restart`« gesetzt. Zu den letzten beiden Routen im Beispiel ist zu sagen, dass Punkt-zu-Punkt-Routen für Wählverbindungen in der Regel dynamisch von den entsprechenden Daemons (etwa `pppd`) gesetzt werden. – Wenn Sie Routen für einzelne Schnittstellen definieren möchten, können Sie die entsprechenden Zeilen statt in die `routes`-Datei auch in eine Datei namens `ifroute-`⟨*Schnittstelle*⟩ (also zum Beispiel `ifroute-eth0`) tun. Die vierte Spalte (die mit dem Schnittstellennamen) wird dabei durch den Namen der Schnittstelle ersetzt, falls Sie sie in der Datei leer lassen.

Bei Fedora und den anderen Red-Hat-Distributionen existiert ähnlich wie bei SUSE ein Verzeichnis `/etc/sysconfig`, in dem sich Dateien befinden, in denen diverse Variablen gesetzt werden. Wie bei SUSE existieren Dateien in der Art von `ifcfg-eth0` für die Konfiguration jeder Schnittstelle, nur dass

sie sich in einem Verzeichnis namens `/etc/sysconfig/network-scripts` befinden. Die SUSE-Dateien sind aber nicht 1 : 1 übertragbar, da sie sich von den Red-Hat-Dateien im internen Aufbau unterscheiden. Bei Red Hat könnten Sie unsere Beispielkonfiguration für `eth0` etwa wie folgt realisieren: In `/etc/sysconfig/network-scripts/ifcfg-eth0` steht

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.168.0.0
NETMASK=255.255.255.0
IPADDR=192.168.0.2
USERCTL=no
```

Die `ifup`- und `ifdown`-Kommandos gibt es auch bei Fedora, aber auch hier können Sie immer nur eine Schnittstelle auf einmal starten oder anhalten.

Statische Routen können Sie bei Red Hat in eine Datei in `/etc/sysconfig/network-scripts` tun, die `route-`⟨*Schnittstelle*⟩ heißt (also zum Beispiel `route-eth0`). In diesem Fall ist das Format wie folgt:

```
ADDRESS0=10.10.3.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
```

(zusätzliche Routen verwenden entsprechend `ADDRESS1`, `NETMASK1`, …, `ADDRESS2` und so weiter). Es gibt auch noch ein älteres Dateiformat, in dem einfach jede Zeile der Datei an »`ip route add`« angehängt wird. Dabei bieten sich also Zeilen an wie

```
10.10.3.0/24 via 192.168.0.1
```

Zu allem Überfluss können Sie auch noch in `/etc/sysconfig/static-routes` statische Routen angeben, ohne sich auf einzelne Schnittstellen beziehen zu müssen. Die Zeilen in dieser Datei werden nur beachtet, wenn sie mit dem Schlüsselwort `any` anfangen; der Rest der Zeile wird an »`route add -`« angehängt (Konsistenz? Wer braucht Konsistenz?), so dass aus einer Zeile wie

```
any net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
```

das Kommando

```
route add -net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
```

resultiert.

4.3 DHCP

DHCP, das »Dynamic Host Configuration Protocol«, dient dazu, Ihnen als Administrator die Mühe abzunehmen, die passenden Netzparameter auf jeder einzelnen Station im Netz konfigurieren zu müssen. Statt dessen holt ein Linux-Rechner sich die Netzparameter – neben der IP-Adresse mit Zubehör typischerweise die Adresse eines Default-Routers und eines oder mehrerer DNS-Server – von einem entfernten DHCP-Server, wenn die Netzwerkkarte gestartet wird.

Voraussetzung dafür, dass das funktioniert, ist natürlich ein vorhandener DHCP-Server. Die Installation und den Betrieb eines DHCP-Servers zu erklären würde den Umfang dieser Unterlage leider sprengen. Sollten Sie jedoch einen der gängigen DSL-Router für Ihren Internet-Zugang benutzen oder in der Firma auf eine kompetente Netzwerk-Abteilung zurückgreifen können, dann ist das nich wirklich Ihr Problem, da die nötige Funktionalität dann in der Regel vorgekocht zur Verfügung steht oder sich leicht aktivieren läßt.

Um DHCP zur Konfiguration zu verwenden, müssen Sie die Konfiguration der gängigen Linux-Distributionen nur geringfügig anpassen:

Setzen Sie bei Debian GNU/Linux oder Ubuntu einfach in `/etc/network/interfaces` statt

```
iface eth0 inet static
```

und den darauffolgenden Zeilen mit den Adressen- und Routeninformationen die Zeile

```
iface eth0 inet dhcp
```

ein. Adresse, Netzmaske und Default-Route werden dann vom DHCP-Server bezogen. Sie können natürlich weiterhin mit `up` und `down` Kommandos ausführen, wenn die Verbindung steht oder bevor sie abgebaut wird.

Bei den Novell/SUSE-Distributionen setzen Sie in der Datei mit der Konfiguration für die betreffende Schnittstelle (`ifcfg-eth0` oder so) statt

```
BOOTPROTO='static'
```

den Parameter

```
BOOTPROTO='dhcp'
```

Die Felder `BROADCAST`, `IPADDR`, `NETMASK` und `NETWORK` lassen Sie einfach leer.

Bei Fedora und den anderen Red-Hat-Distributionen setzen Sie zur Verwendung von DHCP in der Konfigurationsdatei für die Schnittstelle statt

```
BOOTPROTO=none
```

den Parameter

```
BOOTPROTO=dhcp
```

Die Adressenparameter können Sie dann einfach weglassen.

Allgemein unterstützen die Netzkonfigurationsmethoden der Distributionen diverse andere Optionen, unter anderem VLAN (auf demselben Kabel werden mehrere »virtuelle« Netze übertragen, die einander nicht sehen), Verschlüsselung oder Bonding (mehrere Netzwerkkarten arbeiten parallel, für mehr Kapazität und/oder Ausfallsicherheit). Wichtig ist auch der Fall, dass ein mobiler Rechner flexibel an mehreren Netzen teilnehmen kann, etwa im Büro und daheim. Die gebotenen Optionen unterscheiden sich stark von Distribution zu Distribution und können hier nicht im Detail besprochen werden.

4.4 IPv6-Konfiguration

Um Ihren Rechner in ein IPv6-Netz zu integrieren, müssen Sie im Idealfall gar nichts machen: Der Mechanismus der »zustandslosen Adressen-Autokonfiguration« (*stateless address autoconfiguration*, SLAAC) macht es möglich, dass alles automatisch vonstatten geht. Bei IPv6 spielt SLAAC in etwa die Rolle, die DHCP bei IPv4 spielt, jedenfalls für einfache Anwendungen.

SLAAC

Vorgehensweise

Wenn eine neue IPv6-Schnittstelle aktiviert wird, erzeugt die Station zuerst die passende verbindungslokale Adresse. Dabei wird das Präfix `fe80::/64` angenommen und die Stationsadresse aus der MAC-Adresse der Schnittstelle abgeleitet[1]. Anschließend verschickt die Station eine verbindungslokale »Router-Aufforderung« (*router solicitation*, RS) über die Schnittstelle an die Multicast-Adresse `ff02::2`, die alle Router im Subnetz anspricht. Diese bringt den oder die Router auf dem physikalischen Netz der Schnittstelle dazu, über »Router-Ankündigungen« (*router advertisements*, RA) mitzuteilen, welche Präfixe sie routen können. Die Station konstruiert auf deren Basis zusätzliche (beispielsweise global sichtbare) Adressen für die Schnittstelle. – RS und RA sind Bestandteile des »Neighbor Discovery Protocol« (NDP), das wiederum Teil von ICMPv6, dem IPv6-Pendant von ICMP, ist. RAs und damit die davon abgeleiteten IPv6-Adressen bleiben nur für eine gewisse Zeit gültig, wenn sie nicht erneuert werden. Router versenden RAs auch unaufgefordert in periodischen Abständen; die RS dient nur dazu, dass Sie für eine neue Schnittstelle nicht die nächste unaufgeforderte RA abwarten müssen, sondern die nötigen Informationen sofort bekommen können.

Vorteile

Der Vorteil dieses Ansatzes ist, dass er ohne die explizite Konfiguration eines DHCP-Servers auskommt. Auch Redundanz ist leicht zu erreichen, indem man mehrere Router im Subnetz konfiguriert. Außerdem müssen die Router sich nicht wie bei DHCP merken, welche Station gerade welche IP-Adresse zugeteilt bekommen hat (deshalb »zustandslos«). Das Ganze heißt aber nicht, dass Sie bei IPv6 ganz auf DHCP verzichten können (es gibt DHCPv6), da es wichtige Netzwerkinformationen gibt, die SLAAC Ihnen nicht liefert (Stichwort: DNS-Server – wobei es hierfür einen neuen, noch nicht weit unterstützten Standard gibt).

Adressen abfragen

Sie können die Adressen abfragen, die das System einer Schnittstelle zugewiesen hat:

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500▷
 ◁ qdisc pfifo_fast state UP qlen 1000
  link/ether 70:5a:b6:9c:40:6a brd ff:ff:ff:ff:ff:ff
  inet 192.168.178.130/24 brd 192.168.178.255 scope global eth0
  inet6 2001:db8:56ee:0:725a:b6ff:fe9c:406a/64 scope global dynamic
     valid_lft 6696sec preferred_lft 3096sec
  inet6 fe80::725a:b6ff:fe9c:406a/64 scope link
     valid_lft forever preferred_lft forever
```

Sie sehen hier sowohl die verbindungslokale Adresse (»`scope link`«, beginnend mit `fe80::`) als auch eine global sichtbare Adresse (»`scope global dynamic`«, beginnend mit `2001:`), die die Schnittstelle über SLAAC erhalten hat. Wenn Sie genau hinschauen, können Sie auch die MAC-Adresse (in der `link/ether`-Zeile) mit den Stationsteilen der IPv6-Adressen korrelieren.

Privatsphäre

Die von Ihrer MAC-Adresse abgeleiteten Stationsteile Ihrer global sichtbaren IPv6-Adressen sind übrigens ein potentielles Problem, was Ihre Privatsphäre angeht: Wenn Sie immer mit derselben Adresse im Netz unterwegs sind, ist es leicht, Ihre Aktionen (aufgerufene Web-Seiten und ähnliches) dieser Adresse zuzuordnen. Auch wenn Sie, wie es so schön heißt, nichts zu verbergen haben, kann Ihnen niemand gewisse Bauchschmerzen verdenken, die Ihnen das schon aus Prinzip

[1] Das Verfahren dafür ist wie folgt: Betrachten Sie die MAC-Adresse *mn*:*op*:*qr*:*st*:*uv*:*wx*. Das (von links gezählt) 3. Bit von *n*, das bei MAC-Adressen immer 0 ist, wird auf 1 gesetzt (wir nennen das Resultat *n*'), und die Stationsadresse ist dann *mn*'*op*:*qr*`ff:fe`*st*:*uvwx*. Aus der MAC-Adresse `70:5a:b6:9c:40:6a` wird also die Stationsadresse `725a:b6ff:fe9c:406a`.

machen könnte. Abhilfe schaffen die »Privatsphäre-Erweiterungen« (*privacy extensions*), die eine zufällige anderweitig unbenutzte Stationsadresse setzen und in periodischen Abständen neu auswürfeln. Die Privatsphäre-Erweiterungen können Sie für eine Schnittstelle – hier `eth0` – mit `sysctl` aktivieren:

```
# sysctl -w net.ipv6.conf.eth0.use_tempaddr=2
# ip link set dev eth0 down
# ip link set dev eth0 up
```

Um die Einstellung permanent zu machen, können Sie sie in `/etc/sysctl.conf` eintragen.

Manuelle Konfiguration

Schließlich ist es natürlich auch möglich, IP-Adressen manuell zu vergeben. Das können Sie entweder mit `ifconfig` erledigen

```
# ifconfig eth0 inet6 add 2001:db8:abcd::1/64
```

oder mit `ip`:

```
# ip addr add 2001:db8:abcd::1/64 dev eth0
```

Wie Sie diese Konfiguration permanent machen können, hängt von Ihrer Distribution ab; die Techniken dafür entsprechen weitgehend den in Abschnitt 4.2 diskutierten.

4.5 Namensauflösung und DNS

Das DNS oder »Domain Name System« ist eine der Grundlagen für die Skalierbarkeit des Internet. Seine Aufgabe besteht darin, Rechnern textuelle Namen zuzuordnen und bei Bedarf die zugehörigen IP-Adressen herauszufinden (oder umgekehrt). Dies realisiert es im wesentlichen über eine weltweit verteilte »Datenbank« aus DNS-Servern.

Inzwischen kümmert sich das DNS auch um zahlreiche andere Aufgaben, vom Aufspüren der für eine Domain zuständigen Mail-Server bis zur Hilfe bei der Vermeidung von Spam.

Resolver

Programme auf einem Linux-Rechner reden in der Regel nicht direkt mit dem DNS, sondern bedienen sich dafür eines »Resolvers«. Dieser ist üblicherweise Teil der C-Bibliothek. Die zentrale Konfigurationsdatei für den Resolver heißt `/etc/resolv.conf`. Hier werden zum Beispiel die DNS-Server konfiguriert, die der Resolver konsultieren soll. Dazu existieren fünf Hauptdirektiven:

`domain` ⟨*Name*⟩ (lokale Domain) Anhand dieses Eintrags versucht der Resolver, unvollständige Rechnernamen (typisch solche, die keinen Punkt enthalten) um einen Domainanteil zu ergänzen.

Was ein unvollständiger Name ist, wird von der Option `ndots` (siehe Tabelle 4.1) bestimmt.

`search` ⟨$Domain_1$⟩ ⟨$Domain_2$⟩ **…** (Suchliste) Alternativ zu einem einzigen Eintrag mittels `domain` kann mit `search` auch eine Liste mit mehreren Ergänzungen für unvollständige Rechnernamen angegeben werden. Die Einträge in der Liste werden durch Leerzeichen getrennt. Zunächst wird versucht, den unveränderten Rechnernamen aufzulösen. Wenn dies scheitert, werden die Listeneinträge der Reihe nach angehängt und diese Namen ausprobiert. `domain` und `search` schließen sich gegenseitig aus; tauchen beide in der Konfiguration auf, gilt der textuell letzte Eintrag in der Datei.

Tabelle 4.1: Optionen innerhalb `/etc/resolv.conf`

Option	Wirkung
`debug`	Die regulären Betriebsmeldungen werden auf `stdout` ausgegeben (meist nicht unterstützt).
`ndots` ⟨*n*⟩	Die minimale Anzahl von Punkten im Namen, bei welcher der Resolver direkt nachsieht, ohne auf die Suchliste zuzugreifen.
`attempts` ⟨*n*⟩	Die Anzahl der Anfragen an einen Server, bis der Resolver aufgibt. Maximalwert ist 5.
`timeout` ⟨*n*⟩	Der Anfangs-Timeout für Abfrageversuche in Sekunden. Maximalwert ist 30.
`rotate`	Nicht nur der erste, sondern alle Server werden abwechselnd befragt.
`no-check-names`	Deaktiviert die standardmäßige Überprüfung, ob zurückgelieferte Hostnamen nur gültige Zeichen enthalten.

```
nameserver 192.168.10.1
nameserver 192.168.0.99
search     foo.example.com bar.example.com example.com
```

Bild 4.1: Beispiel für `/etc/resolv.conf`

`nameserver` ⟨*IP-Adresse*⟩ (Lokaler DNS-Server) Der lokale Resolver wird den hier eingetragenen DNS-Server befragen. Es sind bis zu drei `nameserver`-Direktiven erlaubt, die bei Bedarf nacheinander abgefragt werden.

`sortlist` ⟨*IP-Adresse*⟩[/⟨*Netzmaske*⟩] (Sortierung) Falls zu einem Rechnernamen mehrere Adressen zurückgeliefert werden, so wird die hier eingetragene bevorzugt. Bis zu 10 Einträge sind in der Sortierliste möglich.

`options` ⟨*Option*⟩ (Optionen) Hiermit lassen sich besondere Resolver-Einstellungen vornehmen, die in der Tabelle 4.1 mit den Vorgabewerten aufgelistet sind. In der Praxis werden diese selten bis nie verändert.

Eine typische `/etc/resolv.conf`-Datei sehen Sie in Bild 4.1.

Die Alternative zum DNS ist die »lokale« Auflösung von Rechnernamen und IP-Adressen über die Datei `/etc/hosts`. Als ausschließliche Methode zur Namensauflösung ist sie höchstens in kleinen Netzen interessant, die nicht ans Internet angebunden sind, aber erwähnen sollten wir sie trotzdem – wenn Sie es nur mit ein paar Rechnern zu tun haben, dann ist es möglicherweise simpler, nur die DNS-Client-Seite zu konfigurieren und Ihren eigenen Rechnern per `/etc/hosts` Namen und Adressen zu geben. Sie müssen dann nur darauf achten, dass die Datei auf allen beteiligten Rechnern gleich ist.

Für kleine Netze empfiehlt sich das Programm `dnsmasq`, das den Inhalt einer `/etc/hosts`-Datei lokal über DNS verfügbar macht und alle anderen DNS-Anfragen ans »echte« DNS weiterleitet. Nebenbei fungiert es auch noch als DHCP-Server.

Bei der Datei `/etc/hosts` handelt es sich um gewöhnlichen ASCII-Text, in dem neben Kommentarzeilen, die mit »#« eingeleitet werden, zeilenweise Einträge vorgenommen werden können. Diese enthalten spaltenweise mindestens die IP-Adresse und den vollständigen Namen (FQDN) des Rechners. Daneben ist es erlaubt, noch Kurznamen für den Rechner anzugeben. Als Trennzeichen der einzelnen Spalten dienen Leer- und/oder Tabulatorzeichen, also »Freiplatz« (*white space*). Bild 4.2 zeigt den Inhalt einer exemplarischen `/etc/hosts`-Datei.

```
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#
# special IPv6 addresses

127.0.0.1       localhost
192.168.0.99    linux.example.com  linux
```

Bild 4.2: Die Datei `/etc/hosts` (SUSE)

In der Frühzeit des Internet – etwa bis in die frühen 1980er Jahre – wurde im wesentlichen eine große `/etc/hosts`-Datei zur Namensauflösung herangezogen. Auch Domains waren noch nicht erfunden. Damals hatte das Internet noch weniger Stationen (Tausende statt Abermillionen), aber die Wartung und Verteilung aktueller Versionen dieser Datei entwickelte sich zu einem zunehmenden Problem. Daher DNS.

Über die genaue Methodik der Namensauflösung durch die C-Bibliothek entscheidet tatsächlich eine Datei namens `/etc/nsswitch.conf`. Darin ist beispielsweise festgelegt, welche Dienste zur Namensauflösung in welcher Reihenfolge verwendet werden. Daneben finden sich Einträge zur Auflösung von Benutzernamen, Gruppen usw., die uns an dieser Stelle nicht weiter interessieren. Eine genaue Beschreibung von Syntax und Funktionsweise können Sie in `nsswitch.conf`(5) einsehen.

Dienste zur Namensauflösung

Der für die Auflösung von Rechnernamen interessante Teil von `/etc/nsswitch.conf` kann etwa so aussehen:

```
hosts: files dns
```

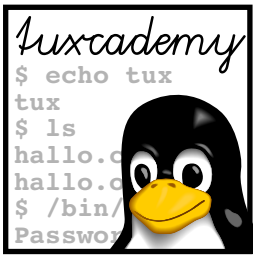
Hier wird also zunächst versucht, Rechnernamen mit Hilfe der lokalen Dateien (namentlich `/etc/hosts`) aufzulösen. Erst wenn dies scheitert, kommt das DNS zum Zug.

Kommandos in diesem Kapitel

dnsmasq	Ein einfacher DHCP- und cachender DNS-Server für kleine Installationen	dnsmasq(8)	82
ifconfig	Konfiguriert Netzwerk-Schnittstellen	ifconfig(8)	71
ifdown	Schaltet eine Netzwerk-Schnittstelle aus (Debian)	ifdown(8)	76
ifup	Schaltet eine Netzwerk-Schnittstelle ein (Debian)	ifup(8)	76
ip	Verwaltet Netzwerkschnittstellen und Routing	ip(8)	75
route	Verwaltet die statische Routing-Tabelle im Linux-Kern	route(8)	72

Zusammenfassung

- Treiber für Netzwerkkarten lädt der Linux-Kernel heute bei Bedarf über die `udev`-Infrastruktur.
- Das Kommando `ifconfig` dient zur Konfiguration von Interface-Parametern auf niedriger Ebene. Sie können damit auch das Loopback-Interface konfigurieren und Aliasnamen für Interfaces vergeben.
- Routen geben an, wie IP-Datagramme an ihren Empfänger geleitet werden.
- Zur Konfiguration von Routen dient das Kommando `route`.
- Das Kommando `ip` ist ein komfortabler Ersatz für `ifconfig` und `route`.
- Die verschiedenen Linux-Distributionen bieten unterschiedliche Methoden zur dauerhaften Netzkonfiguration an.
- Mit DHCP können Linux-Rechner Netzparameter dynamisch von einem zentralen Server beziehen.
- Gängige Mechanismen zur Namensauflösung beruhen auf dem DNS oder auf lokalen Konfigurationsdateien.
- Die Reihenfolge der Namensauflösung wird in der Datei `/etc/nsswitch.conf` eingerichtet.



5

Fehlersuche und Fehlerbehebung im Netz

Inhalt

5.1 Einführung 86
5.2 Lokale Probleme 86
5.3 Erreichbarkeit von Stationen prüfen mit `ping` 87
5.4 Wegleitung testen mit `traceroute` und `tracepath` 89
5.5 Dienste überprüfen mit `netstat` und `nmap` 92
5.6 DNS testen mit `host` und `dig` 95
5.7 Andere nützliche Diagnosewerkzeuge 98
5.7.1 `telnet` und `netcat` 98
5.7.2 `tcpdump` 100
5.7.3 `wireshark` 100

Lernziele

- Strategien zur Fehlersuche im Netz kennen
- Werkzeuge wie `ping`, `traceroute` und `netstat` zur Problemdiagnose einsetzen können
- Einfache Fehler in der Netzkonfiguration beheben können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 3)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)

5.1 Einführung

Systemadministratoren lieben es: Gerade haben Sie sich gemütlich mit einer schönen Tasse Kaffee oder Tee vor Ihrem Rechner niedergelassen und gedenken in aller Ruhe die neuesten Nachrichten auf LWN.net zu lesen, da steht eine lästige Person in der Tür: »Ich komme nicht ins Netz!« Mit dem Idyll ist es also erst mal wieder vorbei. Aber was tun?

Die Vernetzung von Rechnern ist ein komplexes Thema, und so sollten Sie sich nicht wundern, wenn alles Mögliche schief gehen kann. In diesem Abschnitt zeigen wir Ihnen die wichtigsten Werkzeuge und Strategien, mit denen Sie Probleme finden und ausbügeln können.

5.2 Lokale Probleme

Als erstes sollten Sie sich überzeugen, dass die Netzwerkkarte vorhanden ist und erkannt wird. (Zuallererst hilft vielleicht auch ein diskreter Blick hinter den Rechner, ob das Kabel noch im richtigen Port steckt oder nicht vielleicht die Damen und Herren von der Putzkolonne ein bisschen »kreative Rekonfiguration« gespielt haben.)

Schauen Sie nach, was »`ifconfig -a`« ausgibt. Mit diesem Parameter abgesetzt liefert das Programm Ihnen eine Übersicht über *alle* Netz-Schnittstellen des Rechners, auch die aktuell nicht aktiv konfigurierten, und zumindest `lo` und `eth0` (bei einem über Ethernet vernetzten Rechner) sollten zu sehen sein. Ist das nicht der Fall, dann haben Sie Ihr erstes Problem schon gefunden: Möglicherweise stimmt mit dem Treiber oder der Erkennung der Karte etwas nicht.

Alternativ tut es natürlich auch das Kommando »`ip link list`«, das ebenfalls alle vorhandenen Netz-Schnittstellen auflistet, unabhängig vom Konfigurationszustand.

Wenn Sie statt `eth0` nur etwas sehen wie `eth1`, dann kann es sein, dass die Netzwerkkarte im Rechner ausgewechselt wurde und `udev` der neuen Netzwerkkarte mit ihrer ebenso neuen MAC-Adresse auch einen neuen Schnittstellennamen verpasst hat. Bei einigermaßen fest in den Rechner eingebauten Netzwerkkarten sollte das eigentlich nicht passieren (oder wenn, dann nur, weil Sie als Administrator es eigenhändig gemacht haben), aber vielleicht haben Ihre Kollegen heimlich ihre PC(MCIA)-Netzwerkkarten oder USB-UMTS-Sticks vertauscht. Die Abhilfe besteht darin, in der Datei `/etc/udev/rules.d/70-persistent-net.rules` (oder so ähnlich) die Zeile mit dem alten Gerät zu löschen und in der Zeile mit dem neuen Gerät den Schnittstellennamen zu korrigieren. Starten Sie anschließend `udev` neu.

Wenn in der Ausgabe von `ifconfig` oder `ip` überhaupt nichts auftaucht, was Ihrer Netzwerkkarte ähnlich sieht, dann prüfen Sie mit `lsmod`, ob das passende Treibermodul geladen wurde. Wenn Sie nicht wissen, welches das passende Treibermodul überhaupt *ist*, können Sie in der Ausgabe von »`lspci -k`« nach dem Eintrag für Ihre Netzwerkkarte suchen. Dieser könnte ungefähr so aussehen:

```
02.00.0 Ethernet controller: Broadcom Corporation NetXtreme▷
 ◁ BCM5751 Gigabit Ethernet PCI Express (rev 01)
        Kernel driver in use: tg3
        Kernel modules: tg3
```

In diesem Fall sollten Sie sich davon überzeugen, dass das Modul `tg3` geladen ist.

Übungen

5.1 [!1] Welche Netzwerkkarten stehen in Ihrem System zur Verfügung? Sind sie alle konfiguriert? Welche Treiber werden benutzt?

5.3 Erreichbarkeit von Stationen prüfen mit ping

Wenn die Ausgabe von ifconfig die Schnittstelle zeigt und auch die Parameter, die dort zu sehen sind, vernünftig scheinen (schauen Sie vor allem nach der IP-Adresse, der Netzmaske – sehr wichtig – und der Broadcast-Adresse), dann ist es Zeit für ein paar Erreichbarkeitstests. Das einfachste Werkzeug dafür ist ein Programm namens ping, das eine IP-Adresse (oder einen DNS-Namen) übernimmt und versucht, der betreffenden Station ein ICMP-ECHO-REQUEST-Datagramm zu schicken. Diese Station sollte darauf mit einem ICMP-ECHO-REPLY-Datagramm antworten, das ping bemerkt und Ihnen meldet.

Als erstes sollten Sie testen, ob der Rechner mit sich selbst reden kann:

```
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.040 ms
                                                    Abbruch mit Strg+c ...
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.032/0.037/0.040/0.006 ms
```

Diese Ausgabe zeigt Ihnen, dass der »andere Rechner« (in diesem Fall nur die Loopback-Schnittstelle 127.0.0.1) zuverlässig erreicht werden konnte (es sind keine Pakete verlorengegangen).

Was soll »56(84) bytes of data« heißen? Ganz einfach: Ein IP-Datagrammkopf ohne Optionen ist 20 Bytes lang. Dazu kommt noch der Kopf eines ICMP-ECHO-REQUEST-Datagramms im Werte von 8 Bytes. Das erklärt schon mal die Differenz zwischen 56 und 84. Die magische Zahl 56 kommt daher, dass ping normalerweise dafür sorgt, dass im IP-Datagramm genau 64 Bytes Daten übertragen werden, nämlich der 8-Byte-ICMP-Kopf und 56 Bytes »Füllstoff«. Wenn »genug« Füllstoff vorhanden ist, namentlich mindestens die Größe einer struct timeval in C (acht Bytes oder so), dann verwendet ping den Anfang des Füllstoffs für einen Zeitstempel, um die Paketlaufzeit zu berechnen.

Im nächsten Schritt sollten Sie die Schnittstelle Ihrer Netzwerkkarte »anpingen«. Die Ausgabe sollte hier im wesentlichen so aussehen wie beim Ping auf die Loopback-Schnittstelle.

Wenn Sie hier angekommen sind, ohne dass Fehlermeldungen auftreten, legt das den Schluss nahe, dass die elementaren Netzfunktionen des Rechners funktionieren. Die verbleibenden Möglichkeiten für Probleme liegen entweder anderswo im Netz oder weiter oben im Protokollstapel Ihres Rechners.

Der nächste Ping geht auf das Default-Gateway (oder einen anderen Rechner im lokalen Netz). Wenn das gar nicht funktioniert, dann könnte zum Beispiel die Netzmaske falsch gesetzt sein (auch auf dem anderen Rechner!?). Ebenfalls möglich sind Hardwareprobleme, etwa ein heftiger Knick im Kabel oder ein kaputter Stecker – so etwas würde auch eine Verbindung erklären, die mal funktioniert und mal nicht.

Tabelle 5.1: Wichtige Optionen von `ping`

Option	Bedeutung
`-a`	Hörbare Pings
`-b` ⟨*Netzadresse*⟩	Broadcast-Ping
`-c` ⟨*Anzahl*⟩	Anzahl der zu sendenden Datagramme (`ping` beendet sich danach von selbst)
`-f`	»Flut-Ping«: Für jedes versandte `ECHO-REQUEST`-Datagramm wird ein Punkt ausgegeben, für jedes empfangene `ECHO-REPLY` ein Backspace-Zeichen. Das Ergebnis ist eine Reihe von Punkten, aus der Sie ersehen können, wieviele Datagramme bei der Übertragung fallen gelassen werden. Wenn nicht gleichzeitig die Option `-i` angegeben wurde, verschickt `ping` mindestens 100 Datagramme pro Sekunde (mehr, wenn das Netz mehr verarbeiten kann). Das darf allerdings nur `root`; normale Benutzer sind auf ein minimales Intervall von 0,2 Sekunden beschränkt.
`-i` ⟨*Zeit*⟩	Wartet ⟨*Zeit*⟩ Sekunden zwischen dem Verschicken zweier Datagramme. Der Standardwert ist eine Sekunde, außer bei Flut-Ping als `root`.
`-I` ⟨*Absender*⟩	Setzt die Absendeadresse für die Datagramme. ⟨*Absender*⟩ darf eine IP-Adresse sein oder der Name einer Schnittstelle (dann wird deren IP-Adresse verwendet).
`-n`	Anzeige ohne DNS-Namensauflösung
`-s` ⟨*Größe*⟩	Bestimmt die Größe des »Füllstoffs« in Byte; der Standardwert ist 56. Manchmal gibt es Probleme mit sehr großen Datagrammen, die fragmentiert werden müssen, und mit dieser Option kann `ping` so etwas diagnostizieren helfen. (Ganz früher konnte man Rechner mit riesengroßen `ping`-Datagrammen zum Absturz bringen – der berüchtigte *ping of death*.)

Die gängigen rechteckigen Stecker am Ethernet-Kabel werden mit einer Plastezunge in der Buchse fixiert. Diese Zunge bricht leicht ab, und das kann dazu führen, dass der Kontakt nicht 100% hergestellt wird.

»Freiliegende« Kabel sind empfindlich gegenüber scharfkantigen Gegenständen und mögen es auch nicht, wenn Sie (oder Ihre Kollegen) mit dem Bürostuhl über sie hinwegrollern. Einen Verdacht gegen ein Kabel können Sie durch einen prophylaktischen Austausch gegen ein als funktionierend bekanntes Exemplar oder einen Test mit einem Ethernet-Kabelprüfgerät erhärten oder widerlegen. Normalerweise gehören Kabel natürlich in einen Kabelkanal, auf die abgehängte Decke oder unter den hochgestellten Boden.

Jetzt können Sie damit fortfahren, Rechner außerhalb Ihres lokalen Netzes anzupingen. Wenn das klappt, ist das ein gutes Zeichen; wenn Sie überhaupt keine Antworten bekommen, dann haben Sie es entweder mit einem Problem bei der Wegleitung zu tun oder mit einem Firewall, der ICMP-Verkehr à la `ping` zumindest teilweise filtert (was er nicht sollte, aber manche Leute schütten das Kind halt mit dem Badewasser aus).

`ping` unterstützt eine Menge von Optionen, die die Testmöglichkeiten erweitern oder die Arbeitsweise des Programms ändern. Die wichtigsten für Diagnosezwecke sind wahrscheinlich `-f` (Flut-Ping), um intermittierende Netzwerkprobleme schnell zu überprüfen, und `-s`, um eine Größe für die Datagramme angeben zu können.

`-a` kann nützlich sein, wenn Sie unter dem Tisch herumkriechen, um ein wackliges Kabel zu finden.

`ping6` Das entsprechende Kommando zum Test von IPv6 heißt `ping6` und wird im Wesentlichen genauso aufgerufen wie `ping`. Sie müssen nur darauf achten, dass Sie die Schnittstelle angeben, die Sie meinen. Achten Sie auf das „`%eth0`“ am Ende der IPv6-Adresse:

```
$ ping6 fe80::224:feff:fee4:1aa1%eth0
PING fe80::224:feff:fee4:1aa1%eth0(fe80::224:feff:fee4:1aa1)▷
 ◁ 56 data bytes
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=1 ttl=64 time=3.65 ms
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=2 ttl=64 time=4.30 ms
◁◁◁◁◁
```

Gerade bei verbindungslokalen Adressen ist es grundsätzlich möglich, dass mehrere Schnittstellen dieselbe Adresse haben, und deswegen müssen Mehrdeutigkeiten vermieden werden. Ansonsten entsprechen die Optionen von ping6 weitestgehend denen von ping.

Übungen

5.2 [!2] Vergleichen Sie die Laufzeiten für Datagramme zwischen einem ping auf 127.0.0.1 und einem ping auf eine entfernte Station (anderer Rechner im LAN oder Standard-Gateway/DSL-Router/…).

5.3 [2] Wie lange braucht Ihr System, um im Flut-Ping-Modus eine Million Datagramme an sich selbst zu schicken?

5.4 [2] (Falls Ihr lokales Netz IPv6 unterstützt.) Prüfen Sie mit ping6 die Konnektivität zu allfälligen IPv6-Routern in Ihrem LAN (Multicast-Adresse ff02::2). Was für Antworten bekommen Sie zurück?

5.4 Wegleitung testen mit traceroute und tracepath

Wenn Sie eine Station außerhalb Ihres lokalen Netzes nicht mit ping erreichen können, könnte ein Problem mit der Wegleitung dafür verantwortlich sein. Programme wie traceroute und tracepath helfen dabei, solchen Problemen auf den Grund zu gehen.

Der typische Fall ist, dass Sie zwar alle Stationen im lokalen Netz mit ping erreichen können, aber keine jenseits davon. Die naheliegenden Verdächtigen sind dabei zum einen Ihre Default-Route, zum anderen der Rechner, auf den die Default-Route zeigt. Vergewissern Sie sich, dass die Ausgabe von route (oder »ip route list«) die richtige Default-Route liefert. Wenn ein ping auf den Default-Router funktioniert, ein ping auf einen Rechner jenseits des Default-Routers dagegen nicht, dann ist möglicherweise mit dem Default-Router etwas nicht in Ordnung. Prüfen Sie, ob ein anderer Rechner über den Default-Router hinaus Stationen erreichen kann und ob Ihr Rechner vom Default-Router aus zu erreichen ist. (Denken Sie immer auch daran, dass auf dem Default-Router möglicherweise ein Paketfilter läuft, der ICMP blockiert.)

Eine andere Sorte von Problem kann sich ergeben, wenn Sie mit dem Router, der Sie mit dem Internet verbindet, nicht direkt, sondern nur über einen anderen Router Kontakt haben. In diesem Fall kann es passieren, dass Sie zwar ping-Datagramme an den Internet-Router schicken können, dessen Antworten Sie aber nicht erreichen, weil er keine Route hat, die Verkehr in »Ihr« Netz über den dazwischenliegenden Router leitet.

traceroute ist im Prinzip eine erweiterte Form von ping. Hier wird eine entfernte Station nicht einfach auf Lebenszeichen abgefragt, sondern der Weg angezeigt, den die Datagramme im Netz nehmen. Es wird verfolgt, über welche Router ein Datagramm läuft, und wie die Verbindung zu den benutzten Routern ist.

Das ganze beruht im Gegensatz zu `ping` nicht auf ICMP, sondern (traditionell) auf UDP. `traceroute` schickt drei UDP-Datagramme an willkürliche Ports auf der Zielstation (man hofft, dass nicht auf allen drei Ports irgendein Server lauscht). Die ersten drei Datagramme haben eine TTL von 1, die nächsten drei eine TTL von 2 und so weiter. Der erste Router auf dem Weg zum Ziel verringert die TTL jeweils um 1. Für die erste Runde von Datagrammen, die von Anfang an nur die TTL 1 hatten, ist damit Schicht – sie werden verworfen und der Absender bekommt eine ICMP-`TIME-EXCEEDED`-Nachricht, die natürlich (als IP-Datagramm) auch die IP-Adresse des Routers enthält. Die zweiten drei Datagramme werden vom zweiten Router verworfen und so fort. Auf diese Wiese können Sie den genauen Weg der Datagramme zur Zielstation verfolgen. Die Zielstation selbst schickt natürlich kein `TIME-EXCEEDED`, sondern ein `PORT-UNREACHABLE`, so dass `traceroute` erkennen kann, dass es am Ziel angekommen ist.

Aussehen kann das ungefähr so:

```
$ traceroute www.linupfront.de
traceroute to www.linupfront.de (31.24.175.68), 30 hops max,▷
 ◁ 60 byte packets
 1  fritz.box (192.168.178.1)  5.959 ms  5.952 ms  5.944 ms
 2  217.0.119.34 (217.0.119.34)  28.889 ms  30.625 ms  32.575 ms
 3  87.186.202.242 (87.186.202.242)  35.163 ms  36.961 ms  38.551 ms
 4  217.239.48.134 (217.239.48.134)  41.413 ms  43.002 ms  44.908 ms
 5  xe-11-0-1.fra29.ip4.gtt.net (141.136.101.233)  46.769 ms  ▷
 ◁ 49.231 ms  51.282 ms
 6  xe-8-1-2.fra21.ip4.gtt.net (141.136.110.101)  53.412 ms ▷
 ◁ xe-0-2-3.fra21.ip4.gtt.net (89.149.129.37)  49.198 ms ▷
 ◁ xe-8-1-2.fra21.ip4.gtt.net (141.136.110.101)  52.314 ms
 7  21cloud-gw.ip4.gtt.net (77.67.76.90)  52.547 ms  30.822 ms▷
 ◁  30.018 ms
 8  s0a.linupfront.de (31.24.175.68)  38.127 ms  38.406 ms  38.402 ms
```

Die Ausgabe besteht aus mehreren durchnummerierten Zeilen. Eine Zeile entspricht jeweils einer gesendeten Gruppe von drei Datagrammen. Angezeigt werden immer die Station, von der die `TIME-EXCEEDED`-Nachrichten kamen, sowie die Laufzeit der drei Datagramme.

Sternchen in der Ausgabe bedeuten, dass für eins der Datagramme innerhalb von (normalerweise) 5 Sekunden keine Antwort gekommen ist. Sowas kommt vor.

Vielleicht wundern Sie sich, dass die Ausgabe mit `s0a.linupfront.de` endet, wo wir doch `www.linupfront.de` erreichen wollten. Das ist aber kein Problem; die Web-Präsenz `www.linupfront.de` liegt – zusammen mit ein paar anderen nützlichen Diensten – auf dem Rechner, den wir `s0a.linupfront.de` nennen, und das ist halt die Antwort, die das DNS liefert, wenn man es nach dem Namen für die Adresse `31.24.175.68` fragt.

Aus dem Umstand, dass IP-Netze Paketvermittlung verwenden, folgt theoretisch natürlich, dass die Ausgabe von `traceroute` nur eine Momentaufnahme darstellt. Wenn Sie es gleich wieder probieren, können die neuen Datagramme prinzipiell schon eine ganz andere Route zur Zielstation nehmen. Allerdings kommt das in der Praxis nicht so häufig vor.

Die traditionelle Technik mit UDP-Datagrammen funktioniert heute nicht mehr immer, da es übereifrige Firewalls gibt, die Datagramme an »unwahrscheinliche« UDP-Ports verwerfen. Mit der Option `-I` können Sie `traceroute` dazu bringen, statt UDP ICMP zu benutzen (es arbeitet dann im wesentlichen wie `ping`). Sollten Sie an einen besonders übereifrigen Firewall geraten, der auch ICMP filtert, dann können Sie mit `-T` (kurz für »`-M tcp`«) eine TCP-basierte Technik verwenden. Diese

versucht, den Port 80 auf der Zielstation anzusprechen, und bietet sich besonders an, wenn es sich bei der Zielstation um einen Web-Server handelt. (Sie können sich mit der Option `-p` einen anderen Port wünschen.)

Die »TCP-basierte Technik« baut keine komplette TCP-Verbindung zur Zielstation auf und bleibt darum für Anwendungsprogramme dort unsichtbar. `traceroute` bietet auch noch ein paar andere Methoden an.

Sie können `traceroute` mit IPv6 verwenden, indem Sie die Option `-6` angeben. Eine bequeme Abkürzung dafür ist `traceroute6`. Alles andere bleibt beim Alten.

`traceroute6`

Das Programm `tracepath` macht im Grunde das gleiche wie `traceroute`, verzichtet allerdings auf fast alle trickreichen Optionen und kann auch von normalen Benutzern (ohne `root`-Rechte) aufgerufen werden. Außerdem bestimmt es die »Pfad-MTU« (hierzu gleich mehr). Hier ist eine beispielhafte Ausgabe von `tracepath`:

`tracepath`

```
$ tracepath www.linupfront.de
 1?: [LOCALHOST]                          pmtu 1500
 1:  fritz.box                             13.808ms
 1:  fritz.box                              5.767ms
 2:  p5B0FFBB4.dip0.t-ipconnect.de         11.485ms pmtu 1492
 2:  217.0.119.34                          48.297ms
 3:  87.186.202.242                        46.817ms asymm  4
 4:  217.239.48.134                        48.607ms asymm  5
 5:  xe-11-0-1.fra29.ip4.gtt.net           47.635ms
 6:  xe-7-1-0.fra21.ip4.gtt.net            49.070ms asymm  5
 7:  21cloud-gw.ip4.gtt.net                48.792ms asymm  6
 8:  s0a.linupfront.de                     57.063ms reached
     Resume: pmtu 1492 hops 8 back 7
```

Genau wie `traceroute` gibt `tracepath` die Adressen der Router auf dem Weg zur Zielstation aus. Der Rest der Zeile zeigt die Laufzeit der Datagramme sowie Zusatzinformationen; »`asymm 5`« zum Beispiel bedeutet, dass die Antwort des Routers 5 Hops brauchte statt den 4 der Anfrage, aber diese Information ist nicht immer verläßlich.

Das bringt uns zum Problem der »Pfad-MTU«, das sich etwa wie folgt erklären läßt: IP erlaubt fundamental Datagramme von bis zu 65535 Bytes Länge, aber nicht jedes Zugangsverfahren kann solche Datagramme auf einen Sitz übertragen. Bei Ethernet zum Beispiel sind maximal Frames von 1518 Byte möglich, von denen noch 14 Byte für den Frame-Kopf und 4 Byte für eine Prüfsumme am Ende des Frames abgehen. Das heißt, in ein Ethernet-Frame passen höchstens 1500 Bytes an Nutzdaten, und wenn die IP-Schicht darüber ein größeres Datagramm übertragen will, muss das »fragmentiert«, also auf mehrere Frames aufgeteilt werden. Man sagt, dass die *Maximum Transmission Unit* oder MTU für Ethernet 1500 ist.

Pfad-MTU

Natürlich kann die IP-Implementierung der sendenden Station nicht vorausahnen, welche Übertragungsverfahren auf dem Weg zur Zielstation im Spiel sind und ob eine Fragmentierung nötig ist (und wenn ja, wie groß die Fragmente sein dürfen). Das ergibt sich erst, wenn wirklich Daten geschickt werden. *Eigentlich* sollten Router das ja transparent behandeln – kommt am einen Ende ein Datagramm herein, das zu groß ist, um im ganzen am anderen Ende wieder hinausgeschickt zu werden, könnte der Router es fragmentieren –, aber die Hersteller von Routern drücken sich gerne um diese ressourcenintensive Tätigkeit. Statt dessen werden Datagramme typischerweise mit dem gesetzten »Don't-Fragment«-Bit im Kopf auf die Reise geschickt, das es anderen Routern verbietet, sie weiter zu zerstückeln. Kommt so ein Datagramm dann an einen Punkt, wo es zu groß für den nächsten Hop ist, schickt der betreffende Router per ICMP die Meldung »Ziel unerreichbar; Fragmentierung nötig, aber verboten; MTU wäre *n*«. In diesem Fall kann die sendende Station es nochmal mit kleineren Fragmenten versuchen. Dieses Verfahren heißt »Pfad-MTU-Bestimmung« (engl. *path MTU discovery*).

Das Ganze kann immer noch glorreich schief gehen, nämlich wenn ein übereifriger Firewall auf dem Weg den ICMP-Verkehr blockiert. In diesem Fall kommen die Fehlermeldungen über die zu große MTU nie beim Absender der Datagramme an und der weiß nicht, wie ihm geschieht. In der Praxis führt das zum Beispiel dazu, dass Webseiten nicht korrekt angezeigt werden und/oder Verbindungen »hängenbleiben«. Das Problem tritt vor allem da auf, wo ADSL à la »Deutsche Telekom« im Spiel ist, denn dort wird ein Protokoll namens »PPP over Ethernet« (PPPoE) verwendet, bei dem von den 1500 Bytes der üblichen Ethernet-MTU noch 8 Bytes für Verwaltungsinformationen abgehen. Die Probleme verschwinden normalerweise, wenn Sie die MTU für die betroffene Schnittstelle manuell auf 1492 setzen. Die Gegenstelle orientiert sich dann an diesem Wert.

In Debian GNU/Linux (und Ubuntu) können Sie die MTU für eine statisch konfigurierte Schnittstelle setzen, indem Sie eine `mtu`-Klausel in die Definition der Schnittstelle in `/etc/network/interfaces` aufnehmen:

```
iface eth0 inet static
    ⋖⋖⋖⋖
    mtu 1492
    ⋖⋖⋖⋖
```

Beim nächsten Start der Schnittstelle sollte dieser Wert dann wirksam werden.

Wenn Ihre Schnittstelle über DHCP konfiguriert wird und der DHCP-Server eine falsche MTU liefert (soll vorkommen), dann können Sie in der Datei `/etc/dhcp/dhclient.conf` im `request`-Eintrag die Klausel `interface-mtu` löschen. Dann nimmt Linux bei der nächsten DHCP-Aushandlung den Standardwert 1500 an. Einen anderen Wert können Sie explizit über

```
iface eth0 inet dhcp
    ⋖⋖⋖⋖
    post-up /sbin/ifconfig eth0 mtu 1492
    ⋖⋖⋖⋖
```

einstellen. Alternativ geht natürlich auch

```
iface eth0 inet dhcp
    ⋖⋖⋖⋖
    post-up /sbin/ip link set dev eth0 mtu 1492
    ⋖⋖⋖⋖
```

Bei den SUSE-Distributionen können Sie die MTU in der `ifcfg-`-Datei der betreffenden Netzwerkschnittstelle setzen (es gibt eine `MTU=`-Zeile). Alternativ geht das auch mit dem »`/etc/sysconfig`-Editor« des YaST, unter »Hardware/Network«. Sie müssen die Netzwerkschnittstelle anschließend neu starten (mit `ifdown`/`ifup`) oder den Rechner rebooten.

Die Red-Hat-Distributionen erlauben wie die SUSE eine MTU-Einstellung in der `ifcfg-`-Datei der betreffenden Schnittstelle. Auch hier ist ein Neustart der Schnittstelle nötig, damit die Einstellung wirksam wird.

Wenn Sie IPv6 verwenden: `tracepath6` verhält sich zu `tracepath` wie `traceroute6` zu `traceroute`.

5.5 Dienste überprüfen mit `netstat` und `nmap`

Wenn Sie einen Dienst betreiben wollen, Client-Rechner aber keinen Kontakt mit ihm bekommen, sondern mit Fehlermeldungen wie

```
Unable to connect to remote host: Connection refused
```

abgewiesen werden, sollten Sie sicherstellen, dass der Dienst tatsächlich ordnungsgemäß auf Verbindungen »lauscht«. Das können Sie zum Beispiel mit dem Programm netstat:

```
$ netstat -tul
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address       Foreign Address  State
tcp        0      0 red.example.com:www *:*              LISTEN
tcp        0      0 red.example.com:ftp *:*              LISTEN
tcp        0      0 red.example.com:ssh *:*              LISTEN
```

Die Option -l bringt netstat dazu, nur »lauschende« Programme aufzuführen. Mit den Optionen -t und -u können Sie die Ausgabe auf TCP- bzw. UDP-basierte Dienste beschränken.

In der Ausgabe bedeuten die Spalten das Folgende:

Proto Das Protokoll (tcp, udp, raw, …), das von dem Socket benutzt wird.

Recv-Q Die Anzahl der empfangenen, aber (noch) nicht vom Anwenderprogramm bei diesem Socket abgeholten Bytes.

Send-Q Die Anzahl der versendeten Bytes, die von der Gegenstelle noch nicht bestätigt wurden.

Local Address Lokale Adresse und Portnummer des Sockets. Ein Stern (»*«) an dieser Stelle steht bei »lauschenden« Sockets dafür, dass sie auf allen verfügbaren Adressen lauschen, also zum Beispiel 127.0.0.1 *und* der IP-Adresse der Ethernet-Karte.

Foreign Address Die Adresse und Portnummer der Gegenstelle auf dem entfernten Rechner.

State Zustand des Sockets. raw-Sockets haben keine Zustände und udp-Sockets normalerweise auch nicht, in welchem Fall dieses Feld leer bleibt. Ansonsten, insbesondere für tcp-Sockets, sind u. a. folgende Einträge für den Zustand möglich:

- **ESTABLISHED** Eine Verbindung ist etabliert.
- **SYN_SENT** Das Socket versucht, eine Verbindung aufzubauen, und hat das erste Paket des Drei-Wege-Handshakes verschickt, aber noch keine Antwort bekommen.
- **SYN_RECV** Das Socket (ein »lauschendes«) hat eine Verbindungsanfrage bekommen und bestätigt.
- **FIN_WAIT1** Das Socket ist geschlossen, die Verbindung wird gerade abgebaut.
- **FIN_WAIT2** Die Verbindung ist abgebrochen und das Socket wartet auf die Beendigung durch die Gegenstelle.
- **TIME_WAIT** Nach Abbau der Verbindung wartet das Socket darauf, noch im Netzwerk verbliebene Pakete zu bearbeiten.
- **CLOSE** Das Socket wird nicht verwendet.
- **CLOSE_WAIT** Die Gegenstelle hat sich abgemeldet und wartet darauf, dass das Socket geschlossen wird.
- **LISTEN** Das Socket »lauscht« auf eingehende Verbindungen. Solche Sockets werden nur angezeigt, wenn Sie die Optionen -l oder -a angegeben haben.


Ohne `-t` und `-u` liefert `netstat` außer Informationen über TCP- und UDP-Dienste noch Daten über aktive Unix-Domain-Sockets. Diese sind allerdings größtenteils uninteressant.


Wenn Sie die Option `-l` weglassen, bekommen Sie statt dessen eine Liste der aktiven Netzwerkverbindungen (sowohl diejenigen, bei denen Ihr Rechner als Server fungiert, als auch diejenigen, bei denen er als Client auftritt).

Wenn Ihr Dienst in der Ausgabe von »`netstat -tul`« nicht auftaucht, ist das ein Indiz dafür, dass das betreffende Programm nicht läuft. Steht der Dienst in der Liste, könnte es einerseits sein, dass Clients durch eine Firewall-Konfiguration abgewiesen werden, noch bevor sie ihn überhaupt erreichen. Andererseits wäre es möglich, dass der betreffende Port durch ein anderes Programm blockiert wird, das aus irgendwelchen Gründen nicht richtig funktioniert. In diesem Fall können Sie sich mit »`netstat -tulp`« auch noch die Prozess-ID und den Namen des Programms anzeigen lassen, das den Port bedient. Dazu müssen Sie allerdings `root`-Privilegien haben.

`netstat` setzt voraus, dass Sie zumindest Shell-Zugang, wenn nicht gar `root`-Rechte auf dem Rechner haben, wo Sie das Kommando ausführen wollen. Wie sieht es aber damit aus, »von außen« zu testen, welche Ports auf einem Rechner zugänglich sind? Auch dafür gibt es Lösungen. Das Programm `nmap` ist ein **Portscanner**, der über das Netz prüft, welche TCP- oder UDP-Ports auf einem Rechner offen sind, welche gesperrt und welche unbenutzt. Natürlich kann der »Rechner« auch eine Firewall-Infrastruktur sein, so dass `nmap` Ihnen dabei helfen kann, Lücken in Ihrer Sicherheitsstrategie aufzudecken.

Portscanner


`nmap` ist nicht automatisch Bestandteil einer Linux-Installation. Sie müssen es wahrscheinlich manuell nachinstallieren.


Das Scannen von Rechnern, die sich nicht in Ihrem unmittelbaren juristischen Einflußgebiet befinden, kann strafbar sein! (Tatsächlich könnte man Ihnen vermutlich schon aus dem *Besitz* von »Hacker-Werkzeugen« wie `nmap` einen Strick drehen, wenn Sie Pech haben und/oder sich ungeschickt anstellen.) Beschränken Sie sich also auf Rechner, wo definitiv klar ist, dass Sie `nmap` anwenden dürfen. Lassen Sie es sich zur Sicherheit schriftlich von Ihrem Auftraggeber oder hinreichend hohen Vorgesetzten erlauben.

Im einfachsten Fall übergeben Sie `nmap` den Namen oder die Adresse des zu untersuchenden Rechners (seien Sie vorbereitet auf eine gewisse Wartezeit):

```
# nmap blue.example.com

Starting Nmap 4.68 ( http://nmap.org ) at 2009-02-04 00:09 CET
Interesting ports on blue.example.com (172.16.79.2):
Not shown: 1710 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
443/tcp  open  https
MAC Address: 00:50:56:FE:05:04 (VMWare)

Nmap done: 1 IP address (1 host up) scanned in 9.751 seconds
```

`nmap` betrachtet Ports als *open*, wenn sich dort ein Dienst meldet. Ports, für die der untersuchte Rechner eine Fehlermeldung ausgibt, gelten als *closed*, während Ports, auf denen überhaupt keine Reaktion festzustellen ist (etwa weil die Anfragepakete vom untersuchten Rechner oder einem Firewall einfach weggeworfen und nicht einmal mit einer Fehlermeldung beantwortet werden), als *filtered* bezeichnet werden.


Wenn Sie nichts anderes angeben, analysiert nmap die TCP-Ports des Zielrechners mit einem »SYN Scan«. Dabei schickt das Programm an jeden der betreffenden Ports ein TCP-Segment, das das SYN-Flag gesetzt hat (so als ob es eine neue Verbindung aufbauen wollte). Antwortet der Zielrechner mit einem TCP-Segment, das die SYN- und ACK-Flags gesetzt hat, so nimmt nmap an, dass der Port benutzt wird. Es unternimmt aber weiter nichts (insbesondere bestätigt es dieses Segment nicht), so dass die »halboffene« Verbindung nach Ablauf der vorgeschriebenen Fristen vom Zielrechner verworfen wird. Antwortet der Zielrechner statt dessen mit einem Segment, das das RST-Flag gesetzt hat, ist der Port *closed*. Kommt nach einigen Versuchen immer noch keine Antwort oder nur ICMP-Unerreichbarkeitsmeldungen, dann wird der Port auf *filtered* gesetzt. – Für SYN Scans sind root-Rechte erforderlich.


Andere Techniken, die nmap anbietet, sind der »TCP Connect Scan« (der keine besonderen Privilegien benötigt, aber plump und von der Gegenseite leicht zu erkennen ist), der »UDP Scan« und verschiedene andere Varianten von TCP-basierten Scans, etwa um die Regelsätze von Firewalls auszuspähen. Konsultieren Sie die Dokumentation in nmap(1).


nmap kann nicht nur bestimmen, welche Ports auf einem Rechner aktiv sind, sondern in vielen Fällen sogar sagen, welche Software sich dahinter verbirgt. Dazu müssen Sie die Option -A angeben und *richtig* Geduld mitbringen. nmap verläßt sich dafür auf eine mitgelieferte Datenbank von »Signaturen« verschiedener Programme.


Die Möglichkeiten von nmap übersteigen bei weitem das, was wir in dieser Schulungsunterlage erklären können. Lesen Sie die Dokumentation (in nmap(1)) und denken Sie beim Testen immer an die oben gemachte juristische Einschränkung.

5.6 DNS testen mit host und dig

Wenn Verbindungen zu namentlich angesprochenen Rechnern ungebührlich lange dauern oder nach einer Wartezeit gar nicht zustande kommen, es über die IP-Adresse aber üblich flott geht, dann ist vermutlich das DNS schuld. Ebenso kann es sein, dass Ihr Rechner lange braucht, um eine Verbindung aufzubauen, weil die Gegenstelle versucht, einen zu *Ihrer* IP-Adresse passenden Namen zu finden und das aus irgendwelchen Gründen schief geht. Um das DNS zu testen, können Sie zum Beispiel die Programme host und dig verwenden.


»Und was ist mit nslookup?« hören wir Sie sagen. Sorry, aber nslookup ist seit einer Weile verpönt und wird nur noch aus Freundlichkeit unterstützt.

host ist ein sehr einfaches Programm, das im simpelsten Fall einen DNS-Namen entgegennimmt und die dazugehörige(n) IP-Adresse(n) ausgibt:

```
$ host www.linupfront.de
www.linupfront.de is an alias for s0a.linupfront.de.
s0a.linupfront.de has address 31.24.175.68
```

Es funktioniert auch umgekehrt:

```
$ host 193.99.144.85
85.144.99.193.in-addr.arpa domain name pointer www.heise.de
```

(Fragen Sie nicht.)

Sie können die Ausgabe mehrerer DNS-Server miteinander vergleichen, indem Sie die IP-Adresse (oder den Namen, aber die Adresse ist sicherer) eines DNS-Servers bei der Anfrage mit angeben:

```
$ host www.linupfront.de 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

www.linupfront.de is an alias for s0a.linupfront.de.
s0a.linupfront.de has address 31.24.175.68
```

Auf diese Weise können Sie prüfen, ob ein DNS-Server die richtigen Antworten liefert.

Sie können bestimmte Arten von DNS-Datensätzen anfordern, indem Sie die Option `-t` verwenden, etwa wie

```
$ host -t mx linupfront.de                                    MX-Datensatz gewünscht
linupfront.de mail is handled by 10 s0a.linupfront.de
```

Mit `-l` bekommen Sie eine Liste der wichtigsten Namen in einer Domain – jedenfalls wenn Sie dürfen. Zusammen mit der Option `-a` können Sie eine Liste *aller* Namen bekommen.

Das Programm `dig` tut im wesentlichen dasselbe wie `host`, aber erlaubt eine detailliertere Diagnose von Problemen. Es liefert eine ausführlichere Ausgabe als `host`:

```
$ dig www.linupfront.de

; <<>> DiG 9.9.5-10-Debian <<>> www.linupfront.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1443
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.linupfront.de.             IN      A

;; ANSWER SECTION:
www.linupfront.de.      3600    IN      CNAME   s0a.linupfront.de.
s0a.linupfront.de.      3600    IN      A       31.24.175.68

;; Query time: 51 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul 22 18:00:34 CEST 2015
;; MSG SIZE  rcvd: 69
```

Zur Rückwärtsauflösung von IP-Adressen in Namen müssen Sie die Option `-x` angeben:

```
$ dig -x 31.24.175.68

; <<>> DiG 9.9.5-10-Debian <<>> -x 31.24.175.68
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63823
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
```

```
;68.175.24.31.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
68.175.24.31.in-addr.arpa. 86400 IN     PTR     s0a.linupfront.de.

;; Query time: 50 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul 22 18:01:31 CEST 2015
;; MSG SIZE  rcvd: 74
```

Um einen bestimmten DNS-Server zu befragen, geben Sie dessen Adresse hinter einem `@` an:

```
$ dig www.linupfront.de @192.168.20.254
```

Einen DNS-Datensatztyp können Sie einfach hinter dem gesuchten Namen angeben:

```
$ dig linupfront.de mx

; <<>> DiG 9.9.5-10-Debian <<>> linupfront.de mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15641
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;linupfront.de.                 IN      MX

;; ANSWER SECTION:
linupfront.de.          3600    IN      MX     10 s0a.linupfront.de.

;; Query time: 49 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul 22 17:59:36 CEST 2015
;; MSG SIZE  rcvd: 51
```

Prinzipiell können Sie auch das Kommando `getent` verwenden, um die Namensauflösung zu testen: `getent`

```
$ getent hosts www.linupfront.de
31.24.175.68    s0a.linupfront.de www.linupfront.de
```

Der Unterschied zwischen `host` und `dig` auf der einen und `getent` auf der anderen Seite ist, dass die ersteren das DNS direkt befragen. Das letztere Kommando befragt dagegen die C-Bibliothek. Das bedeutet einerseits, dass die Abfragereihenfolge eingehalten wird, die in `/etc/nsswitch.conf` vorgegeben wurde. Andererseits bekommen Sie die Antwort in der Form, wie sie auch in `/etc/hosts` auftauchen würde.

In `/etc/nsswitch.conf` steht für gewöhnlich etwas wie

```
hosts: files dns
```

Das heißt, es wird erst in `/etc/hosts` nachgeschaut und dann im DNS. Der Vorteil ist, dass Sie auf diese Weise genau das sehen, was Anwendungsprogramme sehen, die die C-Bibliothek verwenden. Es könnte zum Beispiel sein, dass aus irgendwelchen Gründen für einen Namen eine Definition in `/etc/hosts` existiert, die dann Vorrang gegenüber dem DNS hat (weil nach einem `/etc/hosts`-Treffer das DNS nicht mehr konsultiert wird).

Von anderen Anwendungen von `getent` sind Sie es vielleicht gewöhnt, dass ein einfaches

```
$ getent passwd
```

Ihnen eine Liste aller dem System bekannten Benutzer im Format von `/etc/passwd` ausspuckt, selbst wenn die Benutzer gar nicht alle in der lokalen Kennwortdatei stehen. Für Benutzer kann das klappen, aber muss nicht (wenn Sie in einem großen Unternehmen arbeiten, haben die Administratoren Ihrer Benutzerdatenbank dem möglicherweise einen Riegel vorgeschoben). Für das DNS führt ein

```
$ getent hosts
```

aber *definitiv* nicht dazu, dass alle weltweit im DNS definierten Namen aufgelistet werden. (Ist wahrscheinlich auch besser so.)

DNS ist ein sehr vielschichtiges Thema, bei dem auch vieles schief gehen kann. Allerdings erfordert die detaillierte Diagnose von DNS-Problemen einiges an Vorkenntnissen. Das Thema DNS behandelt im Detail die Linup-Front-Schulungsunterlage *Das Domain Name System*.

5.7 Andere nützliche Diagnosewerkzeuge

5.7.1 `telnet` und `netcat`

Das `telnet`-Kommando wird verwendet, um sich über das TELNET-Protokoll interaktiv auf einem anderen Rechner anzumelden oder – allgemein – mit einem TCP-Port Kontakt aufzunehmen. Als Dienst für den Fernzugriff sollte TELNET nicht mehr verwendet werden, da keine starke Authentisierung verwendet wird und die Datenübertragung unverschlüsselt stattfindet. Die Secure Shell (`ssh`, Kapitel 10) ist eine sinnvolle Alternative.

Das Client-Programm `telnet` ist aber hervorragend zum Testen vieler anderer Dienste geeignet. Mit »`telnet` ⟨*Adresse*⟩ ⟨*Dienst*⟩« kann eine Verbindung zu jedem beliebigen Port aufgebeut werden (»⟨*Dienst*⟩« ist entweder eine Portnummer oder ein Dienstname aus `/etc/services`). So baut »`telnet 192.168.0.100 80`« eine Verbindung zu einem Web-Server auf. In diesem Fall wäre es sogar möglich, mit den entsprechenden HTTP-Befehlen Dokumente vom Server aufzurufen. Ein anderes Beispiel:

```
$ telnet 192.168.0.1 22
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_6.7p1 Debian-6
```

Im Beispiel wird eine Verbindung zum SSH-Port auf einem anderen Rechner aufgebaut, der entfernte `sshd` meldet sich mit Protokoll- und Programm-Version.

Mit dem *escape character* können Sie eine »Auszeit« von der TCP-Verbindung nehmen und `telnet`-Kommandos eingeben. Die interessantesten Kommandos sind wahrscheinlich `close` (bricht die Verbindung ab), `status` (zeigt den Verbindungsstatus an) und `!` (kann verwendet werden, um auf dem lokalen Rechner bei laufender Verbindung Kommandos auszuführen):

```
$ telnet 192.168.0.1 22
Trying 192.168.0.1...
Connected to 192.168.0.1.
```

```
Escape character is ']^.
SSH-2.0-OpenSSH_6.7p1 Debian-6
Strg+Esc
telnet> status
Connected to 192.168.0.1.
Operating in obsolete linemode
Local character echo
Escape character is ']^.
_
```

Möglicherweise ist bei Ihrem `telnet` das »!«-Kommando herauskonfiguriert. Dann können Sie immer noch mit dem Kommando `z` das `telnet`-Programm in den Hintergrund schicken (denken Sie »Jobkontrolle der Shell«) und später mit `fg` wieder reaktivieren.

Eine Alternative zum TELNET-Client `telnet` ist das Programm `netcat`. Im einfachsten Fall benimmt `netcat` sich wie `telnet` (ist aber nicht ganz so geschwätzig):

```
$ netcat 192.168.0.1 22
SSH-2.0-OpenSSH_6.7p1 Debian-6
```

Oft heißt das Kommando statt (oder zusätzlich zu) `netcat` auch `nc`. Der Rest bleibt allerdings derselbe.

Von `netcat` sind zwei Versionen im Umlauf, eine »traditionelle« (von einem Menschen namens »Hobbit«) und eine aus dem OpenBSD-System. Letztere kann deutlich mehr (etwa IPv6 oder Unix-Domain-Sockets). Wir setzen für den Rest des Abschnitts das OpenBSD-`netcat` voraus.

Bei Debian GNU/Linux ist das Standard-`netcat` die traditionelle Version (aus dem Paket `netcat-traditional`). Wenn Sie die aufgebohrte Version haben möchten, müssen Sie das Paket `netcat-openbsd` installieren. Das OpenBSD-`netcat` installiert sich *nur* unter dem Namen `nc`; die traditionelle Version bleibt unter dem Namen `netcat` erhalten, solange Sie das Paket nicht deinstallieren.

Neben der Client-Seite einer TCP-Verbindung implementiert `netcat` auf Wunsch auch die Server-Seite (natürlich ohne direkt viel Nützliches zu tun). Zum Beispiel: Server-Seite
Mit

```
$ nc -l 4711
```

lauscht `netcat` auf dem Port 4711 auf Verbindungen. In einem anderen Fenster können Sie dann mit

```
$ nc localhost 4711
```

Verbindung zu Ihrem »Server« aufnehmen. Was Sie auf der Client-Seite eintippen, erscheint beim Server und umgekehrt. Dateitransfer »für Arme« geht so: Tippen Dateitransfer »für Arme«
Sie auf dem Zielrechner

```
$ nc -l 4711 >myfile
```

und auf dem Ausgangsrechner

```
$ nc red.example.com 4711 <myfile
```

5.7.2 tcpdump

Netzwerk-Sniffer

Bei `tcpdump` handelt es sich um einen Netzwerk-Sniffer, der die Analyse der über ein Netzwerkinterface laufenden Netzwerkpakete ermöglicht. Die Netzwerkkarte wird dabei in den sog. **Promiscuous Mode** versetzt, in dem alle Pakete, also nicht nur wie üblich die für das lokale Interface bestimmten, gelesen werden. Das Kommando kann deshalb nur als Benutzer `root` verwendet werden.

Promiscuous Mode

Ein Beispiel für die Verwendung:

```
# tcpdump -ni eth0
tcpdump: listening on eth0
14:26:37.292993 arp who-has 192.168.0.100 tell 192.168.0.1
14:26:37.293281 arp reply 192.168.0.100 is-at 00:A0:24:56:E3:75
14:26:37.293311 192.168.0.1.35993 > 192.168.0.100.21: S 140265170:
140265170(0) ...
14:26:37.293617 192.168.0.100.21 > 192.168.0.1.35993: S 135130228:
135130228(0) ack 140265171 ...
14:26:37.293722 192.168.0.1.35993 > 192.168.0.100.21: . ack 1 ...
                                                    Abbruch des Programms
5 packets received by filter
0 packets dropped by kernel
```

Im Beispiel wurde ein Verbindungsaufbau zu einem FTP-Server »belauscht«. Die Parameter »`-ni eth0`« schalten die Namensauflösung und die Auflösung der Portnummern ab und involvieren nur das Interface `eth0`. Zu jedem Paket werden die genaue Uhrzeit, Quell- und Zielrechner, gesetzte Flags im TCP-Header (S: `SYN`-Bit), die Folgenummer der Daten, ein eventuell gesetztes `ACK`-Bit, die erwartete Folgenummer des nächsten Pakets und diverse weitere Informationen angezeigt.

Das erste gezeigte Paket enthält keinen Zielrechner, es handelt es sich um eine ARP-Anfrage: Der Rechner `192.168.0.100` soll seine MAC-Adresse bekannt geben – was er im zweiten Paket dann auch tut. Die nächsten Pakete zeigen einen typischen TCP-Drei-Wege-Handshake.

5.7.3 wireshark

Bei `wireshark` handelt es sich bei wie bei `tcpdump` um einen Netzwerksniffer. Allerdings besitzt `wireshark` einen ungleich größeren Funktionsumfang. Es handelt sich um ein grafisches Programm, das die genaue Analyse aller Netzwerkpakete ermöglicht. Die Anzeige besteht aus drei Fenstern: Im oberen werden die eingelaufenen Pakete angezeigt, im unteren wird der Paketinhalt hexadezimal aufgeschlüsselt, und im mittleren Fenster können die Kopfinformationen (und Nutzdaten) jeder Protokollschicht bis aufs letzte Bit und ungemein komfortabel analysiert werden.

Ähnlich wie `nmap` ist `wireshark` kein Standard-Unix-Werkzeug und muss oft nachinstalliert werden. Sowohl `tcpdump` als auch `wireshark` sollten mit Bedacht angewandt werden, da auch im lokalen Netz schnell gegen geltendes Recht verstoßen werden kann. Schließlich werden eventuell Daten angezeigt, die einen nichts angehen.

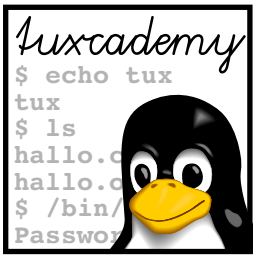
Das Programm `wireshark` hieß bis vor einigen Jahren `ethereal` und ist auf älteren Rechnern möglicherweise noch unter diesem Namen zu finden.

Kommandos in diesem Kapitel

dig Sucht Informationen im DNS (sehr komfortabel) dig(1) 96
getent Ruft Einträge aus administrativen Datenbanken ab getent(1) 97
host Sucht Informationen im DNS host(1) 95
netcat Sehr allgemeines Netzwerk-Client-Programm nc(8) 99
nmap Netzwerk-Portscanner, analysiert offene Ports auf Rechnern nmap(1) 94
ping Prüft grundlegende Netzwerk-Konnektivität mit ICMP ping(8) 87
ping6 Prüft grundlegende Netzwerk-Konnektivität (für IPv6) ping(8) 88
tcpdump Netzwerk-Sniffer, protokolliert und analysiert Netzwerkverkehr tcpdump(1) 99
telnet Erlaubt Verbindungen zu beliebigen TCP-Diensten, insbesondere TELNET (Fernzugriff) telnet(1) 98
tracepath Prüft die Wegleitung zu einer anderen Station, mit Pfad-MTU tracepath(8) 91
tracepath6 Entspricht tracepath, aber für IPv6 tracepath(8) 92
traceroute Prüft die Wegleitung zu einer anderen Station traceroute(8) 89
wireshark Paket-Sniffer, liest und analysiert Netzwerkverkehr (Ex-ethereal) wireshark(1) 100

Zusammenfassung

- Mit Programmen wie netstat, telnet, nmap, tcpdump oder wireshark stehen leistungsfähige Werkzeuge zur Diagnose von Fehlern bei Netzwerkdiensten zur Verfügung.



6

inetd und xinetd

Inhalt

6.1 Netzdienste anbieten mit inetd 104
6.1.1 Überblick. 104
6.1.2 Die Konfiguration des inetd 104
6.2 Der TCP-Wrapper tcpd 106
6.3 Der xinetd. 109
6.3.1 Überblick. 109
6.3.2 Die Konfiguration des xinetd 109
6.3.3 Starten des xinetd 111
6.3.4 Parallelverarbeitung von Anfragen 111
6.3.5 inetd durch xinetd ersetzen 112

Lernziele

- Wissen, wie Dienste mit inetd und xinetd gestartet werden können
- Zugriffskontrolle mit dem TCP-Wrapper und dem xinetd einsetzen können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 3)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)

6.1 Netzdienste anbieten mit inetd

6.1.1 Überblick

Ein Linux-System kann als Server in einem Netzwerk eine Fülle von Diensten zur Verfügung stellen – TELNET, FTP, POP3, IMAP, … Jeder dieser Dienste wird dabei über einen bestimmten TCP- oder UDP-Port angesprochen. Es gibt im Grunde zwei Möglichkeiten, einen solchen Dienst zu erbringen: Zum einen kann ein spezialisierter Prozess (ein Daemon) auf Verbindungen auf dem betreffenden Port lauschen. Ein Web-Server zum Beispiel nimmt auf dem TCP-Port 80 Anfragen entgegen und beantwortet sie, während ein DNS-Server den UDP-Port 53 bedient.

Zum anderen ist es möglich, das Lauschen an vielen Ports einem Programm zu übertragen, das bei einer eingehenden Verbindung auf irgendeinem dieser Ports ein anderes Programm startet, das den eigentlichen Dienst erbringt. Ein solches Programm ist der inetd (oder *Internet Daemon*).

Warum sollten Sie ein Programm wie den inetd verwenden wollen? Es gibt einige offensichtliche Vorteile:

- Viele Dienste werden nur selten gebraucht. Ein spezialisierter Daemon für so einen Dienst würde aber auch dann Ressourcen des Systems binden, wenn der Dienst gerade nicht in Anspruch genommen wird (und sei es Platz im Auslagerungsspeicher). Auf heutigen Rechnern ist das weniger ein Problem als früher, aber das Prinzip bleibt bestehen.

- Die Entwicklung einfacher Netzdienste wird radikal vereinfacht. Während Sie beträchtliche Fachkenntnisse brauchen, um freistehende Daemons zu schreiben, die alle Regeln einhalten und zum Beispiel nicht mit der Zeit den kompletten Arbeitsspeicher des Rechners an sich ziehen, können Sie sich bei Diensten, die über den inetd aufgerufen werden, darauf beschränken, von der Standardeingabe zu lesen und auf die Standardausgabe zu schreiben. Der inetd kümmert sich darum, dass die Daten, die der entfernte Client schickt, auf der Standardeingabe an das Dienstprogramm übergeben werden, und leitet dessen Standardausgabe an den Client weiter. Netzwerkprogrammierung im eigentlichen Sinne ist also überhaupt nicht erforderlich. Außerdem beendet Ihr Server sich nach der Sitzung wieder und gibt so alle Ressourcen wieder frei.

Der inetd bietet sich auch oder vor allem zur Implementierung von Diensten an, die längere »Sitzungen« involvieren, etwa FTP oder SMTP (auf weniger frequentierten Rechnern). Damit können die Kosten dafür, den Prozess zu starten, der den eigentlichen Dienst erbringt, über eine längere Zeit amortisiert werden. Es wäre hingegen eine äußerst unkluge Idee, zum Beispiel einen Web-Server über den inetd zu betreiben, da der Aufwand dafür, zur Bearbeitung einer einzelnen HTTP-Anfrage den kompletten HTTP-Server zu starten, in keinem Verhältnis zur tatsächlich für die eigentliche Anfrage zu erbringenden Arbeit steht.

6.1.2 Die Konfiguration des inetd

Die Konfigurationseinstellungen für den inetd stehen in der Datei /etc/inetd.conf. Jede Zeile der Konfigurationsdatei, die nicht leer oder eine Kommentarzeile ist, bezieht sich dabei auf einen Dienst. Eine Zeile kann beispielsweise so aussehen:

```
ftp  stream  tcp  nowait  root  /usr/sbin/ftpd ftpd
```

Dienstname

Das erste Wort der Zeile identifiziert den zur Verfügung zu stellenden Dienst namentlich oder über eine Portnummer; ein Dienstname muss einem Eintrag in der Datei /etc/services entsprechen, die Informationen darüber enthält, welcher Port dem Dienst zugeordnet ist. Der zweite Eintrag der Zeile gibt an, von welchem Typ der geöffnete Socket ist. Mögliche Werte sind stream (wie hier), dgram, raw, rdm und

Sockettyp

seqpacket. In der Praxis werden Ihnen vermutlich nur die Werte stream und dgram begegnen.

Protokoll

Als nächstes folgt das zum Zugriff auf den Dienst verwendete Protokoll. Protokollnamen müssen in der Datei /etc/protocols definiert sein. Typische Werte sind hier tcp oder udp, wobei der Eintrag stream in der vorigen Spalte den Eintrag tcp mehr oder weniger erzwingt. Dasselbe gilt für dgram und udp.

Parallelität

Im vierten Feld steht entweder wait oder nowait. Dieser Eintrag steuert den Umgang mit Sockets vom Typ dgram (vulgo UDP); für andere Sockettypen sollte nowait angegeben werden. wait bedeutet, dass bei einem Zugriff auf einen Dienst der zur Kommunikation benötigte Port so lange belegt wird, bis die eingehende Anfrage komplett abgearbeitet wurde. Erst danach können neue Anfragen für den entsprechenden Dienst angenommen werden. nowait besagt, dass der Port sofort wieder freigegeben wird, so dass umgehend weitere Anfragen entgegen genommen werden können.

Statt nur nowait können Sie auch nowait.*n* schreiben. Die Ganzzahl *n* gibt dabei die Maximalzahl von Serverprozessen an, die der inetd in 60 Sekunden erzeugt. Wird .*n* weggelassen, gilt der Standardwert 40.

Benutzername
Kommando zum Start des Dienstes

Der nächste Eintrag bezeichnet den Benutzernamen, unter dessen Identität der Dienst ausgeführt wird, während der Rest der Zeile das Kommando zum Start des Dienstes samt etwaiger Optionen angibt. Dabei ist das erste »Wort« der Name der auszuführenden Programmdatei und das zweite eine Zeichenkette, die dem betreffenden Prozess als »Programmname« übergeben werden soll. Erst dann folgen die üblichen Kommandoparameter. Im Beispiel oben ist das Wort ftpd also nicht der erste Parameter des Kommandos /usr/sbin/ftpd, sondern der übergebene Name für den Prozess, dessen Code aus /usr/sbin/ftpd stammt! Mit anderen Worten: Die Programmdatei /usr/sbin/ftpd wird ohne Parameter ausgeführt, aber der resultierende FTP-Server-Prozess denkt, er hieße statt /usr/sbin/ftpd einfach nur ftpd.

Dienste aufnehmen

Es ist einfach, Dienste in die inetd-Konfiguration aufzunehmen oder daraus zu entfernen – Sie müssen nur eine geeignete Zeile für die inetd.conf-Datei entwerfen.

Dienste still legen

Existierende Dienste lassen sich leicht still legen, indem Sie die betreffenden Konfigurationszeilen mit einem »#«-Zeichen in der ersten Spalte auskommentieren.

Konfiguration neu einlesen

Nach allen Änderungen der inetd-Konfigurationsdatei muss der inetd durch ein SIGHUP dazu gebracht werden, die Konfiguration neu einzulesen. Bei den meisten Distributionen geht das bequem über das Init-Skript für den inetd mit dem Parameter reload.

Übungen

6.1 [!1] Schalten Sie in der Datei inetd.conf den echo-Dienst frei und laden Sie die Konfiguration neu. Prüfen Sie die Funktion des Dienstes mit dem Kommando »telnet localhost echo«.

6.2 [2] Warum eignet ein FTP-Server sich besser dafür, über den inetd aufgerufen zu werden, als ein WWW-Server?

6.3 [3] (Für Programmierer.) Schreiben und installieren Sie einen Dienst, der die Geheimschrift von Julius Cäsar implementiert: Jeder Buchstabe wird durch den Buchstaben ersetzt, der im Alphabet 3 Positionen weiter steht, also A durch D, B durch E und so weiter. Ersetzen Sie X durch A, Y durch B und Z durch C:

```
$ telnet localhost caesar
Trying 127.0.0.1...
Connected to linux.example.com.
Escape character is '^]'.
GALLIA OMNIS DIVISA EST IN PARTES TRES
```

```
JDOOLD RPQLV GLYLVD HVW LQ SDUWHV WUHV
Strg+]
telnet> close
Connection closed.
```

Sie können sich der Einfachheit halber bei der Verschlüsselung auf die 26 Großbuchstaben beschränken und alle anderen Zeichen verbatim durchlassen. Testen Sie die Lösung mit `telnet`.

6.2 Der TCP-Wrapper `tcpd`

Ein Problem stellen eventuelle Versuche Unberechtigter dar, einen Dienst zu verwenden. Jeder Dienst müsste für sich selbst prüfen, welche Anforderungen er als berechtigt ansieht und welche er abweisen würde. Da eine Vielzahl heute verfügbarer Dienste eine solche Zugriffssteuerung nicht besitzt, wurde eine zentrale Instanz mit der Aufgabe geschaffen, diese Funktion allen Diensten zur Verfügung zu stellen. Zu diesem Zweck dient der sogenannte TCP-Wrapper (engl. *to wrap* = umschliessen, umwickeln) namens `tcpd`. Erfolgt ein Zugriffsversuch auf einen Dienst, startet der `inetd` anstatt des angesprochenen Dienstes zuerst den TCP-Wrapper. Dieser protokolliert zunächst den Zugriffsversuch mit Hilfe des `syslog`-Dienstes. Anschliessend prüft er anhand der Dateien `/etc/hosts.allow` und `/etc/hosts.deny`, ob der zugreifende Rechner berechtigt ist, den entsprechenden Dienst zu benutzen.

Zugriffssteuerung

`/etc/hosts.allow`

`/etc/hosts.deny`

Zunächst schaut der `tcpd` in der Datei `/etc/hosts.allow` nach einem Eintrag, der den aktuellen Zugriff explizit erlaubt. Gibt es einen solchen, wird der Zugriff gestattet. Andernfalls durchsucht er die Datei `/etc/hosts.deny` nach einem Eintrag, der den entsprechenden Zugriff verbietet. Ist ein solcher vorhanden, wird der Zugriff verweigert. Ist dies jedoch nicht der Fall, wird der Zugriff letztendlich doch gestattet. Sind `/etc/hosts.allow` oder `/etc/hosts.deny` gar nicht vorhanden, gelten sie als leer.

Die übliche und wahrscheinlich »sicherere« Philosophie wäre, jeden Zugriff grundsätzlich zu verweigern, der nicht ausdrücklich erlaubt wird. Dies verträgt sich jedoch nicht mit dem Gedanken, dass ein unkonfigurierter TCP-Wrapper sich so benehmen sollte, als wäre er gar nicht da.

Die Dateien `/etc/hosts.allow` und `/etc/hosts.deny` sind weitgehend ähnlich aufgebaut. Prinzipiell bestehen Einträge in diesen Dateien aus durch Doppelpunkte getrennten Feldern und sehen ungefähr so aus:

```
pop3d : 192.168.10.0/24
```

Daemon-Name

Im ersten Feld steht der Name eines zu startenden Daemons (das zweite Feld aus dem Kommando in `/etc/inetd.conf`) oder eine Reihe von durch Komma getrennten Daemon-Namen.

Soll der Eintrag für alle Programme gelten, schreiben Sie das Schlüsselwort `ALL`; soll er für alle außer ein paar Programmen gelten, schreiben Sie »`ALL EXCEPT …`« (wieder mit einer durch Komma getrennten Daemon-Namensliste).

Das zweite Feld gibt an, für welche Station(en) der Eintrag gelten soll. Im einfachsten Fall geben Sie die Station über ihren Namen oder ihre IP-Adresse an. Auch hier kann `ALL` für alle denkbaren Rechner stehen. Weiterhin existieren die Schlüsselwörter `KNOWN` (für Stationen, deren Namen der `tcpd` aus der IP-Adresse ermitteln kann), `LOCAL` (alle Stationen, deren Namen keinen Punkt enthalten), `UNKNOWN` (alle Stationen, deren Namen der `tcpd` aus der IP-Adresse *nicht* ermitteln kann)

Tabelle 6.1: Textersetzungen in Kommando-Einträgen bei /etc/hosts.allow und /etc/hosts.deny

Schlüssel	Bedeutung
%a	Die IP-Adresse der zugreifenden Station
%c	»Client-Information«: So viel Information, wie die entfernte Station hergibt, etwa in der Form ⟨*Benutzer*⟩@⟨*Stationsname*⟩, ⟨*Benutzer*⟩@⟨*Adresse*⟩, ⟨*Stationsname*⟩ oder ⟨*Adresse*⟩
%d	Der Name des gewünschten Dienstes
%h	Der Name der zugreifenden Station (oder deren IP-Adresse, wenn der Name nicht bestimmt werden kann)
%n	Der Name der zugreifenden Station (oder unknown oder paranoid, wenn der richtige Name nicht bestimmt werden kann)
%p	Prozessnummer des gestarteten Dienstes
%s	»Server-Information«: ⟨*Dienst*⟩@⟨*Stationsname*⟩, ⟨*Dienst*⟩@⟨*Adresse*⟩ oder einfach nur ⟨*Dienst*⟩
%u	Der Name des Benutzers auf der Clientseite oder unknown
%%	Ein einzelnes Prozentzeichen

und PARANOID (alle Stationen, deren Namens- und Adressauflösung über DNS widersprüchliche Angaben ergibt). Ganze IP-Netze können über Netzadresse und Netzmaske angegeben werden.

Die Zeile im Beispiel erlaubt oder verbietet also den Stationen im Netz 192.168.10.0/24 den Zugriff auf den POP3-Daemon, je nachdem, ob sie in /etc/hosts.allow oder /etc/hosts.deny steht.

Optionen

Nach der Stationsangabe können, wiederum durch Doppelpunkte getrennt, weitere Felder folgen, die Optionen für die Bearbeitung der Verbindung angeben, etwa so:

```
ALL: ALL: spawn echo "Zugriff von %u@%h auf %d" >>/var/log/net.log
```

Mit spawn spezifizieren Sie ein Shellkommando, das in einem Kindprozess ausgeführt wird. Die Standardein-, -aus- und -fehlerausgabe des Kommandos werden dabei mit /dev/null verbunden, um der Ein- und Ausgabe des eigentlich (in /etc/inetd.conf) angegebenen Kommandos nicht in die Quere zu kommen. Vorher werden noch »%«-Ausdrücke im Kommandotext gemäß dem Verbindungsversuch ersetzt; die möglichen Ersetzungen zeigt Tabelle 6.1. In unserem Beispiel werden Informationen über den Zugriffsversuch in der Datei /var/log/net.log protokolliert.

Hier sind noch ein paar andere Optionen für Zugriffsregeln:

twist (gefolgt von einem Shell-Kommando) ersetzt den aktuellen Prozess durch das Kommando (wieder nachdem die »%«-Ausdrücke ersetzt wurden), wobei die Standardein-, -aus- und -fehlerausgabe mit dem entfernten Client verbunden sind. Damit können Sie eine Verbindung sozusagen »übernehmen«:

```
in.ftpd : 10.0.0.0/8 : twist /bin/echo 421 Go away.
```

weist eingehende FTP-Verbindungswünsche aus dem 10.0.0.0/8-Netz mit der angegebenen Meldung ab, ohne dafür den FTP-Server bemühen zu müssen. Diese Option muss am Ende eines Eintrags stehen.

allow und deny akzeptieren oder verweigern einen Verbindungswunsch, egal in welcher der Konfigurationsdateien sie stehen. Das macht es möglich, die komplette Konfiguration zum Beispiel in /etc/hosts.allow zu halten, statt sie auf beide Dateien zu verteilen. Diese Optionen müssen am Ende eines Eintrags stehen.

umask entspricht dem umask-Kommando der Shell.

setenv setzt eine Umgebungsvariable (inklusive »%«-Ersetzung), etwa wie in

```
in.ftpd : 10.0.0.0/8 : setenv HOME /tmp
```

Beachten Sie aber, dass viele Daemons ihre Umgebung aufräumen und Einträge entfernen, die ihnen eigenartig vorkommen.

user setzt den Benutzer oder den Benutzer und die Gruppe für den Prozess:

```
user nobody                    Benutzer nobody
user nobody.nogroup            Benutzer nobody, Gruppe nogroup
```

Dies ist nützlich, weil inetd sonst alle Daemons als root ausführt.

banners (gefolgt von einem Verzeichnisnamen) (Nur bei TCP-Diensten.) Prüft, ob im angegebenen Verzeichnis eine Datei steht, die so heißt wie der angegebene Daemon-Prozess, und kopiert gegebenenfalls deren Inhalt an den Client. Dabei werden »%«-Ausdrücke ersetzt.

Tatsächlich ist das die »erweiterte« Kommandosprache für den TCP-Wrapper gemäß host_options(5). Die Standardversion (die Sie unter Linux aber nicht antreffen werden) beschränkt sich darauf, als drittes Feld eines Eintrags ein Shellkommando zuzulassen. Sie ist in host_access(5) dokumentiert.

Da der inetd in Zusammenarbeit mit dem tcpd bei einen Zugriff nicht den eigentlichen Dienst startet, sondern lediglich den Wrapper mit einem entsprechenden Argument aufruft, sehen in diesem Fall Einträge in der Datei /etc/inetd.conf etwas anders aus.

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

Das heißt, aufgerufen wird das Programm /usr/sbin/tcpd (der TCP-Wrapper), das aber als Kommandozeile »ftpd -l -a« übergeben bekommt. Hierbei ist ftpd der Kommandoname aus der Sicht des tcpd, den dieser benutzt, um das wirklich zu startende Kommando – den FTP-Server – und allfällige Einträge in den Dateien /etc/hosts.allow und /etc/hosts.deny zu finden.

Viele Programme sind heutzutage direkt gegen die libwrap gelinkt, den Teil des tcpd, der die eigentliche Arbeit macht. Diese Programme müssen nicht explizit über inetd und tcpd aufgerufen werden, um in den Genuss der Zugriffsbeschränkungen zu kommen.

Übungen

6.4 [!2] Über die Konfigurationszeile

```
ps  stream  tcp  nowait  root  /bin/ps ps auxw
```

wird ein Dienst definiert, der beim Zugriff auf den ps-Port (frei erfunden) eine Prozessliste liefert. Verwenden Sie den TCP-Wrapper, um den Zugriff auf diesen Dienst auf den lokalen Rechner zu beschränken.

6.5 [2] Sorgen Sie dafür, dass beim Zugriff auf den Dienst aus Übung 6.4 eine Nachricht ins syslog-Protokoll geschrieben wird, etwa mit der Kategorie local0 und der Priorität info.

6.3 Der xinetd

6.3.1 Überblick

Einige moderne Distributionen bringen einen möglichen Ersatz für die Kombination aus inetd und tcpd mit, den xinetd (***extended Internet daemon***). Der xinetd vereint alle Funktionen – Portüberwachung, Zugriffskontrolle und Protokollierung – unter einem Dach und somit auch in einer einzigen, zentralen Konfigurationsdatei.

extended Internet daemon

6.3.2 Die Konfiguration des xinetd

Die Konfigurationsdatei des xinetd heißt meistens /etc/xinetd.conf. In dieser Datei werden Leerzeilen oder Zeilen, die mit einem »#« beginnen, ignoriert. Die Konfigurationseinstellungen werden in Abschnitte unterteilt. Jede von ihnen beginnt mit einem Schlüsselwort, die dem Namen eines Dienstes in /etc/services entspricht und enthält Zuweisungen von Werten an Attribute, etwa wie folgt:

/etc/xinetd.conf

Abschnitte

```
default
    {
        ⟨Attribut⟩ ⟨Operator⟩ ⟨Parameter⟩ [<Parameter> …]
    }

service ⟨Dienstname⟩
    {
        ⟨Attribut⟩ ⟨Operatoren⟩ ⟨Parameter⟩ [⟨Parameter⟩ …]
    }
```

Als Operator kommt meist lediglich »=« in Frage, der dem Attribut eine genau umrissene Menge an Werten zuweist. Attribute, denen mehrere Werte zugewiesen werden können, unterstützen auch die Operatoren »+=« und »-=« zum Zuweisen oder Entfernen eines Wertes. Der Abschnitt »default« enthält Grundeinstellungen, die für alle Dienste gültig sind, solange diese keine abweichenden Werte enthalten. Tun sie dies, so werden die Grundeinstellungen entweder durch die spezielleren ersetzt oder mit ihnen kombiniert. In jedem weiteren Abschnitt werden genauere Angaben zu einem speziellen Dienst gemacht. Tabelle 6.2 zeigt einige wesentliche Attribute. – Die exakte Syntax sowie weitere Attribute stehen in der Anleitung zur Datei xinetd.conf. Eine /etc/xinetd.conf-Datei könnte beispielsweise wie folgt aussehen:

Grundeinstellungen

Spezielle Einstellungen

```
defaults
{
    log_type        = FILE /var/log/xinetd.log
    log_on_success  = HOST EXIT DURATION
    log_on_failure  = HOST ATTEMPT RECORD
    instances       = 2
}

service telnet
{
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = root
    server          = /usr/sbin/in.telnetd
    server_args     = -n
    only_from       = localhost
    no_access       =
}
```

Tabelle 6.2: Attribute in der Datei /etc/xinetd.conf

Attribut	Bedeutung
type	Erlaubt Angaben wie INTERNAL, d. h. der Dienst wird direkt durch den xinetd zur Verfügung gestellt, oder UNLISTED, d. h. der Dienst ist nicht in /etc/services verzeichnet.
socket_type	erlaubt Werte wie stream, dgram oder raw.
protocol	Das vom Dienst verwendete Protokoll, muss in /etc/protocols vorhanden sein.
wait	Bei yes handelt es sich um einen *single-threaded*-Dienst; no erlaubt dem xinetd, den Dienst mehrmals gleichzeitig zu starten.
user	Der Benutzer, mit dessen Rechten der Dienst laufen soll; dieser muss ein gültiger Systembenutzer sein.
instances	Anzahl der maximal gleichzeitig erlaubten Instanzen des Dienstes (bei »wait = no«)
server	Der Pfad zum eigentlichen Serverprogramm
server_args	Startparameter für das Serverprogramm
interface	IP-Adresse des Netzwerkinterfaces, an dem der xinetd auf Verbindungsanfragen wartet
only_from	Nur die angegebenen Clients (als DNS-Name oder IP-Adresse, ggf. mit Netzmaske) dürfen zugreifen
no_access	Die angegebenen Clients dürfen nicht zugreifen.
access_times	Der Dienst ist nur zu den angegebenen Zeiten verfügbar.
log_type	Legt die Protokollierungsart des xinetd fest – SYSLOG oder FILE
log_on_success	Definiert, welche Informationen bei erfolgreicher Verbindungsaufnahme protokolliert werden sollen, etwa HOST (der Clientrechner), USERID (der zugreifende Benutzer gemäß [RFC1413] usw.
log_on_failure	Definiert, welche Informationen bei gescheiterter Verbindungsaufnahme protokolliert werden sollen, etwa ATTEMPT (der fehlgeschlagene Zugriff) usw.
disable	Deaktiviert den Dienst (entspricht dem Auskommentieren einer Zeile in /etc/inetd.conf)
disabled	Kann im defaults-Abschnitt gesetzt werden, um eine Reihe von Diensten zu deaktivieren (z. B. »disabled finger ftp«)

Tabelle 6.3: xinetd und Signale

Signal	Wirkung
SIGHUP	Löst eine »harte Neukonfiguration« aus: xinetd liest die Konfigurationsdatei neu ein und beendet die Server, die in der neuen Konfiguration nicht mehr freigeschaltet sind. Für die anderen Server wird die Zugriffskontrolle wiederholt, und alle Verbindungen, die diese Prüfung nicht überstehen, werden gekappt. Wenn mehr Verbindungen zu einem Dienst bestehen, als instances erlaubt, werden zufällig ausgewählte Server beendet, bis die Grenze wieder eingehalten wird.
SIGQUIT	beendet den xinetd
SIGTERM	beendet zunächst die aktiven Dienste, anschliessend den xinetd
SIGUSR1	erstellt einen Speicherabzug in der Datei /var/run/xinetd.dump
SIGIOT	veranlaßt xinetd, eine interne Konsistenzprüfung vorzunehmen, um sicherzustellen, dass die Datenstrukturen des Programms nicht durcheinandergeraten sind

In diesem Beispiel werden zunächst Standardwerte für einige Attribute festgelegt. Anschliessend wird Genaueres zum Dienst telnet definiert. Welche Attribute in einem Abschnitt verwendet werden können, hängt vom tatsächlich zu konfigurierenden Dienst ab.

Bei den SUSE-Distributionen beinhaltet die Datei /etc/xinetd.conf oft lediglich den Abschnitt »defaults«. Eine weitere Zeile mit dem Inhalt »includedir /etc/xinetd.d« sorgt dafür, dass alle im Verzeichnis /etc/xinetd.d enthaltenen Dateien gelesen werden, als ob sie in der /etc/xinetd.conf-Datei stünden. In diesen Dateien finden sich dann die Konfigurationen für einzelne Dienste.

/etc/xinetd.d

6.3.3 Starten des xinetd

Der xinetd kann mit einer Reihe von Optionen gestartet werden, die seine Arbeitsweise in speziellen Punkten festlegen: Optionen

-syslog ⟨*Kategorie*⟩ Legt fest, dass der xinetd seine Protokollnachrichten an den syslog-Daemon als Meldung der angegebenen ⟨*Kategorie*⟩ schickt. Mögliche Kategorien sind zum Beispiel daemon, auth, user und die acht Kategorien local0 bis local7.

-filelog ⟨*Dateiname*⟩ Legt den Namen der Datei fest, die dem xinetd als Protokolldatei dient. Diese Option und die Option -syslog schliessen sich gegenseitig aus.

-f ⟨*Dateiname*⟩ Veranlasst, dass der xinetd beim Starten die angegebene Konfigurationsdatei verwendet (der Standardwert ist /etc/xinetd.conf).

-inetd_compat Sorgt dafür, dass der xinetd neben seiner eigenen Konfigurationsdatei auch die Konfigurationsdatei des inetd (/etc/inetd.conf) einliest.

Um den xinetd mit Optionen zu starten, sollten Sie (bei den SUSE-Distributionen) die XINETD_BIN-Variable im Skript /etc/init.d/xinetd um die entsprechenden Optionen erweitern (wobei der Wert der Variable in Anführungszeichen eingeschlossen werden muss).

Es ist auch möglich, den xinetd über Signale zu steuern. Die wichtigsten Signale und ihre Wirkung sind in Tabelle 6.3 zusammengefasst.

6.3.4 Parallelverarbeitung von Anfragen

Dienste, die vom xinetd gestartet werden, können grundsätzlich in zwei Gruppen unterteilt werden (siehe Attribute wait und instances). Wird bei jeden Zugriff

auf den Dienst ein neuer Prozess gestartet, der die Anfrage von aussen entgegennimmt und behandelt, wird der Dienst *multithreaded* genannt. Nimmt ein Dienste erst dann wieder eine neue Anfrage entgegen, wenn die vorherige komplett abgearbeitet wurde, heißt der Dienst *single-threaded*. Datagrammbasierende Dienste (also solche auf UDP-Basis) sind häufig *single-threaded*, während TCP-basierte Dienste immer *multithreaded* sind.

6.3.5 `inetd` durch `xinetd` ersetzen

Grundsätzlich spricht nichts dagegen, auf demselben System den `inetd` und den `xinetd` parallel zu verwenden (außer dass die Paketverwaltung der Distribution sich möglicherweise gegen den Gedanken sträubt). Sie sollten nur darauf achten, dass die beiden nicht versuchen, sich um dieselben Dienste zu kümmern!

Werkzeuge

Möchten Sie auf einem System einen bestehenden `inetd` durch den `xinetd` ersetzen, können Sie auf Werkzeuge zurückgreifen, die die `inetd`-Konfiguration für den `xinetd` nutzbar machen. Standardmässig enthält das `xinetd`-Paket das Programm `itox`. Mit dem Befehl

```
# itox </etc/inetd.conf >/etc/xinetd.conf
```

können Sie aus Ihrer `inetd`-Konfigurationsdatei eine Konfigurationsdatei für den `xinetd` erzeugen. Dabei ist festzuhalten, dass lediglich aktivierte Dienste in die neue Konfigurationsdatei übernommen werden. Wer alle Einträge aus seiner alten Datei übernehmen will, sollte diese zunächst alle aktivieren (indem Sie die Auskommentierung entfernen) und die nicht benötigten Dienste in der neuen Datei wieder deaktivieren.

Übungen

6.6 [!2] Schalten Sie in der Datei `xinetd.conf` (oder der entsprechenden Datei in `/etc/xinetd.d`) den `echo`-Dienst frei und laden Sie die Konfiguration neu. Prüfen Sie die Funktion des Dienstes mit dem Kommando »`telnet localhost echo`«.

6.7 [3] Definieren Sie einen `ps`-Dienst in Analogie zu Übung 6.4 und schränken Sie den Zugriff auf diesen Dienst so ein, dass nur lokale Prozesse Zugriff bekommen.

6.8 [2] Konfigurieren Sie den `xinetd` so, dass der `ps`-Dienst nur auf dem Loopback-Interface (`127.0.0.1`) zur Verfügung steht.

6.9 [3] Welche der beiden Möglichkeiten zur Beschränkung eines Dienstes auf den lokalen Rechner – TCP-Wrapper oder Binden an `127.0.0.1` – ist vorzuziehen?

Kommandos in diesem Kapitel

`inetd` Internet-Superserver, überwacht Ports und startet ggf. Dienste `inetd`(8) 104

`tcpd` „TCP-Wrapper“, erlaubt oder verbietet Zugriff auf Dienste in Abhängigkeit von der IP-Adresse des Clients `tcpd`(8) 106

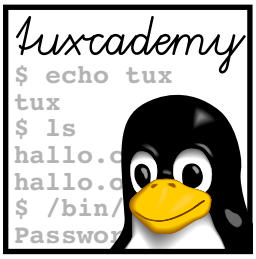
`xinetd` Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste `xinetd`(8) 108

Zusammenfassung

- Der `inetd` beobachtet die Ports für eine ganze Menge von Diensten, deren Server dann nur von Fall zu Fall gestartet werden müssen, statt immer im Hintergrund bereitzustehen.
- Die Konfiguration des `inetd` erfolgt in der Datei `/etc/inetd.conf`.
- Der TCP-Wrapper `tcpd` kann den Zugriff auf Netzwerkdienste auf bestimmte Rechner beschränken.
- Der `xinetd` ist eine modernere Implementierung der Funktionalität von `inetd` und `tcpd` – Portbeobachtung, Zugriffskontrolle und Protokollierung.

Literaturverzeichnis

RFC1413 M. St. Johns. »Identification Protocol«, Februar 1993.
`http://www.ietf.org/rfc/rfc1413.txt`



7

Netzwerkdienste mit systemd

Inhalt

7.1 Vorbemerkung 116
7.2 Persistente Netzwerkdienste 116
7.3 Socket-Aktivierung 118

Lernziele

- Aktivierung persistenter Dienste mit systemd verstehen
- Socket-Aktivierung für Dienste mit systemd verstehen
- Netzwerkdienste in systemd-Ziele integrieren können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 3)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)
- Kenntnisse über systemd

7.1 Vorbemerkung

Systemd ist inzwischen eine extrem verbreitete Alternative zum traditionellen System-V-Init. Aus diesem Grund ist es sinnvoll, auch noch ein paar Worte zum Thema »Netzwerkdienste mit systemd« zu sagen.

Eine ausführliche Einführung zu systemd finden Sie in der Linup-Front-Schulungsunterlage *Linux-Administration I* (ADM1).

Grundsätzlich können wir im Kontext von Netzwerkdiensten unterscheiden zwischen »persistenten« Diensten, die beim Systemstart hochgefahren werden, und Diensten, die bei Bedarf über Socket-Aktivierung gestartet werden. Erstere entsprechen den Diensten, die bei System-V-Init typischerweise über Init-Skripte gestartet werden, letztere den Diensten, um die sich `inetd` oder `xinetd` kümmern.

7.2 Persistente Netzwerkdienste

Persistente Netzwerkdienste unterscheiden sich aus der Sicht von systemd nicht sonderlich von anderen persistenten Diensten, die von systemd gestartet und verwaltet werden. Sie kümmern sich um ihre eigenen Netzwerkverbindungen, so dass systemd damit nichts zu tun hat. Die einzige Aufgabe von systemd besteht darin, den Dienst zum richtigen Zeitpunkt zu starten (so wie System-V-Init das auch machen würde) und gegebenenfalls ein Auge auf den Dienst zu haben, falls er vor seiner Zeit abschmiert (das macht System-V-Init nicht, jedenfalls nicht solange Sie keine Zusatzsoftware installiert haben, die sich darum kümmert).

Sie müssen also eine Unit-Datei für Ihren Dienst zur Verfügung stellen und systemd diese bekanntmachen (mit »`systemctl enable`«). Danach sollte eigentlich alles automatisch gehen. Sie können den Dienst natürlich auch manuell mit »`systemctl start`« & Co. verwalten.

Wenn Sie keine Unit-Datei für Ihren Dienst haben, aber ein Init-Skript für System-V-Init, dann können Sie problemlos dieses verwenden – systemd ist ja kompatibel.

Viele Dienste erwarten, gestartet zu werden, »wenn das Netzwerk zur Verfügung steht«. Das Problem ist, dass es sich dabei um ein sehr schwammiges Konzept handelt. Reicht es aus, dass mindestens eine Netzwerkschnittstelle hochgefahren ist und eine IP-Adresse hat? Müssen tatsächlich andere Stationen erreichbar sein? Das Standard-Gateway? Der DNS-Server? Google? Wie ist das mit WLAN- oder Mobilfunk-Schnittstellen, die mal aktiv sind und mal nicht? Eine der Macken von System-V-Init ist, dass diese Komplexität völlig unter den Teppich gekehrt wird – das Init-Skript für »das Netzwerk« wird ausgeführt, und dann wird postuliert, dass der Rechner im Netz ist und dass sich das auch bis zum Systemhalt nicht mehr ändert.

Systemd verwendet drei verschiedene Ziele, um das Konzept »Netzwerk steht zur Verfügung« abzubilden:

`network-pre.target` ist ein Ziel, das von Diensten benutzt wird, die *vor* der Aktivierung des Netzwerks gestartet werden sollen. Dabei handelt es sich vor allem um Dienste, die eine Firewall konfigurieren, damit diese zur Verfügung steht, bevor Netzwerkschnittstellen aktiviert werden, und so keine »verwundbare Phase« zwischen dem Start des Netzwerks und der Initialisierung des Firewalls existiert. Sie können das Ziel nicht manuell aktivieren.

Netzwerk-Verwaltungssoftware sollte von `network-pre.target` abhängen, allerdings nur in der zeitlichen Ordnung mit

```
[Unit]
After=network-pre.target
```

Eine explizite Abhängigkeit mit `Requires` oder `Wants` sollten Sie vermeiden.

Dienste, die vor der Aktivierung des Netzwerks gestartet werden sollen, sollten

```
[Unit]
Wants=network-pre.target
Before=network-pre.target
```

in ihrer Konfiguration haben.

`network.target` gibt nur an, dass die lokale Netzwerksoftware initialisiert wurde. Es macht keine Annahmen darüber, ob tatsächliche Netzwerkschnittstellen konfiguriert sind. Der Hauptzweck dieses Ziels besteht darin, einen Synchronisierungspunkt für das *Herunterfahren* des Systems zur Verfügung zu stellen – wenn alle Dienste, die »das Netzwerk« voraussetzen, *nach* dem Erreichen von `network.target` gestartet werden, dann impliziert das, dass sie beim Herunterfahren *vor* dem Deaktivieren von `network.target` deaktiviert werden. Das heißt, dass kein Netzwerkdienst brutal seine Verbindung gekappt bekommt.

Sie können `network.target` nicht manuell starten, und das Ziel wird auch nicht von Netzwerkdiensten per `Requires` oder `Wants` vorausgesetzt. So eine Klausel sollte sich nur in der Konfiguration der Netzwerk-Verwaltungssoftware finden. Dienste, die vor dem Netzwerk heruntergefahren werden wollen, sollten

```
[Unit]
After=network.target
```

in ihrer systemd-Konfiguration haben.

`network-online.target` wartet explizit darauf, dass das Netzwerk zur Verfügung steht, wobei die genaue Bedeutung von »zur Verfügung stehen« von der Netzwerk-Verwaltungssoftware festgelegt wird. (Eine konfigurierte externe IP-Adresse wäre zum Beispiel eine naheliegende Voraussetzung.) Im Prinzip können Dienste, die vom Netzwerk abhängen, ein `Requires` oder `Wants` für dieses Ziel haben.

Es wird empfohlen, dies nicht zu übertreiben: Viele Serverdienste haben kein Problem damit, *lokale* Clients zu bedienen, auch wenn keine externe Netzwerkanbindung zur Verfügung steht. Die Abhängigkeit ist eher nützlich für *Clients*, die auf entfernte Dienste zugreifen wollen und ohne eine Netzanbindung nicht funktionieren.

Ein typisches Beispiel dafür sind entfernte Dateisysteme. Aus diesem Grund sorgt systemd dafür, dass für jedes entfernte Dateisystem in `/etc/fstab` eine Abhängigkeit von `network-online.target` generiert wird. Wenn Sie kein entferntes Dateisystem in `/etc/fstab` haben und auch sonst kein Dienst von `network-online.target` abhängt, wird das Ziel beim Systemstart überhaupt nicht berücksichtigt – das ist eine gute Idee, weil dadurch unnötige Verzögerungen vermieden werden, wenn kein Netz zur Verfügung steht.

Systemd interpretiert eine `$network`-Abhängigkeit in einem Init-Skript von System-V-Init als

```
[Unit]
Wants=network-online.target
After=network-online.target
```

Übungen

7.1 [2] Finden Sie heraus, welcher Dienst auf Ihrem System per `Requires` oder `Wants` von `network.target` abhängt. Wie definiert Ihr System ein »aktives Netzwerk«?

7.2 [2] Welche Dienste auf Ihrem System sind von einem »aktiven Netzwerk« abhängig?

7.3 Socket-Aktivierung

»Socket-Aktivierung« ist die Idee, einen Dienst erst dann zu starten, wenn Verbindungswünsche an ihn eingehen. Traditionell wurde das vom `inetd` (später `xinetd`) geleistet, einem Daemon, der auf einer Reihe von Ports lauscht und bei Aktivität gezielt den zum jeweiligen Port gehörenden Hintergrunddienst aktiviert (siehe Kapitel 6). Systemd erweitert dieses Konzept und setzt es konsequent auch für andere Dienste ein.

Systemd propagiert Socket-Aktivierung vor allem für Unix-Domain-Sockets, aber es spricht nichts dagegen, sie auch für TCP- oder UDP-Sockets zu benutzen.

Es gibt drei prinzipielle Szenarien, in denen Socket-Aktivierung sinnvoll ist:

1. Beim Systemstart kann Socket-Aktivierung die Parallelisierung erhöhen und explizite Abhängigkeiten vermeiden. Systemd initialisiert die notwendigen Kommunikationskanäle und organisiert den parallelen Start entsprechender Dienste, wenn Anfragen eingehen. Dies ist nützlich für oft und andauernd gebrauchte Dienste, die so früh wie möglich gestartet werden sollen, etwa syslog oder D-Bus.

2. Selten gebrauchte Dienste werden bei Bedarf gestartet, indem systemd den bekannten Port des Dienstes öffnet und auf Verbindungen wartet. Kommen Verbindungswünsche an, dann startet systemd den eigentlichen Dienst und gibt das *lauschende* Socket an den Dienst weiter. Der Dienst kann dann in eigener Regie weitere Verbindungswünsche behandeln. Ein Beispiel hierfür ist der Druckdienst CUPS.

3. Selten gebrauchte Dienste werden bei Bedarf gestartet, indem systemd den bekannten Port des Dienstes öffnet und auf Verbindungen wartet. Kommen Verbindungswünsche an, dann startet systemd den eigentlichen Dienst und gibt das Socket für die *konkrete Verbindung* an den Dienst weiter. Der Dienst bearbeitet die einzelne Verbindung und beendet sich dann wieder. Dies ist weniger effizient als die anderen beiden Szenarien, aber solche Dienste sind sehr bequem zu entwickeln, da systemd sich um die komplette Netzanbindung kümmert. Dienste, wo sich dieser Ansatz lohnen kann, sind zum Beispiel FTP oder SSH, vor allem auf Rechnern, wo sie selten in Anspruch genommen werden. Man vermeidet dadurch im Hintergrund schlummernde Daemonen.

   Diese Geschmacksrichtung ist die, an die man am ehesten denkt, wenn von `inetd` die Rede ist. `inetd` beherrscht auch das zweite Szenario in unserer Liste, aber das wird nur extrem selten benutzt.

Um einen Dienst wie die Secure Shell (Kapitel 10) mit systemd »Socket-aktivierbar« zu machen, schauen wir uns zunächst an, wie die entsprechende Konfiguration in `inetd` oder `xinetd` aussähe – namentlich für `inetd`:

```
ssh stream tcp nowait root /usr/sbin/sshd sshd -i
```

(die Option »-i« sorgt dafür, dass der SSH-Daemon sshd mit inetd funktioniert statt als freistehender Dienst). Und für xinetd:

```
service ssh {
      socket_type = stream
      protocol = tcp
      wait = no
      user = root
      server = /usr/sbin/sshd
      server_args = -i
}
```

Sie müssen natürlich außerdem wissen, dass die Secure Shell den TCP-Port 22 benutzt. In der inetd-Konfiguration wird das implizit durch das ssh in der ersten Spalte ausgedrückt; dabei handelt es sich um einen Verweis auf die Datei /etc/services, wo die tatsächliche Portnummer zu finden ist.

Um eine äquivalente Konfiguration für systemd zu realisieren, brauchen Sie zwei Dateien. Zuerst einmal müssen Sie den Port beschreiben, auf dem systemd (im Namen der Secure Shell) lauschen soll. Dies geschieht in einer Datei namens sshd.socket[1]:

```
# sshd.socket
[Unit]
Description=Secure Shell service (socket for socket activation)

[Socket]
ListenStream=22
Accept=yes

[Install]
WantedBy=sockets.target
```

ListenStream=22 ist hier das moralische Äquivalent zu »ssh stream tcp« bei inetd, und Accept=yes entspricht dem nowait – systemd soll einzelne Verbindungswünsche im Namen der Secure Shell annehmen und weiterleiten.

Eine zweite Unit-Datei dient dazu, tatsächlich Instanzen des SSH-Daemons zu starten, wenn sie benötigt werden. Da mehrere SSH-Verbindungen gleichzeitig aktiv sein können, definieren wir die Unit-Datei als »Schablone« sshd@.service:

```
# sshd@.service
[Unit]
Description=Secure Shell service (per-connection server)

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

In ExecStart legen wir fest, wie der Daemon gestartet wird (das Minuszeichen am Anfang heißt, dass Rückgabewerte ungleich Null als Erfolg betrachtet werden sollen). StandardInput=socket sorgt dafür, dass der Daemon tatsächlich mit der Clientseite reden kann (die Standardausgabe und die Standardfehlerausgabe ziehen da mit).

Sie werden sich jetzt vielleicht fragen, warum systemd zwei Dateien für etwas braucht, was inetd in einer Zeile schafft. Die Antwort darauf lautet,

[1] Ein Port ist in TCP (und UDP) ein Kommunikations-*Endpunkt*, während ein Socket eine aktive Kommunikations-*Verbindung* beschreibt, also eine Kombination von zwei Ports (Client und Server). Dieser zugegebenermaßen subtile Unterschied hat sich anscheinend nicht bis zu den systemd-Entwicklern herumgesprochen.

dass systemd dadurch viel flexibler ist als `inetd`. Insbesondere sind der »lauschende« Port und die tatsächlichen Verbindungen sauber voneinander getrennt, und Sie können zum Beispiel die Unit für den Port stoppen (und damit künftige Verbindungen verhindern), ohne dass existierende Verbindungen davon beeinflusst werden. (Natürlich geht das auch ohne systemd, nur müssen Sie dann selber herausfinden, welche Prozesse gerade was machen.)

Mit diesen beiden Dateien können wir systemd neu laden und den Dienst starten:

```
# systemctl daemon-reload
# systemctl start sshd.socket
```

Beachten Sie, dass wir die `sshd.socket`-Datei gestartet haben und nicht die `sshd@.service`-Datei. Durch erstere wartet systemd jetzt auf Verbindungen:

```
# systemctl status sshd.socket
● mysshd.socket - Secure Shell service (socket for socket activation)
   Loaded: loaded (/etc/systemd/system/sshd.socket; disabled)
   Active: active (listening) since Di 2015-07-28 13:12:45 CEST; 2s ago
   Listen: [::]:22 (Stream)
 Accepted: 1; Connected: 0
```

Dauerhaft installieren würden Sie den Dienst natürlich mit »`systemctl enable sshd.socket`«. Für unser kleines Experiment hier ist das aber eigentlich nicht nötig.

Wenn Sie (etwa in einem anderen Fenster) `ssh` aufrufen, um mit dem Rechner Verbindung aufzunehmen, sollte das jetzt funktionieren. Der systemd-Status müßte etwas zeigen wie

```
# systemctl --full | grep ssh
sshd@1-192.168.56.101:22-192.168.56.1:46618.service ▷
 ◁ loaded active running  Secure Shell service (per-connection ▷
 ◁ server) (192.168.56.1:46618)
sshd.socket ▷
 ◁ loaded active running  Secure Shell service (socket for socket ▷
 ◁ activation)
```

Die Client-Verbindung könnten Sie jetzt bei Bedarf mit etwas wie

```
# systemctl kill sshd@1-192.68.56.101:22-192.168.56.1:46618▷
 ◁ .service
```

gezielt beenden (erschrecken Sie nicht; systemd macht Kommandozeilenvervollständigung für Dienstnamen, Sie müssen das also nicht alles eintippen).

Es ist keine dumme Idee, die Secure Shell auf diese Weise in systemd einzubinden. Viele Distributionen verlassen sich aber dennoch auf einen dauerlaufenden SSH-Daemon im Hintergrund. Eine Unit-Datei dafür (aus Debian 8) sehen Sie in Bild 7.1.

Systemd ersetzt `inetd`, aber nicht notwendigerweise `xinetd`, da letzterer ein paar eingebaute Bequemlichkeiten hat, die systemd nicht anbietet. Auf die Standarddienste `ECHO`, `DISCARD`, `TIME` und `DAYTIME` können Sie höchstwahrscheinlich verzichten, und diverse der Dienstoptionen sind heute längst nicht mehr relevant (Stichwort »TCPMUX«). Vermissen könnten Sie möglicherweise die TCP-Wrapper, wobei auch dazu zu sagen ist, dass Sie (a) eventuell mit einem richtigen Firewall besser bedient sind, und (b) niemand Sie davon abhält, bei systemd wie einst bei `inetd` das `tcpd`-Programm zu benutzen.

```
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Bild 7.1: Unit-Datei für Secure-Shell-Daemon (Debian 8)

Systemd kann dafür viele nette Sachen, die weder `inetd` noch `xinetd` können. Außerdem ist der Umstand, dass Sie zwei zusätzliche Formate für Konfigurationsdateien *nicht* lernen müssen, ja vielleicht auch etwas wert.

(Dieser Abschnitt ist in weiten Teilen inspiriert von [Poe11].)

Übungen

7.3 [!2] Installieren und starten Sie wie vorstehend gezeigt einen Socket-aktivierten Secure-Shell-Server. Testen Sie einige Verbindungen und probieren Sie auch aus, gezielt bestimmte Verbindungen abzubrechen. (Für den – wahrscheinlichen – Fall, dass es auf Ihrem System schon einen aktiven Secure-Shell-Server gibt: Nennen Sie die Dateien `mysshd.socket` und `mysshd.service` und wählen Sie einen anderen Port, etwa 10022.)

7.4 [2] Geben Sie eine `.socket`- und `.service`-Datei für systemd an, die den klassischen DAYTIME-Dienst realisieren. Dieser liefert auf dem TCP-Port 13 das aktuelle Datum nebst Uhrzeit:

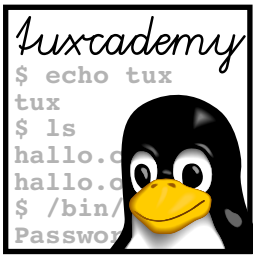
```
$ telnet localhost 13
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Di 28. Jul 13:56:14 CEST 2015
Connection closed by foreign host.
```

Zusammenfassung

- Persistente Netzwerkdienste unterscheiden sich bei systemd nicht sonderlich von anderen persistenten Diensten.
- Systemd regelt die Aktivierung von Diensten in Abhängigkeit von der Verfügbarkeit des Netzwerks über die Ziele `network-pre.target`, `network.target` und `network-online.target`.
- Systemd bietet flexible und leistungsfähige Möglichkeiten zur »Socket-Aktivierung« von Diensten bei Bedarf.

Literaturverzeichnis

Poe11 Lennart Poettering. »Converting inetd Services (systemd for Administrators, Part XI)«, September 2011. http://0pointer.de/blog/projects/inetd.html



8

Die Systemzeit

Inhalt

8.1 Einführung 124
8.2 Uhren und Zeit unter Linux 124
8.3 Zeitsynchronisation mit NTP 126

Lernziele

- Lernen, wie Linux Uhrzeit und Datum verwaltet
- Die Zeit auf einem Rechner über NTP steuern können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 3)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)

8.1 Einführung

Das Konzept »Zeit« ist tatsächlich ein ziemlich komplexes Thema. Unsere »bürgerliche« Zeit, also die Zeit, die am Telefon angesagt wird, nach der Züge, Busse und Flugzeuge verkehren und nach der sich das Fernsehprogramm richtet, leitet sich im wesentlichen aus zwei Quellen her: Die Uhren der Welt laufen gemäß
UTC der sogenannten »koordinierten Weltzeit« (*coordinated universal time* oder UTC), typischerweise mit einem festen Versatz für die Zeitzone – die »mitteleuropäische Zeit« oder MEZ ergibt sich zum Beispiel, indem man zur UTC eine Stunde addiert. Die UTC wiederum basiert grundsätzlich auf der »internationalen Atom-
TAI zeit« (TAI), einem gewichteten Mittel der Zeit von etwa 300 Atomuhren in über 50 nationalen Laboratorien auf der ganzen Welt, in dem Sinne, dass eine Sekunde in der UTC einer Sekunde in der TAI entspricht.


Die Atomuhren synchronisieren sich untereinander über Satellit auf eine Abweichung von um die 0,1 Millisekunden (es gibt ein paar Ausreißer, die es nur auf etwa 10 Millisekunden schaffen).

Die UTC wird außerdem an die mittlere Sonnenzeit auf dem Meridian von
UT1 Greenwich (auch UT1 genannt) angepasst. Da die Erddrehung sich durch Effekte wie Gezeitenreibung und Plattentektonik allmählich auf im Detail unvorhersehbare Weise verlangsamt, sind die Sekunden der UT1 geringfügig länger als die
Schaltsekunden der UTC. Darum muss die UTC hin und wieder eine »Schaltsekunde« einlegen, damit die Atomzeit UTC und die astronomische Zeit UT1 nicht zu sehr auseinanderdriften. Die Regel ist, dass UTC und UT1 nicht um mehr als 0,9 Sekunden voneinander abweichen sollen.


Schaltsekunden passieren potentiell zweimal im Jahr, typischerweise Ende Juni oder Ende Dezember. Sie werden in der Regel 6 Monate vorher angekündigt. Seit der Einführung des Konzepts 1972 bis Juli 2015 gab es 26 Schaltsekunden. (1972 wich die UTC schon 10 Sekunden von der TAI ab, so dass die Differenz jetzt insgesamt 36 Sekunden beträgt.)


Schaltsekunden werden zunehmend als lästig empfunden. Bei der Schaltsekunde vom 30. Juni 2012 zum Beispiel kam ein Fehler im Linux-Kern zum Vorschein, der im Extremfall dazu führen konnte, dass Rechner sich komplett aufhängten. Seit einigen Jahren ist deshalb bei der ITU, dem zuständigen internationalen Gremium, ein Vorschlag anhängig, Schaltsekunden komplett abzuschaffen. Die Meinungen darüber gehen vehement auseinander; eine Entscheidung wurde erst mal bis November 2015 vertagt.


Verwechseln Sie die Schaltsekunden nicht mit den Schalttagen des Gregorianischen oder den Schaltmonaten des islamischen Kalenders. Während letztere sich durch die Definition des Kalenders ergeben und – bis auf Kalenderreformen – vorhersehbar sind, beruhen Schaltsekunden auf astronomischen Beobachtungen und werden nach Bedarf, wenn auch mit einer gewissen Vorwarnung, eingestreut.

Für uns als Menschen im wirklichen Leben sind die UTC und die davon abgeleiteten »Zonenzeiten« wie die MEZ am wichtigsten. Entsprechend sind die Schwerpunkte, die Linux für den Umgang mit Zeit setzt.

8.2 Uhren und Zeit unter Linux

CMOS-Uhr Jeder PC hat eine batteriebetriebene Hardware- oder CMOS-Uhr, die über die Firmware gestellt werden kann und auch dann weiterläuft, wenn der Rechner ausgeschaltet ist. Linux verwendet die CMOS-Uhr nur beim Systemstart zum Stellen
Kernel-Uhr der internen Kernel-Uhr.

Die Kernel-Uhr zählt die Zeit in fortlaufenden Sekunden seit dem 1.1.1970, 0 Uhr UTC. Beim Systemstart wird die Zeitangabe aus der CMOS-Uhr gelesen, in die Kernel-Zeit umgerechnet und zur Initialisierung der Kernel-Uhr benutzt; hierzu dient das Programm `hwclock`. Das System muss wissen, ob die CMOS-Uhr auf UTC oder die lokale Zonenzeit (etwa MEZ/MESZ) gestellt ist; letzteres wird möglicherweise von anderen auf dem Rechner installierten Betriebssystemen gefordert:

`hwclock`

```
# hwclock --hctosys -u          CMOS-Uhr in UTC
# hwclock --hctosys             CMOS-Uhr in Zonenzeit
```

Die Kernel-Uhr wird zum Beispiel für Zeitstempel von Dateien herangezogen. Abfragen können Sie sie mit dem Kommando `date`, während `hwclock` ohne Argumente die Zeit gemäß der CMOS-Uhr ausgibt.

Linux auf 32-Bit-Systemen verwendet eine vorzeichenbehaftete 32-Bit-Ganzzahlvariable zum Speichern der Zeit. Der späteste darstellbare Zeitpunkt ist also $2^{31} - 1 = 2.147.483.647$ Sekunden nach dem 1.1.1970, 0 Uhr UTC, mit anderen Worten der 19. Januar 2038, 3 Uhr, 14 Minuten, 7 Sekunden[1]. Bis dahin benutzen wir alle hoffentlich wenn nicht 64-Bit-Rechner, so doch zumindest einen angepassten Linux-Kern …

Wenn Sie in Ihren eigenen Programmen eine Zeitdarstellung brauchen, sollten Sie nicht notwendigerweise die des Linux-Kerns verwenden, denn diese ist vor allem für betriebssysteminterne Zwecke gedacht (etwa die schon erwähnten Zeitstempel auf Dateien). Als Programmierer für eine Bank oder Versicherung – nur mal so als Beispiel – könnten Sie es heute schon mit Zeitpunkten nach 2038 zu tun bekommen, etwa wenn Sie einem heute Dreißigjährigen eine Kapitallebensversicherung verkaufen wollen, die ausgezahlt wird, wenn er mit 65 (oder dann vermutlich 75) in den Ruhestand geht. Umgekehrt sind sicher viele Ihrer Kunden vor dem 1.1.1970 geboren worden, und negative Linux-Zeitstempel sind nicht wirklich definiert (auch wenn die C-Bibliothek meist das Richtige mit ihnen tut).

Die Zeitzone steht in `/etc/timezone`, dort findet sich ein Eintrag wie »`Europe/Berlin`«, der eine Datei unterhalb von `/usr/share/zoneinfo` benennt. Diese (unleserliche) Datei enthält die Angaben zur Zeitzone wie den Versatz gegenüber der UTC, die Sommerzeitregelungen und ähnliche Details. `/etc/localtime` ist eine Kopie der durch `/etc/timezone` benannten Datei. Benutzer können über die Umgebungsvariable `TZ` eine beliebige andere Zeitzone wählen.

Linux liefert diverse Werkzeuge zur Verwaltung von Zeitzonendateien mit: Mit dem »Zeitzonen-Compiler« `zic` können Sie Ihre eigenen Zeitzonendateien erstellen und ins von der C-Bibliothek geforderte Format übersetzen; `zdump` gibt eine Zeitzonendatei (oder die meisten Informationen daraus) in lesbarer Form aus. Ebenfalls lesenswert sind die Handbuchseiten `tzset`(3) und `tzfile`(5).

Mehr über Zeitzonen unter Linux steht in der Linup-Front-Schulungsunterlage *Linux für Fortgeschrittene*.

Die Kernel-Uhr stellen Sie mit dem Programm `date`, indem Sie die gewünschte Uhrzeit als Parameter angeben (Administratorrechte sind erforderlich): Kernel-Uhr stellen

```
# date 020318012009.30
```

stellt die Uhr auf 18:01:30 Uhr am 3. Februar 2009 (wir überlassen es Ihnen als Übung, herauszufinden, wie sich der Parameter ergibt). Als Minimum angeben müssen Sie Tag, Monat, Stunde und Minute:

[1] So ungefähr zumindest – da bis dahin noch die eine oder andere Schaltsekunde eingelegt werden wird.

```
# date 02031801
```

Mit der Option `-u` stellen Sie die Uhr in UTC.

Beim unter Linux verbreiteten GNU-`date` haben Sie die Möglichkeit, auf eine etwas intuitivere Syntax zum Stellen der Uhr zurückzugreifen:

```
# date --set --date="2015-07-20 18:01:30 +0100"
```

Möglicherweise erlaubt Ihnen auch Ihre grafische Arbeitsumgebung das Stellen der Uhr auf bequemere Weise.

CMOS-Uhr stellen

Um die CMOS-Uhr im laufenden Betrieb zu stellen, sollten Sie zuerst die Kernel-Uhr mit `date` stellen. Anschließend können Sie die Kernel-Zeit mit »`hwclock --systohc`« in die CMOS-Uhr übertragen. Alternativ erlaubt die Option `--date` von `hwclock` das direkte Stellen der CMOS-Uhr (ohne dass die Kernel-Uhr das mitbekommt). In jedem Fall versucht `hwclock`, Daten über die systematische Abweichung der CMOS-Uhr in `/etc/adjtime` zu protokollieren. CMOS-Uhren sind in der Regel grässlich ungenau.

Gängige Linux-Distributionen übertragen beim Herunterfahren des Rechners die Zeit der Kernel-Uhr in die CMOS-Uhr. Dies basiert auf der Prämisse, dass die Kernel-Zeit mit den im nächsten Abschnitt skizzierten Mitteln sehr genau gehalten wird, während die CMOS-Zeit irgendwo ins Dorthinaus abdriftet – demnach ist es keine dumme Idee, sie zumindest hin und wieder auf den Pfad der Tugend zurückzuholen.

Übungen

8.1 [1] Linux für Börsenmakler: Geben Sie die Kommandos an, mit denen Sie drei Uhren (etwa mit `xclock`) auf den Bildschirm bringen können, die die Zeit der wesentlichen Börsen (New York, Frankfurt, Tokyo) anzeigen.

8.2 [2] In welchen Zeiträumen seit dem Beginn der Linux-Zeitzoneninformation galt in Deutschland die Sommerzeit? (*Tipp:* `zdump`)

8.3 Zeitsynchronisation mit NTP

In einem Rechnernetz ist es oft wichtig, dass alle Stationen in etwa die gleiche Zeit verwenden. Netzwerkdateisysteme wie NFS oder Authentisierungs-Infrastrukturen wie Kerberos reagieren höchst allergisch auf Rechner, deren Uhrzeit deutlich von der des Servers abweicht, und ein stabiler Systembetrieb ist so nicht zu gewährleisten. Es ist also eine ausgesprochen vernünftige Idee, die Uhrzeit aller Rechner im lokalen Netz so weit wie möglich zu synchronisieren, am besten automatisch.

Das traditionelle Programm zur Zeitsynchronisation heißt `netdate`. Allerdings sollten Sie darum einen weiten Bogen machen, da es nicht richtig funktioniert.

Ebenso ist es sinnvoll, die Uhrzeit aller Rechner im Netz nicht nur untereinander zu synchronisieren, sondern sie möglichst mit einer genauen externen Zeitbasis abzugleichen, etwa einer Atomuhr. Da Sie vermutlich nicht zu den Glücklichen (?) gehören, die eine solche im Keller stehen haben, werden Sie entweder auf einen Empfänger für Signale vom DCF77-Zeitsender oder GPS-Satelliten oder einen über das Internet zugänglichen Zeitserver setzen. Öffentlich zugängliche Zeitserver werden oft von Universitäten oder Internet-Providern betrieben.

Es ist ungeschickt, die Uhr »sprunghaft« zu stellen, da dadurch Zeitpunkte ausfallen oder doppelt vorkommen können. Das führt möglicherweise zu Problemen, etwa mit `cron`. Es ist besser, die Kernel-Zeit im laufenden Betrieb dadurch korrekt zu halten, dass Sie sie kontrolliert langsamer oder schneller laufen lassen, um Abweichungen auszugleichen, ohne dass die Folge der Sekunden seit dem 1.1.1970 unterbrochen wird. Hierzu sollten Sie das *Network Time Protocol* (NTP) einsetzen. Mehr Informationen hierzu finden Sie im [RFC1305] oder unter `http://www.ntp.org/`. NTP

Ein verbreiteter Daemon zur Zeitsynchronisation heißt `ntpd`. Er kann als Client auftreten und über NTP mit Funkuhren oder Zeitservern kommunizieren sowie als Server die synchronisierte Zeit an andere Rechner weitergeben. Konfiguriert wird `ntpd` in der Datei `/etc/ntp.conf`, in der zum Beispiel das folgende stehen könnte: `ntpd`

```
# Lokale Uhr -- keine gute Zeitquelle
server 127.127.1.0
fudge  127.127.1.0 stratum 10   # unsynchronisiert
# Zeitserver aus dem öffentlichen Pool
server 0.de.pool.ntp.org iburst
server 1.de.pool.ntp.org iburst
server 2.de.pool.ntp.org iburst
# Diverses
driftfile /var/lib/ntp/ntp.drift
logfile   /var/log/ntp
```

Der erste `server`-Eintrag steht für die lokale Uhr, die nicht als zuverlässig gilt und nur in Notfällen herangezogen wird (etwa wenn kein Zeitserver erreichbar ist). Der `stratum`-Wert beschreibt die »Entfernung« der Uhr von der offiziellen Atomzeit; ein Rechner, der direkt an der Atomuhr hängt, befindet sich auf `stratum` 1, ein Rechner, der seine Zeit von diesem bekommt und nicht von der Atomuhr, ist auf `stratum` 2 und so weiter.

Einen Zeitserver zu finden ist mitunter gar nicht so einfach. NTP mit vielen Clients kann ein Netz bzw. einen Zeitserver ziemlich belasten, weswegen Institutionen wie die Physikalisch-Technische Bundesanstalt in Braunschweig, die früher öffentliche Zeitserver angeboten haben, inzwischen davon abgekommen sind. Der beste Ansatz, wenn Sie nicht direkt auf einen Zeitserver zugreifen können, den Ihr Provider oder Ihr Unternehmen betreibt, besteht in der Verwendung des »NTP-Pools«.

In Active-Directory-Netzen auf Microsoft-Basis dienen die Domänencontroller auch als NTP-Zeitserver. Dies ist sinnvoll, weil Active Directory als Kerberos-basiertes System eine halbwegs einheitliche Zeit im Netz voraussetzt.

Im NTP-Pool stehen diverse öffentliche Zeitserver zur Verfügung, die von Clients über ein DNS-»Ringelreihen«-Verfahren angesprochen werden. Das heißt, eine Adresse wie `0.pool.ntp.org` verweist auf einigermaßen zufällige Weise auf einen von einigen tausend öffentlichen Zeitservern irgendwo auf der Welt. Da alle in etwa dieselbe Zeit liefern, ist das für einen Client nicht wirklich ein Problem – aber für die Server-Betreiber heißt es, dass die Last sich gleichmäßig verteilt, statt sich auf ein paar Zeitserver zu konzentrieren, nur weil deren Namen besonders bekannt sind.

Die `iburst`-Option auf den `server`-Zeilen sorgt dafür, dass Ihr `ntpd` sich beim Programmstart sehr zügig die aktuelle Zeit holt.

In der Praxis sollten Sie wie im Beispiel oben drei Zeitserver aus dem NTP-Pool angeben:

```
server 0.pool.ntp.org iburst                    Irgendwo auf der Welt
server 1.pool.ntp.org iburst
server 2.pool.ntp.org iburst
```

Eine bessere Qualität bekommen Sie aber, indem Sie sich auf geographische »Teilpools« konzentrieren:

```
server 0.europe.pool.ntp.org                    Irgendwo in Europa
server 0.de.pool.ntp.org                        Irgendwo in Deutschland
```

Dadurch helfen Sie auch die Netzlast geringer halten. Wenn Ihr Provider einen Zeitserver anbietet, können Sie auch diesen und zwei Zeitserver aus dem Pool verwenden.

Neuere Versionen von `ntpd` unterstützen die Direktive `pool`, die für die Verwendung von NTP-Pools optimiert ist:

```
pool de.pool.ntp.org
```

Sie können mehr als eine `pool`-Direktive angeben (doppelte Server werden entfernt), aber prinzipiell reicht eine.

Wenn Sie die Zeit für ein komplettes Netz mit dem NTP-Pool zu synchronisieren gedenken, dann sollten Sie mindestens einen Rechner aus Ihrem Netz zum NTP-Server machen und nur diesen mit dem NTP-Pool abgleichen. Die anderen Rechner in Ihrem Netz sollten sich die Zeit dann von Ihrem lokalen Zeitserver holen.

In dieser Situation sollten Sie es wahrscheinlich nicht bei einem einzigen Zeitserver belassen. Konfigurieren Sie mindestens zwei (zum Beispiel `ntp1.example.com` und `ntp2.example.com`) und verweisen Sie vom einen auf den anderen mit

```
peer ntp2.example.com   # auf ntp1, auf ntp2 entsprechend andersrum
```

Damit können die beiden Zeitserver sich gegenseitig synchronisieren. Auf den Clients benutzen Sie dann entsprechend

```
server ntp1.example.com iburst
server ntp2.example.com iburst
```

Wohlgemerkt: Damit können Sie Ausfälle eines der Zeitserver überbrücken, aber Ihre Internet-Anbindung zu den externen Zeitservern bleibt ein Zuverlässigkeitsengpass. Wenn Sie sicher gehen wollen, dann brauchen Sie mehrere unabhängige Internet-Anbindungen (im Idealfall solche, die nicht durch denselben Kabelkanal nach draußen geführt sind und dort einem übereifrigen Bagger zum Opfer fallen können). Wenn Sie die nicht aus anderen Gründen schon haben, ist es an diesem Punkt eventuell billiger, ein paar DCF77- oder GPS-Empfänger zu kaufen.

Wenn Sie ein großes lokales Netz haben, dann können die ständigen Synchronisierungsnachrichten der verschiedenen `ntpd`s eine beträchtliche »Grundlast« verursachen. In so einer Situation ist es geschickter, den Server als »Broadcast-Server« zu konfigurieren, der in gewissen Abständen unaufgefordert Zeitansagen »an alle« verbreitet. Mit etwas wie

```
broadcast 192.168.0.255
```

können Sie Ihren `ntpd` zum Broadcast-Server machen, der Zeitansagen an das Netz `192.168.0.0/24` schickt. (Natürlich muss der Broadcast-Server seine Zeit auch von irgendwoher bekommen; die `server`- oder `pool`-Direktiven brauchen Sie also weiterhin.) Auf den Clients sollten Sie die Direktive

```
broadcastclient
```

angeben – `server` oder `pool` sind dort nicht nötig.

Beim Broadcast-Konzept kann ein Angreifer leicht als Broadcast-Server auftreten und falsche Uhrzeiten verbreiten. Aus diesem Grund ist es sinnvoll, die Zeitansagen kryptografisch zu authentisieren. (Tatsächlich wird es im Normalfall vorausgesetzt und muss gegebenenfalls explizit ausgeschaltet werden.) Im einfachsten Fall erzeugen Sie dazu einen Satz symmetrische Schlüssel mit dem Kommando `ntp-keygen`:

```
# mkdir /etc/ntp-keys
# cd /etc/ntp-keys
# ntp-keygen -M
Using OpenSSL version OpenSSL 1.0.1k 8 Jan 2015
Using host blue group blue
Generating new md5 file and link
ntpkey_md5_blue->ntpkey_MD5key_blue.3646747865
```

Dies generiert eine Schlüsseldatei `/etc/ntp-keys/ntpkey_MD5key_blue.3646747865` mitsamt einem symbolischen Link `ntpkey_md5_blue` im selben Verzeichnis. Die Datei enthält zehn MD5- und zehn SHA1-Schlüssel, von denen Sie sich einen beliebigen aussuchen dürfen:

```
# cat /etc/ntp-keys/ntpkey_md5_blue
# ntpkey_MD5key_blue.3646747865
# Fri Jul 24 19:31:05 2015

 1 MD5 hPQB+WrQH|XwILq!Na,  # MD5 key
 2 MD5 devt/tV(zTA@_w5EG6;  # MD5 key 3 MD5 Evk8O2ylOEySK4[C@&g  # MD5 key
<<<<<
10 MD5 $&,7*SQGITy-t?B/8pb&  # MD5 key
11 SHA1 92fc0c06cfe754a949ee79497d59c378878c4ac1  # SHA1 key
12 SHA1 a300fe27c8765a96139ac7f4dcc3f65c78e7c341  # SHA1 key
<<<<<
```

Den Index (linke Spalte) des gewünschten Schlüssels geben Sie dann in der Datei `/etc/ntp.conf` an:

```
keys /etc/ntp-keys/ntpkey_md5_blue
broadcast 192.168.0.255 key 1
```

Die Schlüsseldatei muss auch auf den Clients vorliegen. Dort müssen Sie die folgenden Zeilen in `/etc/ntp.conf` eintragen:

```
keys /etc/ntp-keys/ntpkey_md5_blue
trustedkey 1
broadcastclient
```

Wichtig: Die symmetrischen Schlüssel sollten nicht für normale Benutzer lesbar sein.

Neue Versionen von `ntpd` unterstützen außerdem ein asymmetrisches Verschlüsselungsverfahren. Details dazu stehen in der Dokumentation von `ntpd`.

ntp.drift In `ntp.drift` werden systematische Abweichungen der CMOS-Uhr festgehalten. `ntpd` muss sie dafür eine Weile beobachten, aber funktioniert dann potenziell auch ohne ständige Rückbezüge auf die Zeitserver.

Es gibt keine offiziellen Handbuchseiten für `ntpd`. Die Dokumentation steht nur im HTML-Format zur Verfügung, etwa auf `http://doc.ntp.org/`

ntpdate Sie können die Uhr »ungefähr« stellen, indem Sie das Programm `ntpdate` verwenden, das Sie einfach mit einem oder mehreren Zeitservern als Argument aufrufen können. Damit wird die Zeit einmalig gestellt, was natürlich nicht so schön ist, wie sie von `ntpd` laufend korrigieren zu lassen, aber manchmal auch schon ausreicht (vor allem, wenn Sie es über `cron` regelmäßig wiederholen).

```
# ntpdate 0.de.pool.ntp.org 1.de.pool.ntp.org
# hwclock --systohc
```

Dieser Ansatz ist auch nötig, wenn Sie eine ältere `ntpd`-Version haben, die die `iburst`-Option für Server nicht unterstützt. Damit `ntpd` seine Feinkorrekturen vornehmen kann, muss seine Zeit nämlich schon zumindest in etwa stimmen, und dabei hilft `ntpdate`.

`ntpd` kann einen Einwahlzugang ständig beschäftigen. Sie sollten es nicht zur Synchronisation mit entfernten Zeitservern einsetzen, wenn Sie nicht über eine Standleitung verfügen. Eine Alternative für Inhaber von zeitbasiert abgerechneten
chrony Einwahlzugängen ist `chrony` (nicht Stoff von LPIC-1); siehe `http://chrony.sunsite.dk/`.

ntpq Mit dem Programm `ntpq` können Sie NTP-Server steuern. Es unterstützt eine Reihe von Kommandos, die Sie entweder auf der Kommandozeile oder in einem interaktiven Modus eingeben können. Zum Beispiel können Sie sich die aktuell verwendeten Zeitserver anschauen:

```
$ ntpq -c peers
     remote         refid    st t when poll reach delay  offset jitter
==============================================================================
+panel1.web2.c 5.9.80.113     3 u    7   64   17 33.347 -78.427  1.152
+alvo.fungus.a 91.195.238.4   3 u    7   64   17 32.989 -84.727  2.352
*ntp3.kashra-s .PPS.          1 u    3   64   17 61.659 -80.267  1.383
 liste.cc      192.53.103.104 2 u    2   64   17 34.967 -78.630  1.063
```

(»`ntpq -p`« würde das Gleiche machen.)

Kurz zur Erklärung: `remote` ist der (möglicherweise entfernte) Zeitserver. `refid` ist die Quelle, von der der Zeitserver seine Zeit bekommt. `st` ist das Stratum, also die Entfernung des Zeitservers von der Atomzeit. `t` gibt an, welche Rolle Ihr Rechner in Bezug auf die Gegenstelle spielt – `u` steht für »Unicast-Client«, andere mögliche Werte wären `b` für »Broadcast- oder Multicast-Client«, `l` für eine lokale Referenzuhr, `s` für einen symmetrischen »Peer« und so weiter. `when` bezeichnet die Zeitspanne seit dem letzten Kontakt zur Gegenstelle – eine Zahl ohne Einheit steht für Minuten, `h` und `d` nach der Zahl sind respektive Stunden und Tage. `poll` gibt die Abfragefrequenz an; offiziell reichen die Werte von 4 bis 17 (2^4 = 16 Sekunden bis 2^{17} = 131072 Sekunden, also ca. 36,4 Stunden), tatsächlich antreffen werden Sie eher Werte in Sekunden von 64 bis 1024. `reach` spiegelt wider, wie oft die Abfragen in der letzten Zeit erfolgreich waren: Interpretieren Sie den Wert als oktale Darstellung eines 8-Bit-Schieberegisters, bei dem die jüngste Abfrage das niederwertigste Bit repräsentiert, der Wert 17 in unserem Beispiel steht also für vier erfolgreiche Abfragen (missglückte Abfragen erscheinen als Null-Bits). Zum Schluss gibt `delay` die Zeit für eine »Rundreise« eines Datagramms zur Gegenstelle in Millisekunden an, `offset`

die mittlere Differenz zwischen den Zeiten dieses Rechners und der Gegenstelle (ebenfalls in Millisekunden)[2] und `jitter` die mittlere Fluktuation im Zeitsignal der Gegenstelle, also das quadratische Mittel der Differenz zwischen aufeinanderfolgenden Zeitangaben (ebenfalls in Millisekunden).

Das erste Zeichen jeder Zeile gibt den Status der Gegenstelle an:

`Leerzeichen` Die Gegenstelle redet nicht mit diesem Rechner, *ist* dieser Rechner oder benutzt diesen Rechner als Zeitquelle.

`x` (oder »-«) Dieser Rechner wird ignoriert, weil seine Zeit zu ungenau wirkt.

`#` Gute Gegenstelle, die aber trotzdem ignoriert wird (weil es sechs bessere gibt); kommt als Ersatz in Frage.

`+` Gute Gegenstelle, die berücksichtigt wird.

`*` Aktuell die bevorzugte (»primäre«) Gegenstelle.

Die `refid` kann eine ganze Reihe von verschiedenen Werten haben. Höchstwahrscheinlich sehen werden Sie entweder eine IP-Adresse oder aber eine der gängigen Abkürzungen:

`.LOCL.` Lokale Uhr der unzuverlässigen Sorte (mit sehr hohem Stratum-Wert).

`.PPS.` (engl. *pulse per second*) Ein sehr genaues Zeitsignal, also eine Atomuhr oder ein GPS-Empfänger. GPS-Satelliten gelten als Atomuhren[3]; ein Rechner mit einem GPS-Empfänger ist also auf Stratum 1. Die maximale Ungenauigkeit ist im Bereich von Mikrosekunden pro Sekunde. Das PPS-Signal liefert nur eine hochpräzise Folge von Sekunden, wie eine Art Metronom; wie spät es tatsächlich ist, müssen Sie aus einer anderen Quelle wissen.

`.DCFa.` (und `.DCFp.`) Das DCF77-Zeitsignal, das von einem Langwellensender in Mainflingen (südöstlich von Frankfurt am Main) ausgestrahlt wird und in fast ganz Europa zu empfangen ist. Die erreichbare Genauigkeit hängt davon ab, wieviel Aufwand Sie auf der Empfängerseite treiben wollen; ganz einfache und billige Empfänger wie die in gängigen Funkuhren synchronisieren sich mit einer Abweichung von ±0,1 Sekunde, während genauere Empfänger mit dem amplitudenmodulierten Signal (`.DCFa.`) in der Praxis eine Abweichung im Bereich niedriger einstelliger Millisekundenzahlen schaffen. Wenn man das phasenmodulierte Zeitsignal (`.DCFp.`) ausnutzt und sich wirklich Mühe gibt, sind – in Abhängigkeit von der Tages- und Jahreszeit – sogar Abweichungen von wenigen Mikrosekunden erreichbar.

`.BCST.` Die Gegenstelle ist ein Netz, dem dieser Rechner als Broadcast-Server dient.

Das Programm `ntpq` unterstützt eine ganze Reihe Kommandos, über die Sie mit NTP-Servern reden und Daten abfragen oder (wenn Sie geeignete Zugriffsrechte haben) auch deren Konfiguration ändern können. Hier sind noch ein paar Beispiele:

```
$ hostname
blue
$ ntpq
ntpq> peers
     remote           refid   st t when poll reach delay offset jitter
```

[2] Hierbei wird das quadratische Mittel angegeben, für n Werte also $\sqrt{(x_0^2 + x_1^2 + ... + x_{n-1}^2)/n}$. Dies hat den Effekt, dass größere Abweichungen stärker ins Gewicht fallen.

[3] Was technisch gesehen sogar stimmt …

```
=============================================================================
*red.example.com 129.70.132.37 3 u    1   64  377  1.375 -0.074  0.482
ntpq> associations

ind assid status  conf reach auth condition  last_event cnt
===========================================================
  1  3445  765a    no   yes   ok   sys.peer    sys_peer  5
ntpq> readvar 3445
associd=3445 status=765a authenb, auth, reach, sel_sys.peer, 5 events,▷
 ◁ sys_peer,
srcadr=red.example.com, srcport=123, dstadr=192.168.56.102,
dstport=123, leap=00, stratum=3, precision=-23, rootdelay=37.216,
rootdisp=40.298, refid=129.70.132.37,
reftime=d95fc8c2.3be84526  Sun, Jul 26 2015 22:59:46.234,
rec=d95fc9d8.309cefd3  Sun, Jul 26 2015 23:04:24.189, reach=376,
unreach=0, hmode=6, pmode=5, hpoll=6, ppoll=6, headway=0, flash=00 ok,
keyid=1, offset=0.245, delay=1.375, dispersion=0.960, jitter=0.524,
xleave=0.273,
filtdelay=   1.38  1.38  1.38  1.38  1.38  1.38  1.38  1.38,
filtoffset=  0.24  0.47 -0.07  0.56  0.04 -0.31 -0.06 -0.86,
filtdisp=    0.00  1.01  2.00  2.96  3.95  4.95  5.94  6.92
ntpq> readvar 3445 stratum
stratum=3
```

Übungen

8.3 [2] Diskutieren Sie die Vor- und Nachteile der verschiedenen bei `ntpd` möglichen Zeitquellen – Funkuhr oder Zeitserver im Internet.

8.4 [1] Benutzen Sie `ntpdate`, um die Uhr Ihres Rechners mit einem Zeitserver im Internet zu synchronisieren.

8.5 [!2] Konfigurieren Sie auf Ihrem Rechner einen NTP-Server, der dessen Zeit mit der eines oder mehrerer geeigneter Zeitserver synchronisiert. Vergewissern Sie sich, dass der NTP-Server beim Start die Uhr stellt, auch wenn die Zeit vorher überhaupt nicht richtig war (*Tipp:* Halten Sie den `ntpd` an, verstellen Sie die Uhr mit `date` und starten Sie den `ntpd` dann wieder.)

8.6 [3] Verwenden Sie eine geeignete Testumgebung (Labor- oder Schulungsraumnetz oder virtualisiertes LAN), um einen lokalen NTP-Server nebst Client(s) aufzusetzen. Letztere sollen ihre Zeit vom lokalen Server holen (nicht aus dem Internet). Experimentieren Sie auch mit mehreren NTP-Servern im LAN.

8.7 [2] (Aufbauend auf der vorigen Aufgabe.) Testen Sie die Verteilung von Zeitinformationen per Broadcast. Konfigurieren Sie dazu einen Rechner als Broadcast-Server (generieren Sie dafür einen Satz Schlüssel) und einen anderen als Broadcast-Client. Verstellen Sie auf dem Client die Uhr und starten Sie dort den `ntpd` neu. Beobachten Sie mit Wireshark, wie der Client seine Uhr neu stellt. Wie lange dauert das? Was passiert, wenn Sie anschließend die Uhr noch einmal verstellen, ohne `ntpd` neu zu starten?

Kommandos in diesem Kapitel

date	Gibt Datum und Uhrzeit aus	date(1)	125
hwclock	Steuert die CMOS-Uhr eines PCs	hwclock(8)	124
ntp-keygen	Erzeugt Schlüsselmaterial für ntpd	ntp-keygen(8)	129
ntpq	Steuert NTP-Server	ntpq(8)	130
zdump	Gibt die aktuelle Zeit oder die Zeitzonendefinitionen für verschiedene Zeitzonen aus	zdump(1)	125
zic	Compiler für Zeitzonen-Dateien	zic(8)	125

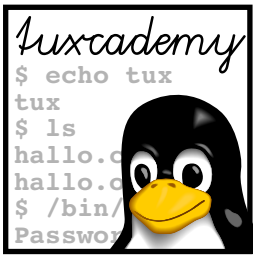
Zusammenfassung

- Ein PC-basiertes Linux-System hat zwei Uhren: Die Uhr im Kernel und eine batteriebetriebene CMOS-Uhr. Linux verwendet die CMOS-Uhr nur zum Stellen der Kernel-Uhr beim Systemstart.
- Linux verwendet eine interne Uhr, die in Sekunden fortlaufend zählt.
- Zur Verwaltung der CMOS-Uhr dient das Programm hwclock.
- Mit ntpd können Sie die Zeit Ihres Linux-Rechners über NTP mit einer offiziellen Zeitbasis synchronisieren.

Literaturverzeichnis

Clock-Mini-HOWTO Ron Bean. »The Clock Mini-HOWTO«, November 2000.
http://www.tldp.org/HOWTO/Clock.html

RFC1305 David L. Mills. »Network Time Protocol (Version 3) – Specification, Implementation and Analysis«, Mârz 1992.
http://www.ietf.org/rfc/rfc1305.txt



9

Drucken unter Linux

Inhalt

9.1 Überblick 136
9.2 Kommandos zum Drucken 138
9.3 CUPS-Konfiguration 142
9.3.1 Grundlagen 142
9.3.2 Installation und Konfiguration eines CUPS-Servers 144
9.3.3 Tipps und Tricks 148

Lernziele

- Den grundlegenden Ablauf eines Druckvorgangs unter Linux verstehen
- Das Drucksystem CUPS kennen
- Die Anwenderkommandos zum Drucken benutzen können
- CUPS-Warteschlangen administrieren können
- Einen CUPS-Server mit lokalen, entfernten und/oder Netzwerkdruckern installieren und konfigurieren können

Vorkenntnisse

- Grundkenntnisse über Ein- und Ausgabe auf Shellebene
- Editieren von Dateien
- Benutzung eines Web-Browsers
- Grundlagen von TCP/IP (für Netzwerkdruck)

9.1 Überblick

Trotz aller Wunschträume vom »papierlosen Büro« wird die meiste Arbeit, die man mit dem Computer erledigt, doch früher oder später zu Papier gebracht. Gute Druckerunterstützung ist darum für ein modernes Betriebssystem wie Linux ein Muss – und historisch gesehen leider gar nicht selbstverständlich.

In einem Mehrbenutzersystem wie Linux ist das Drucken eine deutlich komplexere Aufgabe als in traditionellen »Einbenutzersystemen« wie DOS oder Windows (wo es nur dort komplizierter wird, wo Sie aus Kostengründen einen an Rechner *x* angeschlossenen Drucker auch für Benutzer auf den Rechnern *y*, *z*, … zugänglich machen möchten). Auf einem Linux-Rechner können mehrere angemeldete Benutzer gleichzeitig ein Dokument »drucken«, aber irgendwie muss dafür gesorgt werden, dass die Aufträge in eine Reihenfolge gebracht und nacheinander ausgedruckt werden. Prinzipiell spräche nichts dagegen, den Drucker zu öffnen wie eine Datei (das Gerät heißt typischerweise »`/dev/lp0`«) und Daten hineinzuschreiben, die der Linux-Kern dann an den tatsächlichen Drucker weiterleitet. Nur würde dann möglicherweise die Ausgabe mehrerer gleichzeitig druckender Prozesse vermischt. Außerdem würde dieser Ansatz die Anwendungsprogramme sehr eng an das verwendete Druckermodell und dessen Anschluss (parallele Schnittstelle) koppeln. Aus diesem Grund bevorzugt man die Vorgehensweise, eine Druckausgabe statt direkt an das Drucker-»Gerät« in ein Programm zu leiten, das die zu druckenden Daten zwischenspeichert, bis der Drucker »frei« ist. An dieser Stelle können Sie auch ansetzen und zum Beispiel PostScript-Daten, die das Programm produziert, für den Anwender transparent in druckerspezifische Steuercodes übersetzen. Ein solches Programm(system), das Druckerdaten zwischenspeichert, bis sie gedruckt werden können, und sie dann an den Drucker schickt, heißt **Spooler**[1].

Spooler

Warteschlange

Die verwendete Abstraktion ist die der **Warteschlange**; Druckaufträge werden nicht direkt an einen Drucker geschickt, sondern an eine Warteschlange, in der sie dann ihrer Weiterbearbeitung harren. Einem einzigen »physischen« Drucker können mehrere Warteschlangen zugeordnet werden, etwa eine für Aufträge auf Normal- und eine für Aufträge auf Hochglanzpapier. Genauso können die Aufträge einer Warteschlange an eine Warteschlange auf einem anderen Rechner weitergeleitet werden, die dann tatsächlich für den Ausdruck sorgt.

Berkeley-LPD

Das »traditionelle« Unix- bzw. Linux-Spoolersystem geht zurück auf BSD und wird als »**Berkeley-LPD**«, kurz für *line printer daemon*, bezeichnet. Berkeley-LPD gestattet die Ansteuerung von lokal, etwa parallel oder seriell, angeschlossenen Druckern sowie Druckern, die an andere Rechner angeschlossen sind.

LPD-Probleme

Berkeley-LPD eignet sich gut für die Ansteuerung einfacher Typenrad- oder Matrixdrucker (mit Einschränkungen auch Seitendrucker), ist aber den vielfältigen Möglichkeiten heutiger Drucktechnologie nicht wirklich gewachsen. Selbst einfache Drucker bieten heute zum Beispiel einseitigen oder doppelseitigen Druck, verschiedene Auflösungen, Farb- oder Schwarzweißdruck, Toner- oder Tintensparmodus und viele andere Optionen, die im Prinzip unabhängig voneinander ein- und ausgeschaltet werden können. Die einzige Möglichkeit, dies im Berkeley-LPD abzubilden, stellt die Definition verschiedener Warteschlangen dar, was in letzter Konsequenz schon für einen einzelnen Drucker zu einer kombinatorischen Explosion führt. Weitere Schwächen von Berkeley-LPD sind zum Beispiel, dass nur Administratoren Drucker konfigurieren können, dass es umständlich ist, alle Rechner in einem Netz einheitlich zu konfigurieren, und dass Berkeley-

[1] Volksetymologisch ist »SPOOL« die Abkürzung für »Simultaneous Peripheral Operation On-Line«, also etwa »Bedienung von Zusatzgeräten gleichzeitig zum eigentlichen Betrieb«. Dabei handelt es sich aber zumindest laut [Jargon] um ein *backronym*, also eine nachträgliche Interpretation eines Worts als Abkürzung.

LPD weder Druckerklassen (also mehrere Drucker, die aus derselben Warteschlange bedient werden) noch Abrechnungsmöglichkeiten für Aufträge kennt.

Beim Berkeley-LPD dient die Datei **/etc/printcap** zur Definition der auf einem Rechner verfügbaren Warteschlangen. /etc/printcap

Standard bei nahezu allen gängigen Linux-Distributionen ist heute das **Common Unix Printing System** oder **CUPS**. Hierbei handelt es sich um ein völlig neu konzipiertes Druckersystem, das zwar zum Berkeley-LPD kompatible Benutzerkommandos anbietet, ansonsten aber ganz anders funktioniert und verwaltet wird. Common Unix Printing System CUPS

CUPS basiert auf dem **Internet Printing Protocol**, standardisiert in [RFC2910, RFC2911], das das alte Berkeley-LPD-Protokoll [RFC1179] ablösen soll. IPP stellt in erster Linie standardisierte Operationen zur Verfügung, mit denen ein »Drucker« im weitesten Sinne – hierzu zählen nicht nur »echte« Drucker, sondern auch zum Beispiel Druckerserver, die einen oder mehrere nicht IPP-fähige Drucker ansteuern – nach seinen Fähigkeiten befragt und mit Aufträgen versorgt werden kann. Hierbei findet eine Art »Verhandlung« statt; ein Druckauftrag kann eine Wunschliste von Attributen wie »doppelseitig«, »tonersparend« und so weiter enthalten, der dann so weit wie möglich zu entsprechen versucht wird. Sie können dabei angeben, ob die Nichterfüllbarkeit eines Wunschs dazu führt, dass der Auftrag unbearbeitet bleibt, oder ob diejenige Fähigkeit des Druckers ausgenutzt wird, die dem Gewünschten am nächsten kommt. Internet Printing Protocol

IPP macht selbst keine Annahmen darüber, *wie*, soll heißen mit welcher Sorte Protokoll oder Verbindung, Sie mit einem Drucker reden wollen. Das kanonische Transportprotokoll, das für IPP vorgeschlagen wird, ist allerdings HTTP (siehe [RFC2910]). Dieser Ansatz hat diverse Vorteile:

- HTTP ist ein etabliertes und gut verstandenes Protokoll
- Diverse sehr wünschenswerte Eigenschaften wie Verschlüsselung (mit SSL), Proxy-Fähigkeit, Benutzerauthentisierung usw. bekommen Sie für IPP quasi geschenkt, da im HTTP-Umfeld hierfür Infrastruktur existiert
- Benutzer und Systemadministratoren können übliche Web-Browser verwenden, um mit einem IPP-Server (vulgo Drucker) zu kommunizieren. Web-Browser gibt es auf jeder nennenswerten Plattform angefangen von PDAs, so dass die Administration eines CUPS-basierten Drucksystems prinzipiell von fast überall aus möglich ist
- Für Druckeranbieter ist es einfach, IPP-Unterstützung anzubieten, da heutige *High-End*-Drucker oft schon eine Web-basierte Oberfläche für die Konfiguration unterstützen. Von hier zu einer IPP-Implementierung ist es ein ziemlich kleiner Schritt. Heutzutage gibt es auch zahlreiche frei verfügbare HTTP-Server, die sich als Grundlage für eine IPP-Implementierung in einem Drucker oder Druckerserver eignen.

IPP ist außerdem eine Entwicklung der Internet Engineering Task Force und genießt darum weitverbreitete Herstellerunterstützung. Es ist anzunehmen, dass IPP sich als betriebssystemübergreifende Standardlösung für Netzwerkdrucker durchsetzen wird. Sogar Microsoft Windows unterstützt in halbwegs aktuellen Versionen (Windows 2000 bzw. Windows ME und später) serienmäßig den Zugriff auf IPP-basierte Drucker; für ältere Windows-Versionen läßt die Unterstützung sich nachrüsten – das Paket heißt »Internet Printing Support for Windows« und ist auf der Microsoft-Website zu finden. IETF

CUPS ist eine Implementierung von IPP für diverse Unix-Versionen (darunter Linux), die von der US-amerikanischen Firma *Easy Software Products* entwickelt wurde. CUPS steht unter der GPL.

ESP wurde inzwischen von Apple aufgekauft – ein durchaus nicht unvernünftiger Schritt, da auch MacOS X CUPS zur Druckeransteuerung verwendet. An der Freiheit des Codes wurde bisher allerdings nicht gerüttelt.

Herzstück von CUPS ist ein HTTP-Server (bei CUPS *scheduler* genannt), der nicht nur IPP-Anfragen entgegennimmt und bearbeitet, sondern auch die Online-Dokumentation ausliefert und Statusabfragen für Drucker und Aufträge ermöglicht. Ferner stellt er eine Liste der erreichbaren Drucker zur Verfügung und hält diese aktuell. Zugriff auf den HTTP-Server ist außer über direktes HTTP auch über eine Programmierschnittstelle, das »CUPS-API«, möglich, und CUPS verwendet diese, um Benutzerkommandos zu realisieren, die denen von BSD (`lpr`, `lpq`, `lprm`, …) oder denen von System-V-Unix (`lp`, `lpstat`, …) entsprechen. Außerdem gehören zu CUPS noch diverse **Filter** für unterschiedliche Eingabedatenformate sowie einige **Backends**, um Drucker über verschiedene Zugriffsmedien wie die parallele oder serielle Schnittstelle, USB, SMB, AppDirect (etwa für HP-JetDirect-fähige Drucker) oder LPR anzusprechen. Der Benutzer sieht in jedem Fall nur den von CUPS implementierten IPP-fähigen »Drucker«.

Filter

Backends

9.2 Kommandos zum Drucken

»Direktes« Drucken

Dateien drucken: `lpr`, `lp` und `mpage` Prinzipiell können Sie Daten drucken, indem Sie sie einfach an die Schnittstelle schicken, an die der Drucker angeschlossen ist:

```
# cat daten.txt >/dev/lp0
```

Je nach den eingestellten Zugriffsrechten müssen Sie dafür natürlich Administratorprivilegien haben. So ein Zugriff setzt aber voraus, dass niemand sonst im selben Moment etwas auf dieselbe Art druckt, und außerdem müssen die Daten in einem Format sein, die der Drucker versteht. Teurere Drucker erwarten zum Beispiel Eingaben im PostScript-Format (wenn nicht gar irgendeiner proprietären Kodierung) und können mit reinem Text möglicherweise nichts anfangen. Wir erwähnen diese Methode trotzdem, da sie sich zur Fehlersuche eignet – wenn Sie Ihren Drucker auf diese Weise zum Drucken bringen, dann sind zumindest Hardware, Verkabelung und andere grundlegende Sachen in Ordnung und Sie können die Probleme auf der Softwareseite suchen.

Fehlersuche

Systemprogramme

Im täglichen Leben sollten Sie besser eines der Systemprogramme zum Drucken verwenden, die die Schnittstelle nicht direkt ansprechen, sondern sich an das installierte Drucksystem (etwa CUPS) wenden. So erreichen Sie eine saubere Nacheinanderbearbeitung von Aufträgen. Auch eine Umwandlung der Daten in ein für den Drucker adäquates Format lässt sich arrangieren.

`lpr`

Das gängigste Programm zum Drucken ist `lpr`. Sie können damit entweder die Standardausgabe eines anderen Programms oder das Ergebnis einer Pipeline drucken:

```
$ pr -l50 handbuch.txt | lpr
```

Oder Sie übergeben dem Programm eine Liste von zu druckenden Dateien:

```
$ lpr datei1.txt datei2.ps
```

Beim `lpr`-Aufruf können Sie verschiedene Optionen angeben: Mit

```
$ lpr -#3 -Plaser datei1.txt
```

werden zum Beispiel drei Exemplare der Datei `datei1.txt` auf dem Drucker `laser` gedruckt (genau genommen in die Warteschlange `laser` eingestellt).

Warteschlange wählen

Wenn Sie nichts anderes sagen, stellt `lpr` die Aufträge in die Warteschlange `lp`.

(*Vorsicht:* Manche Distributionen, etwa die von SUSE, erlauben es dem Administrator, das systemweit zu ändern.) Wenn Sie nicht explizit bei jedem Auftrag die Warteschlange mit `-P` wählen wollen, können Sie die Umgebungsvariable `PRINTER` auf den Namen der gewünschten Warteschlange setzen.

Neben dem `lpr`-Kommando, das aus der Tradition von Berkeley-LPD stammt, unterstützen viele Systeme ein fast äquivalentes Programm namens `lp`. Dies kommt aus der System-V-Unix-Tradition und ist vielen Benutzern proprietärer Unix-Systeme geläufig. Dem letzten `lpr`-Beispiel oben entspricht das `lp`-Kommando

`lp`

```
$ lp -n 3 -d color datei.txt
```

Das CUPS-Kommando `lp` erlaubt einige Dinge, die mit `lpr` nicht gehen, etwa das nachträgliche Modifizieren von Aufträgen nach dem Abschicken (solange sie nicht gedruckt worden sind). Damit können Druckaufträge zum Beispiel zeitversetzt gedruckt oder auf unbestimmte Zeit »auf Eis gelegt« und später wieder freigegeben werden.

Die CUPS-Versionen von `lpr` und `lp` unterstützen diverse Optionen, die die Druckausgabe beeinflussen können. Zum Beispiel können Sie doppelseitigen Druck über

```
$ lpr -o sides=two-sides-long-edge manual.pdf
```

oder zwei verkleinerte Seiten auf einer über

```
$ lpr -o number-up=2 manual.pdf
```

anfordern. Etwas wie

```
$ lpr -o landscape sign.ps
```

druckt im Querformat.

Welche Optionen im konkreten Fall zulässig sind, hängt von den Fähigkeiten des angesteuerten Druckers ab, aber hier sind einige gängige CUPS-Optionen aufgeführt:

`media=`⟨*typ*⟩ Bestimmt die Papiergröße und -quelle. Gültige Werte für die Papiergröße sind zum Beispiel `A4` oder `Letter`, als Papierquelle könnten beispielsweise die Papierfachnamen `Upper` und `Lower` angegeben werden. Die genauen Werte für bestimmte Drucker ergeben sich aus den entsprechenden PPD-Dateien.

`landscape` Druckt im Querformat.

`sides={one-sided,two-sided-short-edge,two-sided-long-edge}` Druckt beidseitig; `two-sided-short-edge` ist sinnvoll für Seiten im Querformat und `two-sided-long-edge` für Seiten im Hochformat. `one-sided` sorgt für einseitigen Druck bei Warteschlangen, die standardmäßig doppelseitig drucken.

`jobsheets=`⟨*anfang*⟩**`[,`**⟨*ende*⟩**`]`** Bestimmt, ob am Anfang und Ende des Auftrags eine »Banner«-Seite ausgedruckt wird. Diese Seite enthält Informationen über den Einreicher des Auftrags, Datum, Uhrzeit und z. B. eine etwaige Vertraulichkeitsstufe. Welche Banner-Seiten tatsächlich zur Verfügung stehen, hängt vom lokalen System ab; standardmäßig sind die Optionen `none` (keine Banner-Seite), `standard` (keine Vertraulichkeitsangabe) und verschiedene Seiten à la `unclassified`, `confidential`, `topsecret` definiert.

`page-ranges=`⟨*liste*⟩ Druckt nur eine Teilmenge der Seiten des Auftrags. Die ⟨*liste*⟩ ist eine durch Kommata getrennte Folge von Seitenzahlen oder Bereichen von Seitenzahlen, etwa wie »`1-4,7,9-12`«.

page-set={even,odd} Druckt nur die geraden (even) oder ungeraden (odd) Seiten.

outputorder={normal,reverse} Druckt die Seiten des Auftrags in normaler bzw. der umgekehrter (reverse) Reihenfolge.

number-up={1,2,4,6,9,16} Druckt 1, 2, 4, 6, … Seiten des Auftrags verkleinert auf einer Druckseite.

page-border={none,single,single-thick,double,double-thick} Beim Mehrseitendruck wird um jede verkleinerte Seite kein Rahmen (none), ein einfacher oder doppelter dünner oder dicker (1pt) Rand gezogen.

number-up-layout={btlr,btrl,lrbt,lrtb,rlbt,rltb,tblr,tbrl} Die Anordnung der Seiten im Mehrseitendruck: btlr steht für »*bottom to top, left to right*«, rltb für »*right to left, top to bottom*« und so weiter.

prettyprint (Für Textdruck.) Gibt auf jeder Seite eine Kopfzeile mit einer Seitennummer, dem Auftragsnamen (üblicherweise dem Namen der zu druckenden Datei) und dem Datum aus. Außerdem wird bei C- und C++-Programmen Syntaxhervorhebung gemacht, und Kommentarzeilen werden kursiv gedruckt.

Eine ausführliche Liste der Optionen finden Sie in der Dokumentation, die ein CUPS-Server Ihnen per HTTP auf Port 631 anbietet, auf dem Rechner mit dem CUPS-Server selbst etwa unter http://localhost:631/help/options.html?TOPIC=Getting+Started.

Verfolgen von Aufträgen Auch als Benutzer möchten Sie sich möglicherweise ein Bild vom Zustand der Druckerwarteschlangen machen: Lohnt es sich schon, aufzustehen und zum Druckerraum am anderen Ende des Flurs zu gehen, oder druckt Herr Schmidt aus der Buchhaltung wieder seinen fünfhundertseitigen Bericht für die Geschäftsleitung, während Ihr Brief dahinter Schlange steht?

Hierfür ist das Kommando lpq nützlich: Ohne Parameter aufgerufen liefert es den Inhalt der Standard-Warteschlange:

```
$ lpq
lp is ready and printing
Rank   Owner   Job  Files            Total Size
active hugo    333  bericht.ps       1112942 bytes
```

Wie bei lpr können Sie auch hier mit -P den Namen einer anderen Warteschlange angeben, ansonsten gilt der Wert der Umgebungsvariablen PRINTER oder lp.

lpq mit der Option -a zeigt die Aufträge in allen Warteschlangen, und -l gibt eine »lange« Liste (mit mehr Informationen) aus. Ist das ⟨*Intervall*⟩ angegeben, so wird die Liste alle ⟨*Intervall*⟩ Sekunden neu ausgegeben, bis die Warteschlange leer ist.

Das Programm lpstat arbeitet deutlich anders. Hier gibt es Optionen, die angeben, welche Art von Status ausgegeben werden soll:

-a Zeigt an, ob die Warteschlangen Aufträge akzeptieren

-c Zeigt Druckerklassen und dazugehörige Drucker an

-d Zeigt den aktuellen Standarddrucker an

-o [⟨*Warteschlange*⟩] Zeigt den Inhalt der ⟨*Warteschlange*⟩ an (oder aller Warteschlangen, wenn keine angegeben wurde)

-p Zeigt die Drucker (Warteschlangen) an und ob sie gerade zum Drucken freigeschaltet sind

-r Zeigt an, ob der CUPS-Server läuft

-s Zeigt eine Status-Zusammenfassung (äquivalent zu »`lpstat -dcp`«)

-t Zeigt alle Statusinformationen (äquivalent zu »`lpstat -rdcvapo`«)

-v Zeigt die Drucker (Warteschlangen) und die zugehörigen Anschlußarten an

(auch hier ist nur ein Ausschnitt der kompletten Optionenliste angegeben).

Stornieren von Aufträgen Wenn Sie es sich nach dem Einreichen eines Druckauftrags noch anders überlegen, können Sie den Auftrag mit `lprm` stornieren. Dazu brauchen Sie die Auftragsnummer, die `lpq` in der Spalte »`Job`« ausgibt:

```
$ lprm 333
```

Da die Auftragsnummern für jede Warteschlange getrennt vergeben werden, müssen Sie gegebenenfalls auch hier mit `-P` die Warteschlange angeben, die Sie meinen. Alle Ihre ausstehenden Druckaufträge werden Sie mit

```
$ lprm -
```

los.

Fremde Aufträge stornieren

Als normaler Benutzer können Sie nur Ihre eigenen Druckaufträge stornieren (Herr Schmidt wäre Ihnen böse). Um die Druckaufträge anderer Benutzer mit `lprm` stornieren zu können, müssen Sie Administrator-Rechte in Anspruch nehmen.

Einschränkungen

Sie können mit `lprm` nur Aufträge stornieren, die noch nicht an den Drucker geschickt wurden. Heutige Drucker haben meist interne Speicher, in die ein Auftrag oder mehr hineinpassen, selbst wenn sie noch gar nicht an der Reihe sind, und auf solche Aufträge kann Ihr Rechner keinen Einfluss mehr nehmen. Ob ein großer Auftrag storniert werden kann, der gerade halb an den Drucker geschickt ist, ist systemabhängig.

Das System-V-artige Programm `cancel` verlangt als Argument eine Kombination aus Warteschlangennamen und Auftragsnummer:

```
cancel ⟨Warteschlange⟩-⟨Auftragsnummer⟩
```

Standardwerte für Druckoptionen Die Druckoptionen, die Sie mit `-o` bei `lpr` und `lp` angeben können, haben gewisse systemweite Standardwerte, die bei der Installation der betreffenden Warteschlange in Kraft treten. Später ist es möglich, diese Standardwerte entweder als Systemverwalter für alle Benutzer oder als Benutzer für sich selbst umzudefinieren. Es bleibt einem Benutzer natürlich unbenommen, über Kommandooptionen für einen Auftrag Werte der Optionen einzustellen, die von den Vorgabewerten abweichen.

Zum Einstellen von Druckoptionen dient das Kommando `lpoptions`, das dieselben `-o`-Optionen akzeptiert wie `lpr` und `lp`. Diese Optionen werden für alle Warteschlangen oder, wenn mit `-p` eine Warteschlange angegeben wurde, nur für diese Warteschlange in Kraft gesetzt. `lpoptions` trägt die angegebenen Optionen in die Datei `~/.lpoptions` ein, wo sie später als Standardwerte für `lpr` und `lp` angenommen werden. »`lpoptions -l`« gibt die Optionsnamen, die möglichen Werte und die aktuellen Standardwerte (durch einen Stern gekennzeichnet) aus.

Ruft der Systemverwalter `lpoptions` als Benutzer `root` auf, gelten die Einstellungen für die systemweite Voreinstellung für alle Benutzer. Für `root` gibt es keine Druckvoreinstellungen.

```
application/pdf          pdf string(0,%PDF)
application/postscript   ai eps ps string(0,%!) string(0,<04>%!) \
                         contains(0,128,<1B>%-12345X) + \
                         (contains(0,1024,"LANGUAGE=POSTSCRIPT") \
                          contains(0,1024,"LANGUAGE = Postscript") \
                          contains(0,1024,"LANGUAGE = POSTSCRIPT"))
image/gif                gif string(0,GIF87a) string(0,GIF89a)
image/png                png string(0,<89>PNG)
image/jpeg               jpeg jpg jpe string(0,<FFD8FF>) &&\
                         (char(3,0xe0) char(3,0xe1) char(3,0xe2) char(3,0xe3)\
                          char(3,0xe4) char(3,0xe5) char(3,0xe6) char(3,0xe7)\
                          char(3,0xe8) char(3,0xe9) char(3,0xea) char(3,0xeb)\
                          char(3,0xec) char(3,0xed) char(3,0xee) char(3,0xef))
image/tiff               tiff tif string(0,MM) string(0,II)
```

Bild 9.1: Die Datei `mime.types` (Auszug)

Übungen

9.1 [2] Probieren Sie einige Druckoptionen von CUPS aus: Versuchen Sie zum Beispiel, einen Auftrag mehrfach verkleinert, rückwärts oder beidseitig auszudrucken oder bestimmte Seitenbereiche zu selektieren.

9.2 [2] Stellen Sie als »normaler« Benutzer die Warteschlange so ein, dass normalerweise immer zwei Seiten eines Auftrags verkleinert und auf einer quer genommenen Seite nebeneinander gedruckt werden. Prüfen Sie, dass Aufträge ohne besondere Optionen tatsächlich so bearbeitet werden. Drucken Sie dann einen Auftrag unverkleinert, ohne die Voreinstellung dauerhaft zu ändern.

9.3 CUPS-Konfiguration

9.3.1 Grundlagen

Druckauftrag

Wenn ein Benutzer unter CUPS einen Druckauftrag einreicht – entweder über eins der BSD- oder System-V-kompatiblen Programme, die CUPS mitliefert, oder ein Programm, das das CUPS-API direkt unterstützt wie etwa `kprinter` aus KDE –, wird der Auftrag zunächst in der gewünschten Warteschlange gespeichert. Neben den eigentlichen zu druckenden Daten gehören dazu auch »Metadaten« wie der Benutzername des Einreichers oder die gewünschten Optionen für den Ausdruck (doppelseitig, …). Danach entnimmt der Scheduler den Auftrag der Warteschlange und versucht zunächst, die Druckdaten ins PostScript-Format zu bringen. Hierzu konsultiert er die Datei `/etc/cups/mime.types` (siehe Bild 9.1), um erst einmal den aktuellen Datentyp zu bestimmen. `mime.types` enthält die Namen von MIME-Typen und zu jedem MIME-Typ eine Reihe von Kriterien, anhand derer dieser Typ erkannt werden kann. Beispielsweise wird durch die Regel

Warteschlange

```
image/gif    gif string(0,GIF87a) string(0,GIF89a)
```

eine Datei als GIF-Graphik identifiziert, wenn ihr Name auf »`gif`« endet oder der Inhalt mit den Zeichenketten »`GIF87a`« oder »`GIF89a`« anfängt. (Die genauen Regeln für die Kriterien sind am Anfang der `mime.types`-Datei angegeben).

Wenn der MIME-Typ der zu druckenden Datei bekannt ist, kann anschließend daran gegangen werden, die Daten in ein von CUPS druckbares Format zu bringen, namentlich `application/vnd.cups-postscript`. Dazu stehen verschiedene

```
application/pdf         application/postscript  33      pdftops
application/postscript  application/vnd.cups-postscript 66      pstops
application/vnd.hp-HPGL application/postscript  66      hpgltops
application/x-shell     application/postscript  33      texttops
text/plain              application/postscript  33      texttops
text/html               application/postscript  33      texttops
image/gif               application/vnd.cups-postscript 66      imagetops
image/png               application/vnd.cups-postscript 66      imagetops
image/jpeg              application/vnd.cups-postscript 66      imagetops
```

Bild 9.2: Die Datei `/etc/cups/mime.convs` (Auszug)

Filterprogramme zur Verfügung, deren Einsatz in der Datei `/etc/cups/mime.convs` (Bild 9.2) beschrieben wird. Dort werden verschiedene Programme (rechts auf der Zeile) aufgezählt, die in der Lage sind, Daten vom in der linken Spalte benannten Typ in den in der zweiten Spalte benannten zu konvertieren. Jeder Konvertierung ist ein »Kostenfaktor« zugeordnet, über den direkte Konvertierungen solchen auf »Umwegen« vorgezogen werden können. Beispielsweise können HPGL-Daten auf dem Weg über `application/postscript` ins CUPS-PostScript-Format gebracht werden, wobei insgesamt »Kosten« von 99 Einheiten anfallen; gäbe es einen direkten Konverter, so würde dieser bevorzugt, wenn ihm ein geringerer Kostenfaktor als 99 zugeordnet wäre.

Filterprogramme

Die meisten Formate werden zunächst ins »unabhängige« PostScript übersetzt, während das `pstops`-Programm sie dann ins CUPS-spezifische Format bringt. Dies ist ein durchaus wichtiger Schritt, da bei dieser Gelegenheit beispielsweise die Anzahl der Seiten im Auftrag bestimmt und protokolliert wird (die Sie einer PostScript-Datei ja nicht im allgemeinen ansehen können, ohne das PostScript zumindest ansatzweise auszuführen). Außerdem leistet `pstops` noch ein paar andere Extras, die sehr nützlich sind und vor CUPS bei Unix-Drucksystemen durchaus nicht selbstverständlich waren – zum Beispiel können mehrere Seiten verkleinert auf einer einzigen Druckseite ausgegeben (`ps-n-up`) oder nur bestimmte Seiten des Auftrags tatsächlich gedruckt werden (`psselect`), ohne dass das druckende Anwendungsprogramm dies selbst unterstützen muss.

`pstops`

Extras

Das CUPS-spezifische PostScript wird dann entweder direkt weitergeleitet (wenn der Auftrag letztendlich von einem PostScript-fähigen Drucker ausgegeben wird) oder mit GhostScript in eine druckerspezifische Sprache wie PCL oder ESC/P umgewandelt. Die tatsächliche Ausgabe an den Drucker (oder Weiterleitung an einen anderen Druckerserver) übernimmt eines der »Backends« in `/usr/lib/cups/backend`.

PCL

ESC/P

Wichtigstes Element der CUPS-Konfiguration für einen bestimmten Drucker sind **PPD-Dateien** (kurz für »PostScript Printer Description«). Die PPD-Datei für einen Drucker gibt an, welche besonderen Optionen (Auflösung, Duplexdruck, verschiedene Papierzufuhrmöglichkeiten, …) der Drucker unterstützt, und CUPS kann diese Informationen an Anwenderprogramme weitergeben, die dem Benutzer dann komfortablen Zugriff auf die besonderen Fähigkeiten jedes Druckers ermöglichen. Die PPD-Datei für einen PostScript-Drucker sollte vom Druckerhersteller mitgeliefert oder über das WWW zur Verfügung gestellt werden; für viele – auch Nicht-PostScript- – Drucker finden sich PPD-Dateien neben diversen anderen Tips unter `http://www.linuxprinting.org`. PPD-Dateien sind nützlich für PostScript- genau wie für Nicht-PostScript-Drucker und darum auch für die allermeisten Drucker irgendwo aufzutreiben. Die Chancen stehen gut, dass schon entweder CUPS oder die Linux-Distribution PPD-Dateien für alle bis auf verschiedene ausgefallene Druckermodelle enthalten; gegebenenfalls ist hierfür ein Zusatzpaket der Distribution zu installieren.

PPD-Dateien

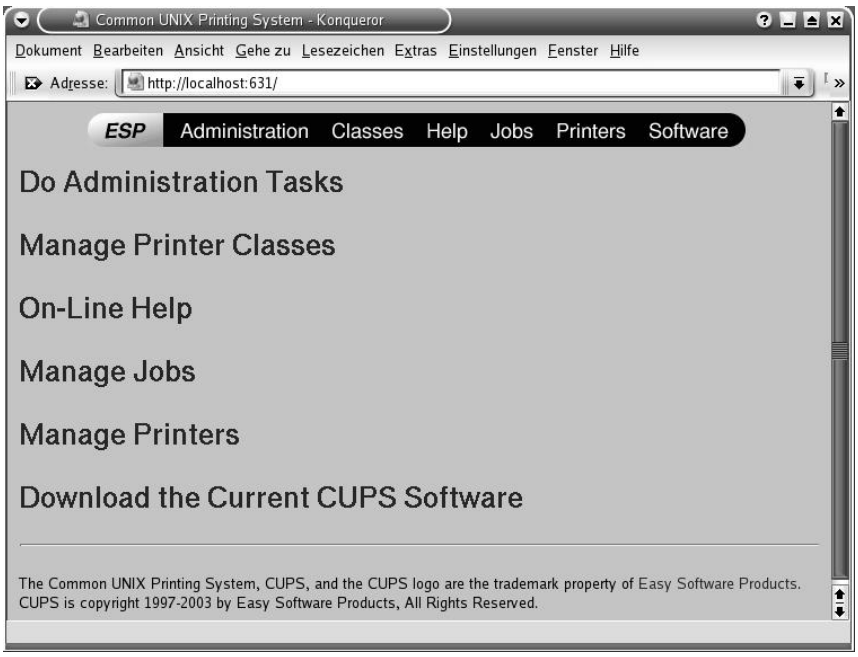


Bild 9.3: Die CUPS-Web-Oberfläche

Übungen

9.3 [2] Schauen Sie auf `http://www.linuxprinting.org` nach, ob Ihr Drucker von Linux unterstützt wird, und rufen Sie ggf. eine PPD-Datei für ihn ab.

9.3.2 Installation und Konfiguration eines CUPS-Servers

CUPS ist heute Bestandteil aller wesentlichen Linux-Distributionen oder kann zumindest leicht nachinstalliert werden.

Die Novell/SUSE-Distributionen verwenden schon seit geraumer Zeit CUPS als Standard-Drucksystem.

CUPS steht in den Paketen `cups` (Server), `cups-bsd` (LPD-artige Kommandos) und `cups-client` (System-V-artige Kommandos) zur Verfügung. Darüber hinaus gibt es noch eine ganze Reihe weiterer Unterstützungspakete (probieren Sie mal »`apt-cache search cups`«).

CUPS bietet verschiedene Möglichkeiten zur Konfiguration von Druckern an. Der CUPS-interne Web-Server erlaubt die Einrichtung von Druckerwarteschlangen mit einem WWW-Browser, und es gibt auch Kommandozeilenwerkzeuge dafür. Die Novell/SUSE-Distributionen gestatten die Konfiguration von Druckern mit dem SUSE-eigenen Administrationswerkzeug YaST.

Konfiguration mit einem Web-Browser Zur Konfiguration von Warteschlangen mit einem Web-Browser rufen Sie auf dem CUPS-Rechner den URL `http://localhost:631/` auf (631 ist der dem IPP zugeordnete TCP-Port). Dort sollte sich die CUPS-Bedienungsoberfläche melden (Bild 9.3). – Prinzipiell ist der Zugriff von jedem Rechner aus möglich, allerdings werden im Auslieferungszustand von CUPS nur IPP-Anfragen vom selben Rechner aus akzeptiert. Über die *Printers*-Schaltfläche gelangen Sie in die Druckerverwaltung (Bild 9.4 zeigt ein Beispiel, wo schon zwei Drucker installiert sind) und über die dortige Schaltfläche *Add Printer* in den Dialog zum Hinzufügen eines neuen Druckers (Bild 9.5).

Hier ist zunächst ein Name für den Drucker (eigentlich die Warteschlange) einzugeben; die beiden Felder *Location* und *Description* sind technisch nicht nötig, aber helfen den Benutzern, mit dem Warteschlangennamen einen tatsächlichen

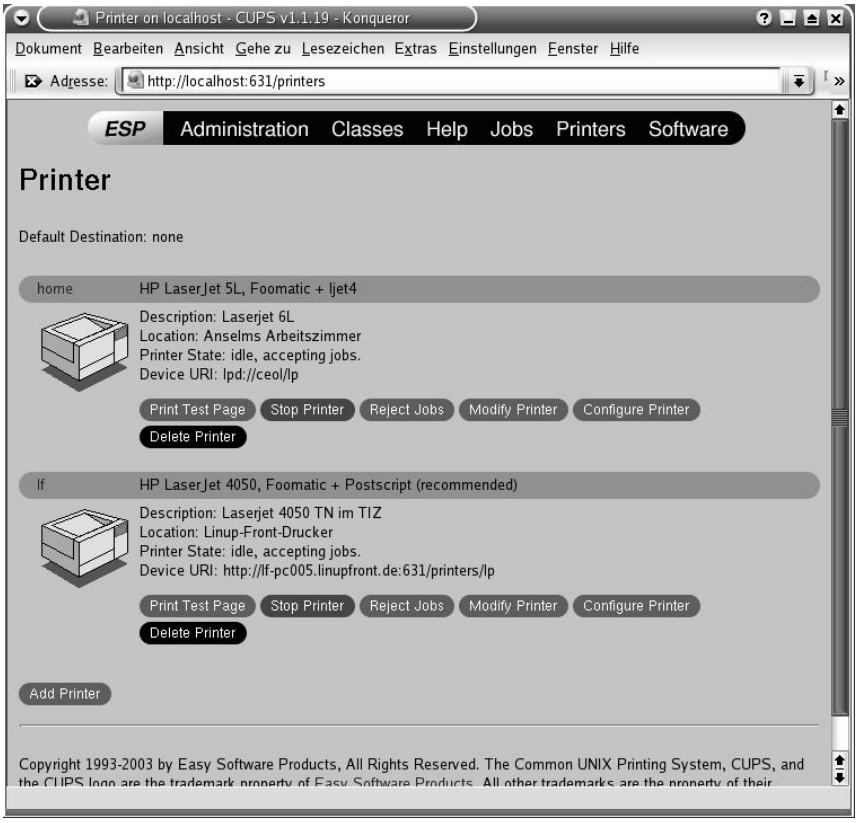


Bild 9.4: Die CUPS-Web-Oberfläche: Druckerverwaltung

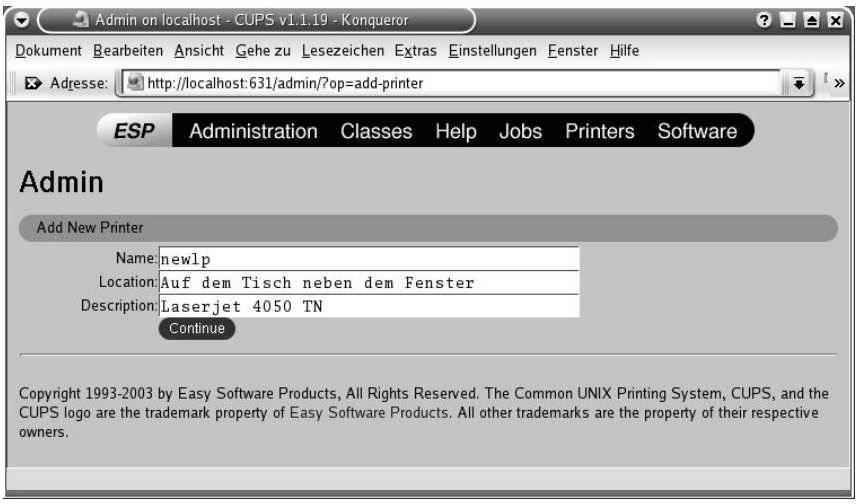


Bild 9.5: Die CUPS-Web-Oberfläche: Drucker hinzufügen

Drucker zu verbinden (sie müssen ja wissen, wo sie ihre Ausdrucke einsammeln können).

Der nächste Dialog dient der Auswahl einer Anschlussmethode (*backend*). Hier stehen Optionen zur Verfügung wie die parallelen oder seriellen Schnittstellen, USB oder verschiedene Netzwerkmöglichkeiten wie LPD, IPP, JetDirect oder Windows-Drucker (über Samba). Je nachdem, was Sie hier auswählen, muss als nächstes ein URL für den Drucker (zum Beispiel `lpd://lpdserver/lp` oder `ipp://ippserver/printers/myprinter`) angegeben werden, der bestimmt, wie der Drucker tatsächlich angesprochen wird; bei direkt an Schnittstellen oder den USB angeschlossenen Druckern erübrigt sich das natürlich.

Zum Schluss müssen Sie sich noch für ein konkretes Druckermodell entscheiden. CUPS bietet hier die PPD-Dateien an, die unter `/usr/share/cups/model` gespeichert sind – bei der GPL-Version sind das leider nur ein paar für die gängigsten Drucker, aber einerseits haben Hersteller wie SUSE die Auswahl sehr aufgepeppt, und andererseits können Sie sich ja eine passende PPD-Datei von `www.linuxprinting.org` besorgen und dort ablegen. Nach der letzten Bestätigung steht der Drucker dann unter *Printers* in der Oberfläche zur Verfügung und kann verwendet werden. Am besten beginnen Sie mit dem Ausdrucken einer Testseite (»*Print Test Page*«), um zu prüfen, ob die Einrichtung tatsächlich geklappt hat.

Das Administrationswerkzeug YaST2 der Novell/SUSE-Distributionen erlaubt eine ähnlich komfortable Druckerkonfiguration wie die CUPS-eigene Web-Oberfläche. Im Gegensatz zum serienmäßigen CUPS enthält der YaST eine wesentlich umfangreichere Datenbank von PPD-Dateien und nimmt auf Wunsch auch eine automatische Druckererkennung vor. Nach einer erfolgreichen Druckererkennung erlaubt YaST wahlweise ein »schnelles automatisches Setup«, bei dem ggf. mehrere Warteschlangen, etwa für Farb- und Schwarzweißdruck, eingerichtet werden, und ein »normales Setup«, bei dem Warteschlangennamen, Druckerort und -typ und ähnliches manuell eingetragen werden. Im normalen Setup können Sie auch Optionen für die Warteschlange vorgeben wie die Papiergröße und Auflösung, etwaige Anfangs- und Schlussseiten oder Zugriffsbeschränkungen für bestimmte Benutzer, die über die CUPS-eigene Web-Oberfläche nicht direkt zugänglich sind, sondern über Kommandozeilenwerkzeuge oder Konfigurationsdateien vereinbart werden müssen.

Zu guter Letzt ist es auch möglich, neue Drucker über das Kommandozeilenwerkzeug `lpadmin` einzurichten. `lpadmin` ist an das gleichnamige System-V-Werkzeug angelehnt, aber nicht vollständig kompatibel. Einen parallel angeschlossenen Laserjet-Drucker könnten Sie beispielsweise wie folgt konfigurieren:

```
# lpadmin -p newlp -E -v parallel:/dev/lp0 -m laserjet.ppd
```

Dabei ist `newlp` der Name der neuen Warteschlange, die Option `-E` schaltet die Warteschlange für Aufträge frei, `-v` gibt die Anschlussart des Druckers an, und `-m` benennt die PPD-Datei für diesen Drucker (muss in `/usr/share/cups/model` stehen). Mit den Optionen `-D` und `-L` können eine Beschreibung und Ortsangabe wie in der Web-Oberfläche angegeben werden. Weitere Informationen befinden sich in `lpadmin`(8).

Das Kommando `lpinfo` kann zur Druckerinstallation nützlich sein. »`lpinfo -v`« gibt eine Liste der verfügbaren Anschlussmöglichkeiten aus und »`lpinfo -m`« eine Liste der vorhandenen PPD-Dateien (und damit Druckertypen).

Mit »`lpadmin -p`« lassen sich auch bestehende Druckerkonfigurationen modifizieren. »`lpadmin -x`« löscht einen nicht mehr benötigten Drucker.

In jedem Fall werden die Informationen über installierte Drucker in der Datei `/etc/cups/printers.conf` abgelegt. Für jeden Drucker steht eine PPD-Datei in `/etc/cups/ppd` (Bild 9.6).

```
# Printer configuration file for CUPS v1.1.19
# Written by cupsd on Thu Jul 31 23:51:00 2003
<DefaultPrinter newlp>
Info Laserjet 4050 TN
Location Auf dem Tisch neben dem Fenster
DeviceURI lpd://localhost/lp
State Idle
Accepting Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
</Printer>
<Printer home>
Info Laserjet 6L
◁◁◁◁◁
```

Bild 9.6: Datei `/etc/cups/printers.conf` (Auszug)

Allgemeine Konfigurationseinstellungen für den CUPS-Scheduler stehen in der Datei `/etc/cups/cupsd.conf`. Die Syntax dieser Datei ist an die des Apache-Web-Servers angenähert. Insbesondere wichtig sind hier die Einstellungen für Zugriffsrechte und Authentisierung. Standardmäßig ist der Zugriff auf den CUPS-Scheduler nur vom lokalen Rechner (der IP-Adresse `127.0.0.1`) aus möglich. Für den Einsatz von CUPS als Druckerserver in einem lokalen Netz muss diese Restriktion gelockert werden. Dies kann durch Definitionen der Form

```
<Location /printers>
  Order Deny,Allow
  Deny from all
  Allow from 127.0.0.1
  Allow from 192.168.123.0/24
</Location>
```

oder

```
<Location /printers/newlp>
  Order Allow,Deny
  Allow from 127.0.0.1
  Allow from 192.168.123.0/24
  Deny from 192.168.123.45
</Location>
```

geschehen. Dabei beschränkt das erste Beispiel den Zugriff auf alle Drucker auf den lokalen Rechner und die Rechner im Netz `192.168.123.0/24`; im zweiten Beispiel wird nur der Zugriff auf den Drucker `newlp` für den lokalen Rechner und die Rechner im lokalen Netz freigegeben, bis darauf, dass der Rechner `192.168.123.45` ausgesperrt bleibt. Auf die gleiche Weise lässt sich auch der Zugriff auf die Administrationsfunktionalität der CUPS-Web-Oberfläche steuern.

Neben einer Authentisierung auf der Basis der IP-Adresse ist auch eine benutzerbasierte Authentisierung über Benutzernamen und Kennwörter möglich. Ebenso ist es möglich, SSL bzw. TLS zur Sicherung von IPP-Anfragen einzusetzen, mit einer Authentisierung des CUPS-Servers und ggf. auch des Benutzers über X.509-Zertifikate. In letzter Konsequenz folgt daraus die Möglichkeit, CUPS als sicheren, effizienten »Fax«-Dienst über das Internet einzusetzen.

Übungen

9.4 [1] Welche verschiedenen Backends bietet Ihre CUPS-Implementierung an?

9.5 [2] Installieren Sie einen Drucker. Wenn an Ihren Rechner kein Drucker direkt angeschlossen ist, dann geben Sie als Ziel für die Druckaufträge einen Netzdrucker an (Ihr Trainer gibt Ihnen gegebenenfalls die nötigen Informationen.)

9.6 [2] Geben Sie die Warteschlange Ihres Druckers für die anderen Rechner im lokalen Netz frei. Vergewissern Sie sich, dass die Rechner Druckaufträge einreichen können und dass diese gedruckt werden.

9.7 [2] Warum kann es sinnvoll sein, für denselben Drucker mehrere Warteschlangen einzurichten?

9.3.3 Tipps und Tricks

T-Shirts und ähnliches Um aus beliebigen Anwendungsprogrammen heraus Bilder spiegelverkehrt zu drucken – etwa auf T-Shirt-Transferpapier –, kann die Druckeroption `mirror` benutzt werden:

```
$ lpr -o mirror mypic.jpg
```

Fehlersuche Wenn das Drucken mit CUPS nicht so funktioniert, wie es soll, sollten Sie zunächst das CUPS-Protokoll studieren. CUPS protokolliert Details seiner Tätigkeit in die Datei `/var/log/cups/error_log` (Ort der Datei kann von Distribution zu Distribution variieren). Der Umfang des Protokolls wird von der `LogLevel`-Direktive in der Datei `/etc/cups/cupsd.conf` bestimmt; der Wert `debug2` schreibt das ausführlichste Protokoll.

Kommandos in diesem Kapitel

cancel	Storniert einen Druckauftrag	cancel(1)	141
lp	Reicht einen Druckauftrag ein	lp(1)	139
lpadmin	Dient zur Verwaltung von Druckerwarteschlangen	lpadmin(8)	146
lpinfo	Zeigt verfügbare Drucker-Geräte und -Treiber an	lpinfo(8)	146
lpoptions	Verwaltet Standardeinstellungen für Druckerwarteschlangen	lpoptions(1)	141
lpq	Gibt den Status einer Druckerwarteschlange aus	lpq(1)	140
lpr	Reicht einen Druckauftrag ein	lpr(1)	138
lprm	Storniert einen Druckauftrag	lprm(1)	141
pstops	Bereitet PostScript-Druckaufträge für CUPS auf		143

Zusammenfassung

- Drucken unter Linux ist eine komplexe Aufgabe.
- CUPS ist eine Implementierung des engl. *Internet Printing Protocol*, eines HTTP-basierten Industriestandards für den Umgang mit Netzwerkdruckern.
- CUPS erlaubt die Anpassung des Systems an gegebene Drucker über PPD-Dateien. Es unterstützt Filterprogramme für diverse Datenformate und kann mit verschiedenen Anschlussmethoden für Drucker umgehen.
- Die Konfiguration von Druckern für CUPS ist über den CUPS-eigenen Web-Server, die Kommandozeile oder Distributionswerkzeuge wie den »YaST« bei den SUSE-Distributionen möglich.

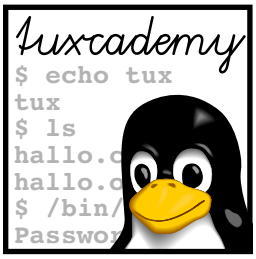
Literaturverzeichnis

Jargon Eric S. Raymond. »The Jargon File«. Auch veröffentlicht als *The Hacker's Dictionary*. `http://www.jargon.org/`

RFC1179 L. McLaughlin III. »Line Printer Daemon Protocol«, August 1990. `http://www.ietf.org/rfc/rfc1179.txt`

RFC2910 R. Herriot, S. Butler, P. Moore, et al. »Internet Printing Protocol/1.1: Encoding and Transport«, September 2000. `http://www.ietf.org/rfc/rfc2910.txt`

RFC2911 T. Hastings, R. Herriot, R. deBry, et al. »Internet Printing Protocol/1.1: Model and Semantics«, September 2000. `http://www.ietf.org/rfc/rfc2911.txt`



10

Die Secure Shell

Inhalt

10.1 Einführung 152
10.2 Anmelden auf entfernten Rechnern mit ssh 152
10.3 Andere nützliche Anwendungen: scp und sftp 156
10.4 Client-Authentisierung über Schlüsselpaare 156
10.5 Portweiterleitung über SSH 159
10.5.1 X11-Weiterleitung 159
10.5.2 Beliebige TCP-Ports weiterleiten 160

Lernziele

- Die Secure Shell (SSH) anwenden und konfigurieren können

Vorkenntnisse

- TCP/IP-Kenntnisse (Kapitel 3)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)
- Kenntnisse über Linux-Systemkonfiguration
- Grundkenntnisse über Kryptografie sind hilfreich

10.1 Einführung

SSH (*Secure Shell*) ist ein Netzwerkprotokoll der TCP/IP-Familie. Es ermöglicht die Datenübertragung im Netz mit sicherer Authentisierung und Verschlüsselung. Zu seinen Anwendungen gehören interaktive Anmeldevorgänge, Übertragung von Dateien und die gesicherte Weiterleitung anderer Protokolle (engl. *tunneling*).

Verschlüsselung ist wichtig dafür, dass Unbefugte, die den Netzverkehr belauschen, keine Kenntnis von den übertragenen Inhalten nehmen können. Authentisierung stellt einerseits sicher, dass Sie als Benutzer mit dem richtigen Server kommunizieren, und andererseits, dass der Server Ihnen Zugriff auf das richtige Benutzerkonto einräumt.

OpenSSH

OpenSSH, das mit den meisten Linux-Distributionen ausgeliefert wird, stellt eine frei verfügbare Implementierung dieses Protokolls dar. Diese Implementierung enthält einige SSH-Clients sowie einen SSH-Server (`sshd`).

Angriffe

Richtig angewendet kann SSH die folgenden Angriffe verhindern:

- »DNS Spoofing«, also ge- oder verfälschte DNS-Einträge
- »IP Spoofing«, bei dem ein Angreifer von einem Rechner aus Pakete schickt, die so tun, als ob sie von einem anderen (vertrauenswürdigen) Rechner kämen.
- IP Source Routing, bei dem ein Rechner vortäuschen kann, dass Pakete von einem anderen (vertrauenswürdigen) Rechner kommen.
- Ausspähen von Klartextkennwörtern und Nutzdaten durch Stationen entlang des Übertragungswegs.
- Manipulation von übertragenen Daten durch Stationen entlang des Übertragungswegs.
- Angriffe auf den X11-Server durch ausgespähte X-Authentisierungsdaten und vorgetäuschte Verbindungen zum X11-Server.

SSH hilft nicht gegen andere Angriffe auf Ihren Rechner oder den SSH-Server. Insbesondere kann ein Angreifer, der auf einem der beiden Rechner auf andere Weise Administratorrechte erlangt hat, auch SSH kompromittieren.

Einsatzmöglichkeiten

SSH ersetzt die unsicheren Protokolle TELNET, RLOGIN und RSH für interaktive Anmeldevorgänge. Zusätzlich bietet es die Möglichkeit, Dateien von einem entfernten Rechner zu kopieren und ist so ein sicherer Ersatz für RCP und viele Anwendungen von FTP.

Protokollversionen

Das SSH-Protokoll existiert in zwei Versionen, 1 und 2. Viele Clients unterstützen beide Versionen und die meisten Server können Verbindungen mit beiden Versionen entgegennehmen. Machen Sie trotzdem einen Bogen um die Version 1, die diverse Sicherheitslücken aufweist.

10.2 Anmelden auf entfernten Rechnern mit `ssh`

Um sich über SSH auf einem entfernten Rechner anzumelden, müssen Sie das Kommando `ssh` aufrufen, etwa so:

```
$ ssh blue.example.com
hugo@blue.example.com's password: geHe1m
Last login: Mon Feb  2 10:05:25 2009 from 192.168.33.1
Debian GNU/Linux (etch/i686) blue.example.com
hugo@blue:~$ _
```

Dabei geht ssh davon aus, dass Ihr Benutzerkonto auf dem entfernten Rechner genauso heißt wie auf dem lokalen Rechner. Wenn das nicht so ist, können Sie Ihren entfernten Benutzernamen wie folgt angeben:

```
$ ssh hschulz@blue.example.com
```

Hinter den Kulissen passiert beim Verbindungsaufbau ungefähr das Folgende:

- Client und Server senden einander Informationen über ihre Rechnerschlüssel, unterstützte kryptografische Verfahren und ähnliches. Der Client prüft, ob der öffentliche Schlüssel noch so ist wie früher (dazu gleich mehr) und handelt mit dem Server ein gemeinsames Geheimnis aus, das dann als (symmetrischer) Schlüssel für die Verschlüsselung der Verbindung dient. Im selben Schritt prüft der Client die Authentizität des Servers und bricht die Verbindung ab, falls daran Zweifel bestehen. Die (unappetitlichen) Details stehen in [RFC4253].
- Der Server prüft die Authentizität des Clients über eines von mehreren Verfahren (hier im Beispiel fragt er nach einem Kennwort). Das Kennwort wird bereits über die verschlüsselte Verbindung übertragen und kann im Gegensatz zu Protokollen wie FTP oder TELNET von »Mithörern« nicht abgefangen werden.

Der erste Schritt ist dabei durchaus entscheidend. Das folgende Beispiel zeigt, was passiert, wenn Sie zum ersten Mal mit dem entfernten Rechner Kontakt aufnehmen:

```
$ ssh blue.example.com
The authenticity of host 'blue.example.com (192.168.33.2)' can't be▷
 ◁ established.
RSA key fingerprint is 81:24:bf:3b:29:b8:f9:f3:46:57:18:1b:e8:40:5a▷
 ◁ :09.
Are you sure you want to continue connecting (yes/no)? _
```

Der Rechner blue.example.com ist hier noch unbekannt, und ssh bittet Sie darum, seinen Rechnerschlüssel zu verifizieren. *Das ist ernst gemeint.* Wenn Sie diesen Überprüfungsschritt überspringen, verlieren Sie die Garantie dafür, dass niemand Ihre Verbindung belauscht.

Die Gefahr ist hier, dass jemand Ihren Verbindungswunsch abfängt und so tut, als wäre er blue.example.com. Hinter den Kulissen kann er selbst eine Verbindung zu blue.example.com aufbauen, über die er alles schickt, was Sie ihm naiverweise schicken, und Ihnen die Antworten von blue weiterleiten. Sie sehen keinen Unterschied, aber der Angreifer kann alles lesen, was Sie übertragen. Man nennt das einen *Man-in-the-middle*-Angriff.

Zur Überprüfung kontaktieren Sie den Administrator des entfernten Rechners (zum Beispiel per Telefon) und bitten ihn, Ihnen den »Fingerabdruck« (engl. *fingerprint*) des öffentlichen Rechnerschlüssels vorzulesen. Dieser kann mit »ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key« ermittelt werden und muss mit dem »RSA key fingerprint« aus dem SSH-Anmeldedialog übereinstimmen.

Die SSH-Schlüsselpaare eines Rechners finden Sie in den Dateien ssh_host_*x*_key bzw. ssh_host_*x*_key.pub im Verzeichnis /etc/ssh. *x* steht dabei für ein bestimmtes kryptografisches Verfahren, das Clients verwenden können, um die Authentizitität des Servers zu überprüfen.

SSH-Schlüsselpaare

Mögliche Werte für *x* sind (Juli 2015):

rsa Das RSA-Verfahren. Ist (nach dem Stand der Forschung) sicher, solange Sie Schlüssel verwenden, die länger als 1024 Bit sind. (2048 Bit klingen gut. Benutzen Sie 4096 Bit, wenn Sie Edward Snowden sind oder anderweitig davon ausgehen, dass Organisationen wie die NSA Sie gezielt – und nicht nur nebenbei zufällig – auf dem Kieker haben.)

dsa Das DSA-Verfahren. Erlaubt nur 1024-Bit-Schlüssel und sollte heutzutage vermieden werden, auch weil es anfällig gegen Schwächen bei der Zufallszahlen-Erzeugung ist.

ecdsa Das DSA-Verfahren auf der Basis elliptischer Kurven. Hier haben Sie die Auswahl zwischen 256, 384 und 521 Bits[1]. (Elliptische Kurven brauchen nicht so viele Bits, darum sind die niedrigen Zahlen unproblematisch.)

ed25519 Ein schnelles und (nach bisheriger Kenntnis) sehr sicheres Verfahren, das auf Daniel J. Bernstein zurückgeht. In der Secure Shell allerdings noch ziemlich neu.

Mit 2048-Bit-RSA machen Sie für die nächsten paar Jahre mit ziemlicher Sicherheit nichts falsch. Wenn Sie sicher sind, dass Ihre Clients und Server Ed25519 unterstützen, ist das eine empfehlenswerte Alternative.

Ein »Schlüsselpaar«, nur um das mal gesagt zu haben, ist übrigens ein Paar von zwei zusammengehörenden Schlüsseln (!), einem privaten und einem öffentlichen Schlüssel. Der öffentliche Schlüssel darf überall herumerzählt werden, solange der private Schlüssel vertraulich bleibt. Was mit dem öffentlichen Schlüssel verschlüsselt wurde, kann *nur* mit dem privaten Schlüssel aus demselben Paar wieder entschlüsselt werden und umgekehrt.

Ist der öffentliche Schlüssel des entfernten Rechners authentisch, dann beantworten Sie die Frage mit »yes«. `ssh` speichert den öffentlichen Schlüssel dann in der Datei `~/.ssh/known_hosts` ab, um ihn bei späteren Verbindungsaufbauten als Vergleichsgrundlage heranzuziehen.

Sollten Sie beim Aufbauen einer `ssh`-Verbindung jemals eine Meldung sehen wie

```
$ ssh blue.example.com
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle▷
 ◁ attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
38:fa:2e:d3:c7:c1:0f:26:2e:59:e8:16:a4:0a:0b:94.
Please contact your system administrator.
Add correct host key in /home/hugo/.ssh/known_hosts to get rid of▷
 ◁ this message.
Offending key in /home/hugo/.ssh/known_hosts:4
RSA host key for blue.example.com has changed and you have requested▷
 ◁ strict checking.
Host key verification failed.
```

dann besteht der Verdacht, dass Sie gerade einem *Man-in-the-middle*-Angriff ausgesetzt sind – der öffentliche Rechnerschlüssel, den der Server präsentiert, stimmt nicht mit dem überein, der für den Server in der `known_hosts`-Datei abgelegt ist. Sie

[1] Ja, wirklich 521, das ist kein Tippfehler für 512. ($2^{521}-1$ ist eine Mersenne-Primzahl, und das macht die Implementierung schneller. 521 Bits sind allerdings ziemlicher Overkill.)

sollten sich an den Systemadministrator wenden und nachfragen, was Sache ist – vielleicht musste der Rechnerschlüssel aus anderen Gründen geändert werden.

Dieses Verhalten können Sie ändern, indem Sie eine entsprechende Einstellung in der Datei `~/.ssh/config` machen:

`StrictHostKeyChecking ask`	*Voreinstellung*
`StrictHostKeyChecking no`	*Alles immer akzeptieren*
`StrictHostKeyChecking yes`	*Nichts Neues akzeptieren*

Bei »`StrictHostKeyChecking yes`« können Sie nur Verbindungen zu Rechnern aufbauen, die schon in Ihrer `known_hosts`-Datei stehen. Alle anderen werden abgewiesen.

Nachdem Sie über `ssh` eine Verbindung aufgebaut haben, können Sie an dem entfernten Rechner so arbeiten, als säßen Sie davor. Sie können die Verbindung mit `exit` oder Strg+d beenden.

Wenn Sie nichts anderes sagen, gilt bei interaktiven `ssh`-Sitzungen eine Tilde (»~«), wenn ihr in der Eingabe unmittelbar ein Zeilentrenner vorausgeht, als Sonderzeichen, mit dem Sie die `ssh` steuern können. Insbesondere bricht die Kombination »~.« die Verbindung ab, was nützlich sein kann, wenn sich am »anderen Ende« ein Programm aufgehängt hat. Sie können auch noch andere interessante Dinge tun – schauen Sie sich den Abschnitt »ESCAPE CHARACTERS« in `ssh`(1) an.

Übrigens sind Sie mit `ssh` nicht auf interaktive Sitzungen beschränkt, sondern können auf dem entfernten Rechner auch einzelne Kommandos ausführen:

```
$ ssh blue.example.com hostname
hugo@blue.example.com's password: geHe1m
blue.example.com
$ _
```

Dabei müssen Sie natürlich beachten, dass die Shell auf *Ihrem* Rechner die Kommandozeile bearbeitet und zum Beispiel etwaige Suchmuster zu ersetzen versucht, bevor das Kommando an den entfernten Rechner übertragen wird. Verwenden Sie gegebenenfalls Anführungszeichen oder Rückstriche.

Übungen

10.1 [!1] Verwenden Sie das `ssh`-Kommando, um sich auf einem anderen Rechner anzumelden (Ihr Trainer erklärt Ihnen gegebenfalls, auf welchem). Was passiert? Melden Sie sich ab und melden Sie sich nochmals auf demselben Rechner an. Was ist anders?

10.2 [2] Entfernen Sie den Eintrag für den entfernten Rechner aus Übung 10.1 aus der Datei `~/.ssh/known_hosts` und stellen Sie in der Datei `~/.ssh/ssh_config` den Parameter `StrictHostKeyChecking` auf `yes`. Versuchen Sie dann nochmals, sich auf dem entfernten Rechner anzumelden. Was passiert? Was passiert, wenn der Parameter `StrictHostKeyChecking` auf `no` steht?

10.3 [2] Darf die Datei `~/.ssh/known_hosts` nur für den Benutzer lesbar sein und wenn ja, warum? (Wenn nein, warum nicht?)

10.4 [!2] Führen Sie auf dem entfernten Rechner mit einem einzigen `ssh`-Aufruf die Kommandos `hostname` und `date` aus.

10.3 Andere nützliche Anwendungen: `scp` und `sftp`

Mit `scp` können Sie Dateien zwischen zwei Rechnern über eine SSH-Verbindung kopieren:

```
$ scp blue.example.com:hello.c .
hugo@blue.example.com's password: geHe1m
hello.c   100% |**********************| 33 KB   00:01
```

Die Syntax ist dabei an das `cp`-Kommando angelehnt: Genau wie dort können Sie zwei Dateinamen (Quelle und Ziel) oder eine Reihe von Dateinamen und ein Verzeichnis angeben. Mit der Option `-r` kopiert `scp` Verzeichnisinhalte rekursiv.

Sie können sogar Dateien zwischen zwei verschiedenen entfernten Rechnern kopieren:

```
$ scp hugo@blue.example.com:hello.c \
>    hschulz@pink.example.com:hello-new.c
```

Das Kommando `sftp` ist locker an gängige FTP-Clients angelehnt, aber verwendet eine SSH-Verbindung. Mit FTP hat es ansonsten überhaupt nichts zu tun – insbesondere können Sie es nicht verwenden, um mit einem FTP-Server zu kommunizieren.

Nachdem Sie mit einem Kommando wie

```
$ sftp hugo@blue.example.com
```

eine Verbindung aufgebaut haben, können Sie mit Kommandos wie `get`, `put` oder `mget` Dateien zwischen Ihrem lokalen Rechner und dem entfernten Rechner übertragen, mit `ls` den Inhalt eines Verzeichnisses auf dem entfernten Rechner anschauen und mit `cd` dort in andere Verzeichnisse wechseln. Am Anfang einer Sitzung stehen Sie auf dem entfernten Rechner in Ihrem Heimatverzeichnis.

10.4 Client-Authentisierung über Schlüsselpaare

Im Normalfall authentisiert der SSH-Server Sie als Benutzer über ein Kennwort, das für Ihr Benutzerkonto auf dem Server hinterlegt ist (üblicherweise in `/etc/passwd` oder `/etc/shadow`). Da die Kennwortabfrage erst erfolgt, wenn die verschlüsselte Verbindung bereits steht, ist das grundsätzlich sicher vor unerwünschten Mithörern. Allerdings könnte es Ihnen ein Dorn im Auge sein, dass das Kennwort selbst auf dem Server liegt – auch wenn es verschlüsselt ist, könnte die Kennwortdatei Crackern in die Hände fallen, die dann »John the Ripper« darauf loslassen. Es wäre besser, wenn auf dem entfernten Rechner überhaupt nichts Geheimes von Ihnen gespeichert wäre.

Dies können Sie erreichen, indem Sie statt der einfachen kennwortbasierten Client-Authentisierung die Client-Authentisierung über Schlüsselpaare (engl. *public-key authentication*) verwenden. Kurz gesagt erzeugen Sie dafür ein Paar aus einem öffentlichen und einem privaten Schlüssel und hinterlegen den öffentlichen Schlüssel auf dem SSH-Server. Der öffentliche Schlüssel muss nicht besonders geschützt werden (immerhin ist er öffentlich); auf den privaten Schlüssel müssen Sie gut aufpassen, aber er verläßt nie Ihren eigenen Rechner (den Sie ja nie aus den Augen lassen, nicht wahr?).

Sie können den privaten Schlüssel auch auf einem USB-Stick hinterlegen, wenn Ihnen das sicherer vorkommt.

Der Server kann Sie als rechtmäßigen Inhaber des privaten Schlüssels authentisieren, der zu dem hinterlegten öffentlichen Schlüssel passt, indem er eine Zufallszahl auswürfelt, diese mit dem hinterlegten öffentlichen Schlüssel verschlüsselt

und Ihnen schickt. Sie entschlüsseln (oder genauer gesagt Ihre `ssh` entschlüsselt) die verschlüsselte Zufallszahl dann mit dem privaten Schlüssel. Das Ergebnis geht zurück an den Server; er vergleicht es mit seiner ursprünglichen Zufallszahl, und wenn die beiden übereinstimmen, glaubt er Ihnen, dass Sie Sie sind.

Das ganze geht natürlich über die verschlüsselte Verbindung und ist darum sicher vor unerwünschten Zuhörern oder Fieslingen, die an den Daten herumbasteln wollen.

Um die Client-Authentisierung über Schlüsselpaare verwenden zu können, müssen Sie zuerst ein Schlüsselpaar generieren. Das geht mit dem Kommando `ssh-keygen`:

```
$ ssh-keygen -t rsa -b 2048                                          oder ed25519
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hugo/.ssh/id_rsa): ↩
Created directory '/home/hugo/.ssh'.
Enter passphrase (empty for no passphrase): geheim
Enter same passphrase again: geheim
Your identification has been saved in /home/hugo/.ssh/id_rsa.
Your public key has been saved in /home/hugo/.ssh/id_rsa.pub.
The key fingerprint is:
39:ab:15:f4:2f:c4:e6:21:26:c4:43:d7:27:22:a6:c4 hugo@blue
The key's randomart image is:
+---[RSA 2048]----+
|   .  . ..       |
|    Eoo.. o .    |
|   . o+... o     |
|    .. o +       |
|      . S *      |
|       o 0 o     |
|        o o .    |
|       o   .     |
|      .          |
+-----------------+
```

Das Kommando fragt zunächst nach dem gewünschten Speicherort für das Schlüsselpaar. Die Vorgabe ist dabei vernünftig und Sie sollten sie einfach bestätigen.

Als nächstes möchte `ssh-keygen` eine *passphrase* wissen. Diese dient zur Verschlüsselung des *privaten* Schlüssels, um zu verhindern, dass jemand, dem Ihr privater Schlüssel in die Hände fällt, sich gegenüber dem SSH-Server als Sie ausgeben kann.

Sie können (oder sollten) hier durchaus einen längeren Satz angeben. Ein kürzeres Kennwort aus einer bunten Mischung von Buchstaben, Ziffern und Sonderzeichen ist wahrscheinlich auch in Ordnung. Es gelten die üblichen Regeln für solche Geheimnisse.

Schlüssel ohne *passphrase* müssen Sie benutzen, wenn Sie unbeaufsichtigte SSH-Verbindungen benötigen, etwa für Shellskripte und `cron`-Jobs. In diesem Fall drücken Sie bei den Fragen nach der *passphrase* einfach auf ↩.

Es ist möglich, einen öffentlichen Schlüssel auf dem Server fest mit einem bestimmten Kommando zu verbinden. Client-Aufrufe, die diesen öffentlichen Schlüssel verwenden, führen dann nicht zu einer Shell-Sitzung, sondern zu einem sofortigen Start des angegebenen Kommandos. Das Sicherheitsrisiko von unverschlüsselten privaten Schlüsseln zum Gebrauch durch Skripte kann damit signifikant vermindert werden.

Das Ergebnis von `ssh-keygen` sind die beiden Dateien `id_rsa` und `id_rsa.pub` im Verzeichnis `~/.ssh`. Dabei enthält erstere den privaten und letztere den öffentlichen Schlüssel.

Wenn Sie bei der Schlüsselgenerierung »`-t ed25519`« angegeben haben, heißen die Dateien natürlich `id_ed25519` und `id_ed25519.pub`.

Das Kommando `ssh-keygen` gibt Ihnen außerdem den Fingerabdruck des öffentlichen Schlüssels und ein *randomart image* aus. Dabei handelt es sich um eine grafische Darstellung des öffentlichen Schlüssels, eine Art grafischen Fingerabdruck. Dieser soll Sie theoretisch in die Lage versetzen, mit einem Blick sehen zu können, ob ein öffentlicher Schlüssel sich geändert hat oder nicht. Das Konzept ist, vorsichtig gesagt, umstritten.

Es hält Sie natürlich niemand davon ab, `ssh-keygen` mehrmals aufzurufen, um mehrere Schlüsselpaare mit verschiedenen Verschlüsselungsverfahren zu erzeugen. (Oder mehrere Schlüsselpaare mit demselben Verschlüsselungsverfahren zur Verwendung mit verschiedenen Servern. Dabei müssen Sie natürlich darauf achten, dass Sie verschiedene Dateinamen benutzen.)

Im nächsten Schritt müssen Sie den öffentlichen Schlüssel, also die `id_rsa.pub`-Datei, in der Datei `~/.ssh/authorized_keys` in Ihrem Benutzerkonto auf dem entfernten Rechner ablegen. Am bequemsten geht das mit dem Kommando `ssh-copy-id`:

```
$ ssh-copy-id hugo@blue.example.com
hugo@blue.example.com's password: geHe1m                    Ein letztes Mal
Now try logging into the machine, with "ssh 'hugo@blue.example.com'",▷
 ◁ and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

$ _
```

Natürlich können Sie das auch »zu Fuß« mit `scp` und/oder `ssh` erledigen. Achten Sie dabei nur darauf, etwaige schon vorhandene Schlüssel in `~/.ssh/authorized_keys` nicht zu überschreiben, auf die Sie vielleicht noch Wert legen.

Wenn Sie in der Datei `/etc/ssh/sshd_config` auf dem Server die Einträge `PasswordAuthentication` auf `no` und `PubkeyAuthentication` auf `yes` setzen, dann können Benutzer sich *nur* noch über die Schlüsselpaar-Methode authentisieren. Das ist grundsätzlich eine gute Idee, da Cracker sich einen Spaß daraus machen, mit automatischen Programmen SSH-Server nach naheliegenden Kennwörtern abzuklopfen.

Die Authentisierung über Schlüsselpaare ist zumindest bei Verwendung einer *passphrase* nicht komfortabler als die Kennwortauthentisierung, dafür aber um einiges sicherer. Falls Sie sich häufiger hintereinander auf dem gleichen Rechner als der gleiche Benutzer anmelden wollen, ist das ständige Wiedereingaben der *passphrase* allerdings lästig. Für solche Fälle wurde der `ssh-agent` entwickelt.

`ssh-agent` Der `ssh-agent` merkt sich die *passphrase* und übergibt sie bei jedem Aufruf eines SSH-Client-Programmes an dieses. Das Programm wird mit »`ssh-agent bash`«
`ssh-add` gestartet. Dabei öffnet sich eine neue `bash`, in der Sie die Passphrase mit `ssh-add` bekannt machen müssen:

```
$ ssh-add
Enter passphrase for /home/hugo/.ssh/id_rsa: Habe nun, ach!
Identity added: /home/hugo/.ssh/id_rsa (/home/hugo/.ssh/id_rsa)
```

Jedem aus der neuen Shell heraus gestarteten `ssh-`, `scp-` oder `sftp`-Programm wird vom SSH-Agent die *passphrase* übergeben. Der Agent »vergisst« die *passphrase* wieder, wenn Sie mit einem `exit` die Shell verlassen oder ihn mit »`ssh-add -D`« anweisen, die gespeicherten Identitäten wieder zu löschen.

Bei Debian GNU/Linux werden die Loginshell bzw. die grafische Umgebung auf Wunsch schon mit dem `ssh-agent` gestartet, so dass Sie gleich mit `ssh-add` Ihre *passphrase* eingeben können.

Der Fairness halber sollten wir erwähnen, dass die Verwendung des `ssh-agent` zwar die Bequemlichkeit erhöht, aber auf Kosten der Sicherheit geht. Wenn Sie Ihren Rechner unbeaufsichtigt lassen (oder Ihr in den Schlummermodus versetztes Notebook Ihnen abhanden kommt), kann eine unberechtigte Person möglicherweise die SSH-Programme benutzen, ohne nach einer *passphrase* gefragt zu werden. Dasselbe gilt für Programme, die irgendwie Zugang zu Ihrer Sitzung bekommen, also Viren, Würmer und ähnliches Viechzeug …

Übungen

10.5 [!2] Erzeugen Sie sich mit `ssh-keygen` ein RSA-Schlüsselpaar für die SSH-Protokollversion 2. (Denken Sie dran, mindestens 2048 Bit!) Installieren Sie den öffentlichen Schlüssel auf dem entfernten Rechner und überzeugen Sie sich, dass Sie bei der Anmeldung nicht mehr nach dem Kennwort für den entfernten Rechner gefragt werden. Was müssen Sie statt dessen eingeben?

10.6 [!1] Bestimmen Sie den »Fingerabdruck« Ihres öffentlichen Schlüssels.

10.7 [2] Unter welchen Umständen könnten Sie auf die Verwendung einer *passphrase* für den privaten Schlüssel verzichten wollen?

10.5 Portweiterleitung über SSH

10.5.1 X11-Weiterleitung

Ausführen von grafischen Programmen

Die X11-Weiterleitung ermöglicht das Ausführen von grafischen Programmen (X-Clients) auf einem entfernten Rechner, wobei die Grafikausgabe und Bedienung auf dem lokalen Rechner erfolgt. Sie müssen sich dafür lediglich per `ssh` auf dem entfernten Rechner anmelden und dabei die Option `-X` (großes X!) verwenden. Voraussetzung ist, dass auf der Server-Seite die X11-Weiterleitung (Parameter `X11Forwarding` in `/etc/ssh/sshd_config`) aktiviert ist.

Nach einer Anmeldung mit »`ssh -X` [⟨*Benutzername*⟩`@`]⟨*Rechner*⟩« befinden Sie sich auf dem Server und können beliebige X-Programme ausführen, die auf dem lokalen X-Server angezeigt und bedient werden können. Das beruht auf mehreren Faktoren:

- Beim Anmelden mit `-X` wird automatisch die `DISPLAY`-Variable gesetzt – sie zeigt auf einen »Proxy«-X-Server, den der `sshd` zur Verfügung stellt. Dadurch werden X-Clients auf dem entfernten Rechner an diesen Server verwiesen.
- Alle Äußerungen eines auf dem entfernten Rechner gestarteten X-Clients an den dortigen Proxy-X-Server werden an den (echten) X-Server auf dem SSH-Client weitergeleitet.
- Die weitergeleiteten X-Pakete des X-Clients werden von der SSH außerdem verschlüsselt, so dass die Verbindung von außen nicht zu belauschen ist (Tunneling).


Sie können das X11-Forwarding auch pauschal in Kraft setzen, um sich die `-X` sparen zu können. Dazu müssen Sie die Zeile »`ForwardX11 yes`« in die Datei `~/.ssh/config` (oder systemweit `/etc/ssh/ssh_config`) aufnehmen.

Das X11-Forwarding ist dem herkömmlichen Umleiten der X-Pakete per `DISPLAY`-Variable auf jeden Fall vorzuziehen, nicht nur wegen der Sicherheit, sondern, weil sie schlicht komfortabler ist. Sie bezahlen dafür mit einem gewissen Extraaufwand für die Verschlüsselung, was bei heutigen Rechnern aber in der Regel nicht ins Gewicht fällt.


Ganz ohne Sicherheitsrisiken ist auch X11-Forwarding nicht: Benutzer, die die Dateizugriffsrechte auf dem entfernten Rechner unterlaufen können (etwa weil sie dort `root` sind), können auf das lokale X-Display zugreifen, indem sie sich den Inhalt Ihrer `.Xauthority`-Datei auf dem entfernten Rechner anschauen. Aus diesem Grund sollten Sie X11-Forwarding wahrscheinlich nicht pauschal einschalten. Dasselbe Risiko besteht natürlich bei »herkömmlichem« X11-basierter Ausgabeweiterleitung über `DISPLAY`.

10.5.2 Beliebige TCP-Ports weiterleiten

Portweiterleitung

SSH kann nicht nur das X-Protokoll, sondern so gut wie jedes andere TCP-basierte Protokoll weiterleiten und tunneln. Dazu stehen die Optionen `-R` und `-L` zur Verfügung. Das folgende Kommando leitet Verbindungen an den lokalen TCP-Port 10110 zunächst über eine SSH-Verbindung auf den Rechner `blue.example.com` um. Von da geht es (unverschlüsselt) weiter auf den TCP-Port 110 (POP3) auf dem Rechner `mail.example.com`:

```
$ ssh -L 10110:mail.example.com:110 hugo@blue.example.com
```

Der Nutzen dieses Szenarios ist etwa wie folgt: Stellen Sie sich vor, Ihr Firewall sperrt POP3, aber läßt SSH durch. Über die Portweiterleitung kommen Sie via SSH ins interne Netz und können vom Rechner `blue.example.com` aus rein im internen Netz mit dem Mailserver reden. In Ihrem Mailprogramm geben Sie dann `localhost` und den lokalen TCP-Port 10110 als »POP3-Server« an.


Theoretisch könnten Sie auch den lokalen TCP-Port 110 weiterleiten, aber dazu müssen Sie `root` sein.


Der Name des Rechners für die Weiterleitung (hier `mail.example.com`) wird aus der Sicht des SSH-Servers (hier `blue.example.com`) aufgelöst. Das heißt, eine Weiterleitung der Form

```
$ ssh -L 10110:localhost:110 hugo@blue.example.com
```

verbindet Sie mit dem Port 110 auf `blue.example.com`, nicht etwa auf Ihrem Rechner.


Eine Weiterleitung der Form

```
-L 10110:mail.example.com:110
```

öffnet den Port 10110 auf *allen* IP-Adressen Ihres Rechners. Damit wird die Weiterleitung prinzipiell auch für andere Stationen zugänglich, die über das Netz mit diesem Port Kontakt aufnehmen können. Um das zu verhindern, können Sie ausnutzen, dass `ssh` es Ihnen erlaubt, eine lokale Adresse für den weitergeleiteten Port anzugeben: Mit

```
-L localhost:10110:mail.example.com:110
```

gilt die Weiterleitung nur für die lokale Schnittstelle.

Wenn Sie `ssh` wie gezeigt aufrufen, bekommen Sie außer der Portweiterleitung auch eine interaktive Sitzung. Wenn Sie das nicht möchten – etwa weil die Weiterleitung in einem `cron`-Job stattfindet –, können Sie die Option `-N` angeben, die `ssh` auf die Weiterleitung beschränkt und keine interaktive Sitzung aufbaut.

Eine andere (möglicherweise bessere) Technik für das automatische Weiterleiten von Diensten verwendet einen `ssh`-Aufruf der Form

```
$ ssh -f -L 10110:mail.example.com:110 blue sleep 10
$ getmail_fetch -p10110 localhost hugomail MaIl123 Maildir/
```

Die Option `-f` sorgt dafür, dass der `ssh`-Prozess unmittelbar vor der Ausführung des »`sleep 10`«-Kommandos in den Hintergrund geht. Das heißt, ein Kommando, das Sie unmittelbar nach dem `ssh`-Aufruf ausführen (hier `getmail_fetch`, das Mail über POP3 abruft), hat 10 Sekunden Zeit, eine Verbindung über den lokalen Port 10110 aufzubauen. Der `ssh`-Prozess beendet sich entweder nach 10 Sekunden oder aber wenn die (letzte) Verbindung über den lokalen Port 10110 wieder abgebaut wird, was auch immer später passiert.

Die Portweiterleitung funktioniert auch umgekehrt: Mit

```
$ ssh -R 10631:localhost:631 hugo@blue.example.com
```

wird der TCP-Port 10631 *auf dem SSH-Server* geöffnet und Verbindungen, die Programme dort mit diesem Port aufnehmen, über die SSH-Verbindung auf Ihren lokalen Rechner geleitet. Ihr lokaler Rechner übernimmt dann die unverschlüsselte Weiterleitung an das angegebene Ziel, hier den TCP-Port 631 auf Ihrem lokalen Rechner selbst. (Diese Form der Weiterleitung ist ungleich weniger wichtig als die über `-L`.)

Die `-R`-Weiterleitung bindet den entfernten Port normalerweise an die `localhost`-Schnittstelle auf dem SSH-Server. Sie können sich grundsätzlich wie oben gezeigt eine andere Schnittstelle wünschen (»`*`« steht für »alle«), aber ob das funktioniert, hängt von der Konfiguration des SSH-Servers ab.

Portweiterleitungen können Sie auch nachträglich einrichten. Dazu müssen Sie die Tastenkombination »`~C`« verwenden (es muss ein großes C sein), die eine »Kommandozeile« zur Verfügung stellt:

```
◁◁◁◁◁                                   Hier läuft gerade eine SSH-Sitzung
remote$ ↩
remote$ ~ C
ssh> -L 10025:localhost:25
Forwarding port.
◁◁◁◁◁                                   SSH-Sitzung geht weiter
```

Auf der »Kommandozeile« können Sie (unter anderem) `-L`- und `-R`-Optionen nachtragen, so als ob Sie sie gleich beim `ssh`-Aufruf angegeben hätten. Mit `-KR` gefolgt von der Portnummer können Sie eine `-R`-Weiterleitung auch wieder rückgängig machen (`-KL` gibt es leider nicht). Mit der Tastenkombination »`~#`« bekommen Sie einen Überblick über die gerade aktiven Verbindungen:

```
◁◁◁◁◁
remote$ ~#
The following connections are open:
  #2 client-session (t4 r0 i0/0 o0/0 fd 6/7 cfd -1)
  #3 direct-tcpip: listening port 10025 for localhost port 25,▷
 ◁ connect from 127.0.0.1 port 57250 ▷
 ◁ (t4 r1 i0/0 o0/0 fd 9/9 cfd -1)
◁◁◁◁◁
```

Wie Sie den vorstehenden Abschnitten zweifellos entnommen haben, bietet `ssh` diverse Möglichkeiten an, Schindluder zu treiben, die einem IT-Sicherheitsbeauftragten Tränen in die Augen steigen lassen dürften. Bitte werten Sie dieses Kapitel als Vorstellung einiger Fähigkeiten der `ssh`, nicht als Empfehlung dafür, möglichst viele davon tatsächlich zu verwenden (jedenfalls ohne guten Grund). Als Betreiber eines SSH-Servers sollten Sie insbesondere dessen Dokumentation (etwa `sshd_config`(5)) studieren, um zu lernen, wie Sie die Verwendung der gefährlicheren Optionen unterdrücken können. Für eine ausführliche Beschreibung der SSH-Serverkonfiguration ist in dieser Schulungsunterlage leider kein Platz.

Übungen

10.8 [!1] Wie können Sie die `ssh` verwenden, um von einem normalen Benutzerzugang aus auf dem lokalen Rechner bequem X-Clients mit `root`-Rechten zu starten?

10.9 [3] Verwenden Sie die `ssh`, um den Port 4711 (oder einen anderen geeigneten lokalen Port) auf Ihrem Rechner auf den `echo`-Port (Port 7) des entfernten Rechners umzuleiten. Überzeugen Sie sich mit einem Paketsniffer (`tcpdump` oder `wireshark`), dass eine Verbindung zum lokalen Port 4711, etwa mit »`telnet localhost 4711`« tatsächlich eine verschlüsselte Datenübertragung zum entfernten Rechner bewirkt und erst auf diesem wieder im Klartext weitergeführt wird.

Kommandos in diesem Kapitel

`scp` Sicheres Dateikopierprogramm auf SSH-Basis `scp`(1) 155
`sftp` Sicheres FTP-artiges Programm auf SSH-Basis `sftp`(1) 156
`ssh` „Secure Shell“, erlaubt sichere interaktive Sitzungen auf anderen Rechnern `ssh`(1) 152
`ssh-add` Akkreditiert private Schlüssel beim `ssh-agent` `ssh-add`(1) 158
`ssh-agent` Verwaltet private Schlüssel und Kennwörter für die SSH `ssh-agent`(1) 158
`ssh-copy-id` Kopiert öffentliche SSH-Schlüssel auf andere Rechner `ssh-copy-id`(1) 158
`ssh-keygen` Generiert und verwaltet Schlüssel für die SSH `ssh-keygen`(1) 157
`sshd` Server für das SSH-Protokoll (sicherer interaktiver Fernzugriff) `sshd`(8) 152

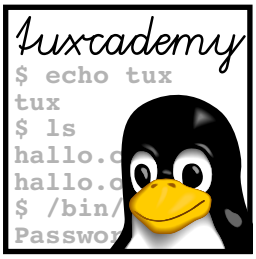
Zusammenfassung

- Die Secure Shell erlaubt das komfortable und sichere Anmelden auf entfernten Rechnern (und ersetzt so TELNET, RSH und RLOGIN) sowie die gesicherte Übertragung von Dateien ähnlich RCP und FTP.
- Mit OpenSSH steht eine leistungsfähige Implementierung der Secure Shell frei zur Verfügung.
- Der Benutzer hat die Wahl zwischen kennwortbasierter Authentisierung und Authentisierung über ein asymmetrisches Schlüsselpaar. Letztere ist sicherer, aber aufwendiger zu konfigurieren.
- Die Secure Shell kann die X11-Grafikdarstellung und -interaktion sowie beliebige TCP-Verbindungen über die verschlüsselte Strecke weiterleiten.

Literaturverzeichnis

BS01 Daniel J. Barrett, Richard Silverman. *SSH, The Secure Shell: The Definitive Guide*. Sebastopol, CA: O'Reilly & Associates, 2001. ISBN 0-596-00011-1. http://www.oreilly.com/catalog/sshtdg/

RFC4253 T. Ylonen, C. Lonvick. »The Secure Shell (SSH) Transport Layer Protocol«, Januar 2006. http://www.ietf.org/rfc/rfc4253.txt



11

Elektronische Post

Inhalt

11.1 Grundlagen 166
11.2 MTAs für Linux 166
11.3 Grundlegende Funktionen 167
11.4 Verwaltung der Nachrichten-Warteschlange 168
11.5 Lokale Zustellung, Aliasadressen und benutzerspezifische Weiterleitung 169

Lernziele

- Die gängigen Mailserver-Programme unter Linux namentlich kennen
- Grundlegende Mail-Weiterleitung und Aliasadressen konfigurieren können
- Die wichtigsten Kommandos zur Verwaltung eines Mailservers kennen

Vorkenntnisse

- TCP/IP-Kenntnisse (Kapitel 3)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)
- Kenntnisse über Linux-Systemkonfiguration

11.1 Grundlagen

Elektronische Post ist einer der am häufigsten genutzten Dienste im Internet. Eine zentrale Rolle dabei spielen *Mail Transfer Agents* (MTAs) – Programme, die elektronische Post weiterleiten oder empfangen können. Während Benutzer direkt mit *Mail User Agents* oder MUAs – Programmen wie KMail, Mutt oder Outlook Express – interagieren, um Nachrichten zu lesen, zu schreiben, zu beantworten, zu sortieren oder zu löschen, bedienen MUAs sich der Dienste der MTAs, um Nachrichten zu ihren Adressaten zu befördern. MTAs können dabei bei Internet-Providern untergebracht oder auch vor Ort installiert sein. Zu den Aufgaben von MTAs gehört auch das Umschreiben von Adressen in »kanonische« oder für die Versendung von Rückantworten brauchbare Form, das Wiederholen von missglückten Sendeversuchen, die Benachrichtigung des Nachrichtenabsenders über Fehler bei der Zustellung oder diverse Optimierungsaufgaben für Versandzeit, Netzlast oder Kosten. MTAs untereinander verwenden auf dem Internet das Protokoll SMTP (*Simple Mail Transfer Protocol*).

MTA

MUA

SMTP

Im Umfeld elektronischer Post sind auch andere Protokolle wie POP3 oder IMAP von Bedeutung, allerdings nicht für die LPIC-1-Zertifizierung. Das Thema wird im Detail in der Linup-Front-Schulungsunterlage *Linux als Mailserver* behandelt.

11.2 MTAs für Linux

Ein gängiger MTA für Unix- und Linux-Systeme ist das Programm Sendmail, das Anfang der 1980er Jahre von Eric Allman an der Berkeley-Universität entwickelt und bis heute weiterentwickelt wurde. Trotz seiner extrem komplexen Konfiguration und einer langen Geschichte von Sicherheitslücken hat Sendmail noch eine große Fangemeinde (auch unter Linux-Distributoren). Andere populäre MTAs sind Postfix von Wietse Venema, Exim von Philip Hazel und Qmail von Dan J. Bernstein.

Sendmail

Postfix
Exim
Qmail

In einem heftigen Anfall von *design by committee* haben die Erfinder der LPIC-1-Zertifizierung festgelegt, dass Kandidaten *alle* diese MTAs kennen müssen – glücklicherweise nur auf relativ grundlegender Ebene. Ebenfalls glücklicherweise bemühen sich zumindest Postfix und Exim um eine gewisse Kompatibilität zu Sendmail, was bestimmte Aspekte der Konfiguration und Kommandostruktur angeht; nur Qmail tanzt da aus der Reihe. Im Folgenden erklären wir die wichtigsten Eigenschaften aller dieser MTAs auf der elementaren Ebene, die für LPIC-1 verlangt wird.

Architektur

Allgemein kann man sagen, dass Sendmail und Exim von der Architektur her am engsten verwandt sind. Bei beiden läuft der komplette MTA als ein einziger Prozess. Postfix und Qmail dagegen teilen die MTA-Funktionalität vor allem aus Sicherheitsgründen auf eine ganze Familie von Prozessen auf. Der Vorteil dieses Ansatzes besteht darin, dass jeder Prozess sich auf einen Teil der Aufgabe konzentrieren kann, eine Kommunikation zwischen den einzelnen Prozessen nur über wohldefinierte Schnittstellen möglich ist und jeder Prozess mit den minimalen nötigen Rechten laufen kann. Während zumindest potentiell alle Teile von Sendmail und Exim Zugang zu Administrator-Rechten haben, ist das bei Postfix und Qmail auf den »Chefprozess« sowie typischerweise diejenigen Prozesse beschränkt, die Mail an Benutzer zustellen und darum deren Identität annehmen können müssen (wofür Administrator-Privilegien nötig sind). Qmail ist absichtlich in seiner Funktion eingeschränkt, ebenfalls aus Sicherheitsgründen (Funktionalität, die nicht vorhanden ist, kann auch keine Sicherheitsprobleme haben); dabei fallen viele Eigenschaften weg, die die anderen MTAs von Hause aus mitbringen und die bei Qmail explizit nachgerüstet werden müssen. Von den hier

genannten MTAs sind Sendmail und Qmail mit Abstand am schwierigsten zu warten.


Wenn Sie in der Position sind, sich frei für einen MTA entscheiden zu können, dann verwenden Sie Postfix.

11.3 Grundlegende Funktionen

Ein MTA wie Sendmail hat zwei wesentliche Aufgaben:

- Er **lauscht** auf dem für SMTP vorgesehenen TCP-Port 25 auf Verbindungen von anderen MTAs, die Nachrichten für lokale Empfänger einliefern wollen. Diese Nachrichten werden über einen MDA oder *Mail Delivery Agent* in die Postfächer der Empfänger geschrieben oder gemäß deren Wünschen zum Beispiel an andere Adressen weitergeleitet. Zum Empfang von Nachrichten über TCP-Port 25 muss der MTA entweder permanent als freistehender Daemon laufen oder bei Bedarf über einen »Über-Dienst« wie `inetd` oder `xinetd` aufgerufen werden. Letzteres lohnt sich nur bei sehr geringem Mailaufkommen.

  Empfang von Nachrichten

  MDA

- Er **versendet** Nachrichten, die von lokalen MUAs und anderen Programmen entweder über direkten Programmaufruf oder ebenfalls SMTP und den TCP-Port 25 zur Zustellung eingereicht werden. Da diese Nachrichten genauso gut an lokale Benutzer wie an entfernte geschickt werden können, ist diese Funktion nicht von der Empfangsfunktion zu trennen.

  Versand von Nachrichten


Sie müssen nicht beide Funktionen gleichzeitig in Anspruch nehmen. Wenn Sie zum Beispiel nicht mit Nachrichten aus dem Internet rechnen – etwa weil Ihr Rechner nicht über einen direkten Internetanschluss verfügt –, brauchen Sie prinzipiell auch keinen MTA auf dem TCP-Port 25 lauschen zu lassen, jedenfalls nicht auf anderen IP-Adressen als `localhost`.


Umgekehrt können Sie den Versand lokal erzeugter Nachrichten ermöglichen, ohne sie auch lokal zuzustellen; in lokalen Netzen ist es oft sinnvoll, einen einzigen Rechner zum Mailserver zu machen und auf anderen Rechnern nur einen MTA mit einer Minimalkonfiguration zu installieren, der alle eingereichten Nachrichten an den Mailserver weiterleitet, wo sie dann lokal zugestellt oder ins Internet geschickt werden.

Sendmail schreibt Nachrichten an lokale Benutzer normalerweise in deren Postfächer im Verzeichnis `/var/mail` (auf vielen System findet sich auch noch das offiziell veraltete Verzeichnis `/var/spool/mail`, meist irgendwie über einen symbolischen Link mit `/var/mail` zusammengeworfen), wo MUAs oder IMAP- oder POP-Server dann auf sie zugreifen können. Jedes Postfach ist dabei eine Datei, die normalerweise so heißt wie der Benutzer, dem es gehört; neue Nachrichten werden einfach hinten an die Datei angehängt. Damit das richtig funktioniert, müssen alle Programme, die das Postfach benutzen, in der Lage sein, es zu *sperren*, damit nicht gleichzeitig der MUA Nachrichten löscht, während der MTA neue anhängt; die Abwesenheit standardisierter und funktionssicherer Datei-Sperrmethoden bei Linux macht das manchmal zu einem riskanten Vorhaben.

Postfächer


Qmail verwendet im Gegensatz zu Sendmail ein eigenes Postfachformat namens »Maildir«, das statt einer großen Datei für alle Nachrichten ein geschicktes Arrangement von Verzeichnissen verwendet, in dem jede Nachricht in ihrer eigenen Datei steht. Dadurch werden Probleme mit den Sperrmechanismen weitestgehend ausgeschlossen. Maildir-Postfächer liegen nach Konvention nicht in `/var/mail`, sondern im Heimatverzeichnis des betreffenden Benutzers, was Kontingentierung und Sicherheitskopien vereinfachen kann.

Maildir

Postfix und Exim arbeiten in ihrer Standardeinstellung genauso wie Sendmail, bieten aber optional auch die Zustellung in Maildir-Dateien à la Qmail an.

Warteschlange

Nachrichten, die anderswohin weitergeleitet werden sollen, versucht Sendmail in der Regel baldmöglichst loszuwerden; klappt das nicht auf Anhieb, entweder weil die Gegenstelle gerade nicht zu erreichen ist oder weil Sie in einer Konfiguration mit Wählanschluss mit dem tatsächlichen Verschicken warten wollen, bis sich eine nennenswerte Anzahl von Nachrichten angesammelt hat, werden die unzugestellten Nachrichten im Verzeichnis `/var/spool/mqueue` zwischengespeichert. Auch die anderen MTAs verwenden ähnliche Verzeichnisse zum Speichern von unausgelieferten Nachrichten und andere interne Daten – `/var/spool/postfix` bei Postfix, `/var/spool/exim` (oder so etwas) bei Exim oder `/var/qmail` bei Qmail.

Es ist normalerweise eine gute Idee, solche Verzeichnisse – `/var/mail` bzw. `/var/spool/mqueue` & Co. – nicht auf der `/`-Partition unterzubringen, damit eifrig Mail verschickende oder empfangende Benutzer nicht den kompletten Plattenplatz dort auffüllen und so den Systembetrieb gefährden können.

11.4 Verwaltung der Nachrichten-Warteschlange

Noch nicht zugestellte Nachrichten – sei es wegen Fehlern am anderen Ende oder aus lokalem Geiz bei Wählzugängen – legen die MTAs in Warteschlangen ab. Wenn Sie nachschauen wollen, was so alles in der Warteschlange steht, können Sie das bei Sendmail mit dem Kommando `mailq` (Abkürzung für »`sendmail -bp`«) tun.

Dasselbe funktioniert auch für Exim und Postfix, die mit Absicht Sendmail-kompatible Programme für diesen Zweck mitbringen; nur bei Qmail müssen Sie auf dessen eigene Programme zurückgreifen, etwa `qmail-qread` oder `qmail-qstat`.

Warteschlange abarbeiten

Ein als Daemon gestarteter Sendmail kann die Warteschlange in periodischen Abständen abarbeiten. Das ist eine gute Idee, damit Zustellversuche etwa wegen nicht erreichbarer Ziel-MTAs später wiederholt werden können. Mit der `sendmail`-Option `-q` auf der Kommandozeile, unmittelbar gefolgt von einer Zeitangabe, können Sie erreichen, dass Sendmail das tut:

```
# sendmail -bd -q30m
```

startet `sendmail` als Daemon und sorgt alle 30 Minuten für einen Durchlauf durch die Warteschlange.

Sie können Sendmail dazu bringen, die Warteschlange sofort abzuarbeiten, indem Sie ihn mit `-q` (ohne Zeitangabe) aufrufen:

```
# sendmail -q
```

Sendmail versucht dann, alle Nachrichten in der Warteschlange zuzustellen. Das Kommando können Sie automatisch aufrufen, nachdem Ihr Rechner eine Internet-Verbindung hergestellt hat, um Nachrichten an entfernte Empfänger zuzustellen, oder Sie verwenden `cron`, um zu definierten Zeitpunkten eine Verbindung ins Netz aufzubauen und Mail auszuliefern. Auch hier gilt für Postfix und Exim im Wesentlichen das Gleiche; um Qmail dazu zu bringen, seine Warteschlange abzuarbeiten, müssen Sie dem `qmail-send`-Prozess ein `SIGALRM` schicken.

11.5 Lokale Zustellung, Aliasadressen und benutzerspezifische Weiterleitung

Nachrichten an lokale Empfänger schreiben Sendmail & Co. normalerweise in deren Postfach in `/var/mail` (Qmail statt dessen ins Verzeichnis `Maildir` im Heimatverzeichnis). Es gibt aber mehrere Möglichkeiten, von dieser Voreinstellung abzuweichen:

`/etc/aliases`

Die Datei `/etc/aliases` (möglicherweise auch `/etc/mail/aliases`) gestattet es Ihnen, für bestimmte lokale Adressen eine andere Art der Auslieferung zu konfigurieren: Ein Eintrag der Form

Weiterleitung

```
root: hugo
```

leitet an `root` adressierte Nachrichten an den lokalen Benutzer `hugo` weiter – eine sehr sinnvolle Vorgehensweise, da Sie aus Sicherheitsgründen nicht als `root` Mail lesen sollten. Sie können auch an mehrere Zieladressen weiterleiten:

```
hugo: \hugo, hschulz@example.net
```

Hier werden Nachrichten an `hugo` sowohl an die Adresse `hschulz@example.net` weitergeleitet als auch lokal im Postfach von `hugo` abgelegt. Der Rückstrich ist notwendig, damit sich keine Endlosschleife bildet.

Das folgende Beispiel illustriert einige weitere Möglichkeiten von `/etc/aliases`:

```
datei:    /tmp/maildatei.txt
programm: "|/usr/local/bin/programm bla"
liste:    :include:/var/lib/liste.txt
```

Nachrichten an `datei` werden einfach an `/tmp/maildatei.txt` angehängt. Das Format ist dabei dasselbe wie bei den Postfächern in `/var/mail`. Nachrichten an `programm` werden dem Kommando »`/usr/local/bin/programm bla`« auf dessen Standardeingabe übergeben. Und für Nachrichten an `liste` schließlich wird die Datei `/var/lib/liste.txt` konsultiert; jede Zeile in dieser Datei darf eine weitere Umleitung im selben Format wie auf der rechten Seite von `/etc/aliases` definieren. (Dies ist besonders nützlich für »Mailinglisten«, die ihre Abonnentenlisten in solchen Dateien hinterlegen, damit Mailinglistensoftware diese bequem manipulieren kann.)

Auch Postfix und Exim verstehen diese Sorte Alias-Dateien, zum Teil mit Extras; so ist es zum Beispiel möglich, zu bestimmen, ob die Weiterleitung von Nachrichten an Dateien oder Programme nur in `/etc/aliases`, in `:include:`-Listen oder Kombinationen davon erlaubt ist.

Um eine entsprechende Weiterleitung mit Qmail zu definieren, müssen Sie für die Postfachadresse eine Datei in `/var/qmail/alias` anlegen:

Weiterleitung mit Qmail

```
echo &hugo >/var/qmail/alias/.qmail-root
```

leitet alle Mail an `root` an den Benutzer `hugo` weiter. (Das »&« am Anfang zeigt an, dass der Rest der Zeile eine Mailadresse ist.) In Qmail-Aliasdateien können Sie in etwa dasselbe machen wie bei Sendmail & Co.; das Einzige, was nicht direkt unterstützt wird, sind die `:include:`-Listen.

Sendmail & Co. lesen `/etc/aliases` nicht direkt, sondern benutzen ein binäres Datenbankformat, das schnellere Zugriffe zulässt. Die Details sind systemabhängig! Aber damit Ihre Änderungen an `/etc/aliases` wirksam werden, müssen Sie die Datei mit dem Kommando `newaliases` (kurz für »`sendmail -bi`«) ins Binärformat übersetzen.

Binärformat

Auch hier benehmen Postfix und Exim sich im Wesentlichen kompatibel; Postfix hat außerdem ein Programm namens `postalias`, das dasselbe tut. Qmail kocht, wie oben erwähnt, seine eigene Suppe und verwendet kein Binärformat, so dass das Problem sich nicht stellt.

`~/.forward`

Benutzer können für sich selbst ähnliche Einstellungen treffen, indem sie dieselben Weiterleitungen, die auf der rechten Seite von `/etc/aliases` erlaubt sind, in einer Datei namens `.forward` in ihrem Heimatverzeichnis ablegen. Diese Einstellungen befolgt Sendmail, wenn eine Nachricht an den betreffenden Benutzer zugestellt werden soll. Das beliebteste Beispiel ist wohl

```
$ cat /home/hugo/.forward
\hugo, "|/usr/bin/vacation hugo"
```

Das `vacation`-Programm ist ein automatischer »Mailbeantworter«, der eingehende Nachrichten mit einer automatischen Antwort aus `~/.vacation.msg` bescheidet. Damit können Sie zum Beispiel Ihre Korrespondenten darüber informieren, dass Sie wegen eines Urlaubs längere Zeit nicht zur Verfügung stehen.

Postfix und Exim kommen ebenfalls mit `.forward`-Dateien zurecht. Qmail unterstützt ein moralisches Äquivalent unter dem Namen `.qmail-default` (die möglichen Einträge sind subtil verschieden, lesen Sie in `dot-qmail`(5) nach).

Sicherheitsrisiko

Die Zustellung von Nachrichten an beliebige Programme und damit der Fernaufruf beliebiger Programme über das Internet durch Personen ohne »echte« Zugangsberechtigung auf dem System stellt natürlich ein Sicherheitsrisiko dar. Viele Systemverwalter beschränken die Auswahl an in `~/.forward` zulässigen Programmen auf einige als sicher angesehene.

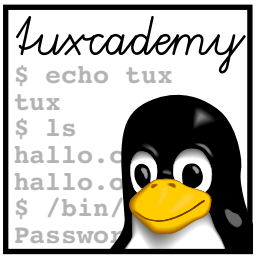
Kommandos in diesem Kapitel

`mailq` Zeigt den Zustand der Mail-Warteschlange an (Sendmail & Co.) `sendmail`(1) 168

`newaliases` Aktualisiert die Alias-Datenbank des Mailservers (Sendmail/Postfix) `newaliases`(1) 169

`sendmail` MTA-Administrationskommando (Sendmail, aber auch andere – kompatible – MTAs) `sendmail`(1) 168

`vacation` Automatischer E-Mail-Beantworter, etwa für längere Abwesenheiten `vacation`(1) 170



12

Grundlagen von GnuPG

Inhalt

12.1 Asymmetrische Kryptografie und das »Netz des Vertrauens« 172
12.2 GnuPG-Schlüssel generieren und verwalten 175
12.2.1 Schlüsselpaare generieren. 175
12.2.2 Öffentliche Schlüssel publizieren 178
12.2.3 Öffentliche Schlüssel importieren und beglaubigen 179
12.3 Daten verschlüsseln und entschlüsseln 183
12.4 Dateien signieren und Signaturen prüfen 184
12.5 GnuPG-Konfiguration 186

Lernziele

- Die Konzepte von GnuPG verstehen
- GnuPG-Schlüssel generieren und verwalten können
- GnuPG zum Verschlüsseln und Entschlüsseln von Dateien verwenden können
- GnuPG zum Signieren von Dateien und Verifizieren von Signaturen verwenden können

Vorkenntnisse

- Bearbeiten von Dateien mit Linux
- Umgang mit einem Texteditor
- Kenntnisse in Kryptografie sind hilfreich

12.1 Asymmetrische Kryptografie und das »Netz des Vertrauens«

Verschlüsseln
Signieren

GnuPG erlaubt das Verschlüsseln und Signieren von Dateien und elektronischer Post. »Verschlüsseln« bedeutet, den Dateiinhalt auf eine Weise unkenntlich zu machen, die ausschließt, dass Unbefugte von ihm Kenntnis nehmen können, gleichzeitig aber Besitzern des korrekten Schlüssels den Zugang ermöglicht. Durch »Signieren« können Sie dokumentieren, eine bestimmte Version einer Datei tatsächlich signiert zu haben. Änderungen an der signierten Datei machen die Signatur ungültig. – GnuPG-Signaturen sind Teil von Paketverwaltungswerkzeugen wie `apt` oder RPM und spielen eine wichtige Rolle in der Authentizitätssicherung von Linux-Distributionen.

Die Verschlüsselung, die GnuPG vornimmt, ist dergestalt, dass selbst gut ausgestattete Geheimdienste nach aktueller Kenntnis nicht in der Position sind, nur aus einem verschlüsselten Text ohne den tatsächlichen Schlüssel den Klartext herausfinden zu können. (Es gibt darum Länder auf der Erde, wo Verschlüsselungssoftware wie GnuPG verboten ist[1]. Das uns naheliegendste Beispiel dafür ist vielleicht Frankreich, wo das von 1996 bis 1999 der Fall war – inzwischen ist man dort zum Glück etwas vernünftiger geworden.)

Welche Interpretation Sie einer digitalen GnuPG-Signatur angedeihen lassen, ist letzten Endes Ihre Sache. In Debian GNU/Linux zum Beispiel signiert ein Debian-Entwickler ein bestimmtes Softwarepaket, bevor er es auf dem Debian-FTP-Server der Allgemeinheit zur Verfügung stellt. Diese Signatur sagt nichts darüber aus, ob die Software im Paket frei von Fehlern ist oder das Paket selbst vollständig oder überhaupt zu irgend etwas zu gebrauchen ist. Die Signatur dient einzig und allein dazu, zu verhindern, dass Personen außerhalb des Kreises der Debian-Entwickler Pakete in die Distributionsinfrastruktur einschmuggeln.

Um die Prinzipien hinter GnuPG erklären zu können, müssen wir etwas weiter ausholen.

Symmetrische Kryptografie

Wir unterscheiden zwei Klassen von kryptografischen Verfahren, also Methoden, Daten zu ver- und entschlüsseln. Bei »symmetrischer Kryptografie« wird zum Verschlüsseln und Entschlüsseln derselbe Schlüssel verwendet. Wollen Alice und Bob[2] mit einem solchen Verfahren vertraulich kommunizieren, müssen sie sich zunächst auf einen gemeinsamen Schlüssel einigen. Kommt noch Charlie dazu, sind schon drei Schlüssel nötig, damit die Paare »Alice/Bob«, »Alice/Charlie« und »Charlie/Bob« jeweils vertraulich kommunizieren können. Bei vier Parteien werden sechs Schlüssel gebraucht, bei zehn Parteien schon fünfundfünfzig und bei hundert 5050. (Die Anzahl der benötigten Schlüssel wächst quadratisch mit der Anzahl der beteiligte Parteien.) Alle diese Schlüssel müssen so generiert und verbreitet werden, dass kein Unbefugter Kenntnis von einem Schlüssel nehmen kann, der ihn nichts angeht.

Schlüsselverteilung

Das Problem bei symmetrischer Kryptografie ist nicht die Kryptografie selbst, sondern die Schlüsselverteilung. Zunächst müssen die Schlüssel natürlich sicher übertragen werden, und dafür wäre Kryptografie schon nützlich. Ferner wäre auf dem Internet mit Millionen von Anwendern die Anzahl von Schlüsseln, die nötig wären, um jeden Teilnehmer vertraulich mit jedem anderen kommunizieren zu lassen, so astronomisch, dass wahrscheinlich kein Platz wäre, um andere Daten zu speichern. Die Lösung des Schlüsselverteilungs-Dilemmas besteht in »asymmetrischer Kryptografie«.

Asymmetrische Kryptografie
Schlüsselpaare

Bei asymmetrischer Kryptografie wird nicht derselbe Schlüssel zum Ver- und Entschlüsseln verwendet. Statt dessen existieren Schlüsselpaare aus je einem

[1] Wie es heißt: "*If cryptography is outlawed, only outlaws will have cryptography.*"

[2] Kryptografen reden immer über »Alice« und »Bob«, wenn sie zwei Parteien meinen, die vertrauliche Daten austauschen wollen.

öffentlichen und einem privaten Schlüssel. Der öffentliche Schlüssel kann weithin bekannt gemacht werden, der private Schlüssel bleibt vertraulich. Alice kann Bob jetzt eine vertrauliche Nachricht schicken, indem sie sie mit Bobs öffentlichem Schlüssel verschlüsselt. Bob verwendet seinen privaten Schlüssel, um die Nachricht zu entschlüsseln, und da nur er den privaten Schlüssel hat, der zu seinem öffentlichen Schlüssel passt, ist die Vertraulichkeit gesichert.

Wir kehren hier ein paar Details unter den Teppich. Das wichtigste ist wohl, dass die gängigen asymmetrischen Kryptoverfahren sich nicht zum Verschlüsseln großer Datenmengen eignen – sie sind einfach zu langsam und unbequem. Statt dessen verwendet man asymmetrische Kryptografie, um *Schlüssel* für symmetrische Kryptoverfahren zu verschlüsseln und zu verschicken und diese Schlüssel später für die eigentliche Kommunikation zu benutzen. (Symmetrische Kryptoverfahren sind sehr schnell und effizient und können beliebig große Datenmengen verschlüsseln.)

Im Klartext heißt das: Wenn Alice mit Bob vertraulich kommunizieren möchte, bestimmt sie einen zufälligen Schlüssel für ein symmetrisches Kryptoverfahren, verschlüsselt diesen Schlüssel – nennen wir ihn mal *S* – mit Bobs öffentlichem Schlüssel und schickt das Ergebnis an Bob. Bob benutzt seinen privaten Schlüssel, um die Nachricht von Alice zu entschlüsseln, und erhält so den Schlüssel *S*. Alice und Bob können jetzt *S* in ihrem symmetrischen Kryptoverfahren verwenden, um nach Herzenslust vertrauliche Daten auszutauschen.

»Hybride« Verschlüsselung

Signaturen funktionieren umgekehrt: Alice kann eine Datei signieren, indem sie sie mit ihrem *privaten* Schlüssel verschlüsselt. Bob kann sich davon überzeugen, dass Alice eine Datei signiert hat, indem er sie mit ihrem öffentlichen Schlüssel (der ihm zur Verfügung steht, denn er ist ja schließlich öffentlich) wieder entschlüsselt. Wenn dabei etwas Sinnvolles herauskommt, muss die Datei vorher von Alice signiert, also mit ihrem privaten Schlüssel verschlüsselt worden sein.

Signatur

In der Praxis signiert man nicht komplette Dateien (dafür sind die asymmetrischen Kryptoverfahren, wie erwähnt, zu plump und zu langsam), sondern »kryptografische Prüfsummen«, die man mit Verfahren wie MD5 oder SHA-1 über den Dateiinhalt gebildet hat. Das ist unter dem Strich genauso gut, da jede Änderung am Dateiinhalt die Prüfsumme ändert und die Signatur damit ungültig wird.

kryptografische Prüfsummen

Asymmetrische Kryptografie löst das Problem der Schlüsselverteilung, aber wirft ein neues Problem auf. Woher weiß Bob, dass der Schlüssel, mit dem er Alices Signatur auf der Datei prüfen möchte, wirklich Alices öffentlicher Schlüssel ist? Vielleicht hat eine bösartige Person (der Jargon nennt sie »Mallory«) ihm einen gefälschten Schlüssel untergeschoben? Was also fehlt, ist eine Methode zur Authentisierung der öffentlichen Schlüssel.

Authentizität von Schlüsseln

Die Secure Shell löst das Problem »zu Fuß«, indem es die Authentisierung der öffentlichen Schlüssel auf den einzelnen Anwender abwälzt. Wie in Kapitel 10 beschrieben wurde, präsentiert `ssh` Ihnen jedesmal, wenn Sie mit einem neuen entfernten Rechner Kontakt aufnehmen, dessen öffentlichen Schlüssel, und es wäre eindeutig an Ihnen, sich von dessen Authentizität zu überzeugen.

Die SSL- bzw. TLS-Infrastruktur, die zum Beispiel für »sichere« Web-Seiten verwendet wird, löst dieses Problem, indem eine Hierarchie von »Zertifizierungsstellen« eingeführt wird (der Fachausdruck ist *public key infrastructure* oder »PKI«). Diese Zertifizierungsstellen bestätigen, dass ein bestimmter öffentlicher Schlüssel zu einer bestimmten Person oder Institution »gehört«, und unterstreichen diese Behauptung durch eine digitale Signatur. Die Gesamtheit aus öffentlichem Schlüssel, dem Namen des Inhabers sowie der digitalen Signatur und

Zertifizierungsstellen

PKI

Zertifikat

den Namen der Zertifizierungsstelle bezeichnet man auch als **Zertifikat**. Sie als Anwender können die Signatur auf dem Zertifikat mit dem öffentlichen Schlüssel der Zertifizierungsstelle prüfen (den Sie sich vorher aus einer vertrauenswürdigen Quelle besorgt haben – typischerweise sind die wichtigsten davon in Ihren Browser eingebaut worden). Wenn alles rein technisch stimmt, dann hängt die Antwort auf die Frage, ob Sie dem öffentlichen Schlüssel aus dem Zertifikat vertrauen können, nur noch davon ab, ob Sie der bestätigenden Zertifizierungsstelle vertrauen, dass die Kollegen dort anständige Arbeit leisten.

Netz des Vertrauens

GnuPG verwendet absichtlich keine solche hierarchische Struktur, sondern setzt auf das »Netz des Vertrauens« (engl. *web of trust*). Es gibt keine zentralisierte Zertifizierungsstelle (oder Menge von Zertifizierungsstellen), der alle Anwender vertrauen müssen, ob sie wollen oder nicht (Für Nicht-Vertrauer wäre es nutzlos, die Infrastruktur zu verwenden). Statt dessen geht das Konzept davon aus, dass Sie als GnuPG-Anwender die öffentlichen Schlüssel von Personen aus Ihrem Umfeld signieren (das GnuPG-Wort ist »beglaubigen«) und damit bestätigen, dass der betreffende Schlüssel zur betreffenden Person gehört. Angenommen, Sie möchten eine vertrauliche Nachricht an Charlie senden, den Sie nicht persönlich kennen. Irgendwo im Netz finden Sie einen öffentlichen Schlüssel, der behauptet, Charlies Schlüssel zu sein. Ob Sie das glauben können, hängt davon ab, ob Sie (bzw. Ihr GnuPG-Programm) eine »Vertrauenskette« von sich selbst zu dem fraglichen öffentlichen Schlüssel konstruieren können: Vielleicht haben Sie Alices öffentlichen Schlüssel beglaubigt, Alice hat dann ihren privaten Schlüssel benutzt, um Bobs öffentlichen Schlüssel zu beglaubigen, und der wiederum ist ein alter Kumpel von Charlie und hat dessen öffentlichen Schlüssel beglaubigt. Mit anderen Worten: Bobs Signatur auf Charlies öffentlichem Schlüssel sagt »Dieser Schlüssel gehört zu Charlie«. Wenn also Bobs Signatur korrekt ist, ist alles in Butter. Bobs Signatur auf Charlies Schlüssel können Sie mit Bobs öffentlichem Schlüssel prüfen, und dass dieser tatsächlich zu Bob gehört, hat Alice mit *ihrer* Signatur auf Bobs öffentlichem Schlüssel bestätigt. Um Alices Signatur zu prüfen, brauchen Sie wiederum Alices öffentlichen Schlüssel, und dass dieser tatsächlich zu Alice gehört, haben Sie selbst mal durch eine Signatur amtlich gemacht. Wenn alle Signaturen in der Kette gültig sind, können Sie also davon ausgehen, dass der öffentliche Schlüssel nicht nur angeblich, sondern auch tatsächlich zu Charlie gehört – »gültig« ist.

Eigentümer-Vertrauen

Für die Praxis ist das beschriebene Verfahren etwas zu blauäugig. GnuPG unterstützt deswegen außerdem noch das Konzept des »Eigentümer-Vertrauens« (engl. *owner trust*): Sie können sich für die öffentlichen Schlüssel in Ihrem Besitz merken, welchen Grad von Pingeligkeit Sie ihren Eigentümern im Umgang mit GnuPG-Schlüsseln zutrauen. Eigentümer-Vertrauen reicht von »voll« (seine Signatur auf einem Schlüssel ist so gut wie Ihre eigene) über »etwas« (der Eigentümer gibt sich redlich Mühe) bis »gar nicht« (er schlampt rum). Zuletzt gibt es noch »unbekannt«. Wohlgemerkt haben diese Kategorien nichts damit zu tun, wie gerne Sie die betreffenden Personen menschlich mögen!

Im Detail ist ein öffentlicher Schlüssel in GnuPG »gültig«, wenn

- er in einer Kette von beglaubigten Schlüsseln höchstens fünf »Schritte« von Ihrem eigenen Schlüssel entfernt ist und
- er mit hinreichend vielen (aus Ihrer Sicht) vertrauenswürdigen Schlüsseln beglaubigt wurde. Das heißt insbesondere entweder,
  - Sie haben ihn selber beglaubigt, oder
  - er wurde mit einem Schlüssel beglaubigt, dem Sie »voll« vertrauen, oder
  - er wurde mit drei Schlüsseln beglaubigt, denen Sie »etwas« vertrauen.

(Die Parameter lassen sich anpassen; hier angegeben sind die Standardwerte.)

Als GnuPG-Anwender, der am Netz des Vertrauens teilnehmen möchte, ist es an Ihnen, nach den Regeln zu spielen. Sie sollten vor allem die öffentlichen Schlüssel anderer Leute nur beglaubigen, wenn Sie sich wirklich über deren Identität sicher sind. Lassen Sie sich im Zweifel einen amtlichen Lichtbildausweis zeigen oder auch zwei – man wird Ihnen diese Gründlichkeit nicht übel nehmen. Beglaubigen Sie niemals Schlüssel nur auf der Basis von Hörensagen!

Übungen

12.1 [!2] Welche Vorteile hat das »Netz des Vertrauens« gegenüber dem PKI-Ansatz? Welche Vorteile hat PKI gegenüber dem »Netz des Vertrauens«?

12.2 [2] Nehmen Sie an, im oben beschriebenen Verfahren zur Bestimmung der »Gültigkeit« von GnuPG-Schlüsseln darf eine Kette zu einem gültigen Schlüssel maximal drei Schritte lang sein, und ein Schlüssel ist gültig, wenn er entweder direkt, mit einem »voll« vertrauten oder zwei »etwas« vertrauten Schlüsseln beglaubigt wurde (nur damit das Beispiel übersichtlicher wird).

Nehmen Sie ferner das Folgende an: Alice hat selbst die öffentlichen Schlüssel von Bob und von Carla beglaubigt. Bob und Carla haben beide den Schlüssel von Doris beglaubigt. Carla und Doris haben außerdem den Schlüssel von Eric beglaubigt.

1. Angenommen, Alice traut Bob »voll« und Carla »etwas«. Welche Schlüssel sind aus der Sicht von Alice »gültig«?
2. Was ändert sich, wenn Alice Doris »etwas« vertraut?
3. Eric beglaubigt den öffentlichen Schlüssel von Fiona. Was muss passieren, damit Fiona Teil des »Netzes des Vertrauens« von Alice wird?

12.2 GnuPG-Schlüssel generieren und verwalten

12.2.1 Schlüsselpaare generieren

Sie selbst brauchen prinzipiell kein GnuPG-Schlüsselpaar, um Nachrichten an andere Benutzer zu verschlüsseln oder die Signaturen anderer Benutzer zu überprüfen. Sie müssen sich nur die passenden öffentlichen Schlüssel besorgen. Allerdings lernen Sie so aus einer erfolgreichen Signaturprüfung nur, dass die Bits in der signierten Datei nicht verfälscht wurden – um Vertrauen aufbauen zu können, dass die Signatur *wirklich* von der behaupteten Person vorgenommen wurde, müssen Sie selbst am »Netz des Vertrauens« teilnehmen, denn nur so können Sie sicherstellen, dass der öffentliche Schlüssel, mit dem Sie die Signatur prüfen, »gültig« ist (nach den Regeln im vorigen Abschnitt). Die ernsthafte Verwendung von GnuPG beginnt also mit dem Erzeugen eines eigenen Schlüsselpaars.

Wenn Sie GnuPG einsetzen, um die Integrität von Paketen Ihrer Distribution zu prüfen, dann stellt Ihr Distributor Ihnen normalerweise öffentliche GnuPG-Schlüssel zum Verifizieren von Paketen zur Verfügung. Wenn diese von einer gepressten (nicht gebrannten) CD-ROM kommen, die Sie im Laden in einer dekorativen Schachtel mit passendem Firmenlogo gekauft haben, stehen die Chancen gut, dass diese öffentlichen Schlüssel authentisch sind. Bei selbstgebrannten CD-ROMs von »Kumpels« sieht das mitunter anders aus.

Schlüsselpaar erzeugen

Sie können ein Schlüsselpaar erzeugen, indem Sie das Programm `gpg` mit der Option `--gen-key` aufrufen:

```
$ gpg --gen-key
gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: Verzeichnis `/home/hugo/.gnupg' erzeugt
gpg: Neue Konfigurationsdatei `/home/hugo/.gnupg/gpg.conf' erstellt
gpg: WARNUNG: Optionen in `/home/hugo/.gnupg/gpg.conf' sind▷
 ◁ während dieses Laufes noch nicht wirksam
gpg: Schlüsselbund `/home/hugo/.gnupg/secring.gpg' erstellt
gpg: Schlüsselbund `/home/hugo/.gnupg/pubring.gpg' erstellt
Bitte wählen Sie, welche Art von Schlüssel Sie möchten:
   (1) DSA und Elgamal (voreingestellt)
   (2) DSA (nur unterschreiben/beglaubigen)
   (5) RSA (nur signieren/beglaubigen)
Ihre Auswahl? 1
Das DSA-Schlüsselpaar wird 1024 Bit haben.
ELG-E-Schlüssel können zwischen 1024 und 4096 Bit lang sein.
Welche Schlüssellänge wünschen Sie? (2048) 2048
Die verlangte Schlüssellänge beträgt 2048 Bit
```

Die Standardoption »DSA und Elgamal« erzeugt tatsächlich *zwei* Schlüsselpaare, ein DSA-Schlüsselpaar für Signaturen und ein untergeordnetes Elgamal-Schlüsselpaar zum Ver- und Entschlüsseln[3]. 2048 Bit sind heutzutage vernünftig. 1024 Bit sind ein bisschen wenig, und 4096 Bit wären fast paranoid (dadurch wird vor allem der Rechenaufwand und die Größe von Signaturen erhöht).

Als nächstes müssen Sie bestimmen, wie lange der Schlüssel gelten soll. Für normale Benutzer spricht in der Regel nichts dagegen, »Schlüssel verfällt nie« zu wählen.

```
Bitte wählen Sie, wie lange der Schlüssel gültig bleiben soll.
         0 = Schlüssel verfällt nie
      <n>  = Schlüssel verfällt nach n Tagen
      <n>w = Schlüssel verfällt nach n Wochen
      <n>m = Schlüssel verfällt nach n Monaten
      <n>y = Schlüssel verfällt nach n Jahren
Wie lange bleibt der Schlüssel gültig? (0) 0
Schlüssel verfällt nie
Ist dies richtig? (j/N) j
```

Sie können die Gültigkeitsdauer nachträglich ändern, müssen dann aber Ihren öffentlichen Schlüssel neu verteilen, damit sich das herumspricht.

Im nächsten Schritt ordnen Sie dem Schlüsselpaar eine »User-ID« zu. Diese besteht aus einem Namen, einer E-Mail-Adresse und einem optionalen Kommentar:

```
Sie benötigen eine User-ID, um Ihren Schlüssel eindeutig zu machen;
das Programm baut diese User-ID aus Ihrem echten Namen, einem
Kommentar und Ihrer Email-Adresse in dieser Form auf:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Ihr Name ("Vorname Nachname"): Hugo Schulz
Email-Adresse: hugo.schulz@example.com
```

[3] »DSA« und »Elgamal« sind die Kryptoverfahren, die GnuPG standardmäßig verwendet. Ersteres ist der (nicht zum Verschlüsseln verwendbare) »Digital Signature Algorithm«, letzteres ein Verfahren, das 1984 von Taher Elgamal beschrieben wurde. Die Sicherheit von Elgamal-Verschlüsselung beruht auf der Beobachtung, dass es schwierig ist, diskrete Logarithmen in endlichen zyklischen Gruppen zu bestimmen – bewiesen ist das aber nicht.

```
Kommentar: Beispiel
Sie benutzen den Zeichensatz `utf-8'
Sie haben diese User-ID gewählt:
  "Hugo Schulz (Beispiel) <hugo.schulz@example.com>"

Ändern: (N)ame, (K)ommentar, (E)-Mail oder (F)ertig/(B)eenden? f
```

Zu guter Letzt müssen Sie noch eine *passphrase* eingeben, mit der die privaten Schlüssel verschlüsselt werden. Wählen Sie einen möglichst langen und unoffensichtlichen Text:

```
Sie benötigen eine Passphrase, um den geheimen Schlüssel zu schützen.

Geben Sie die Passphrase ein: Drei! @ffen essen 96,547 Klöße?
Geben Sie die Passphrase nochmal ein: Drei! @ffen essen 96,547 Klöße?
```

Mit der *passphrase* steht und fällt die Sicherheit von GnuPG – sie ist Ihr einziger Schutz, falls Ihr Schlüsselpaar in die falschen Hände fällt.

Anschließend werden die Schlüsselpaare generiert. Dazu werden Zufallszahlen hoher Qualität gebraucht, die Linux typischerweise aus den Zeitabständen zwischen Tastendrücken, Mausbewegungen und anderen unvorhersehbaren Ereignissen erzeugt:

```
Wir müssen eine ganze Menge Zufallswerte erzeugen.  Sie können dies
unterstützen, indem Sie z.B. in einem anderen Fenster/Konsole
irgendetwas tippen, die Maus verwenden oder irgendwelche anderen
Programme benutzen.
++++++++++....++++++++++..++++++++++.++++++++++++++++++++++++++++++++
◁◁◁◁◁
gpg: /home/hugo/.gnupg/trustdb.gpg: trust-db erzeugt
gpg: Schlüssel 79ECB1F2 ist als uneingeschränkt vertrauenswürdig▷
 ◁ gekennzeichnet
Öffentlichen und geheimen Schlüssel erzeugt und signiert.

gpg: "Trust-DB" wird überprüft
gpg: 3 marginal-needed, 1 complete-needed, PGP Vertrauensmodell
gpg: Tiefe: 0  gültig:   1  unterschrieben:   0  Vertrauen: 0-, 0q,▷
 ◁ 0n, 0m, 0f, 1u
pub   1024D/79ECB1F2 2009-01-04
  Schl.-Fingerabdruck = 4F4C EE34 7B52 DAE1 6ACE  68F7 E9E5 642F 79▷
 ◁ EC B1F2
uid              Hugo Schulz (Beispiel) <hugo.schulz@example.com>
sub   2048g/BBDCE67E 2009-01-04
```

Damit sind Sie auch schon fast fertig.

Als nächstes sollten Sie noch ein »Widerrufszertifikat« erzeugen. Mit dem Widerrufszertifikat können Sie Ihren öffentlichen Schlüssel für ungültig erklären, falls Ihr privater Schlüssel kompromittiert werden oder verloren gehen sollte. Es sagt, dass der öffentliche Schlüssel nicht mehr zur Verschlüsselung neuer Nachrichten verwendet werden sollte; er kann immer noch zum Prüfen existierender Signaturen benutzt werden. Ein Widerrufszertifikat erzeugen Sie so:

Widerrufszertifikat

```
$ gpg --output revoke.asc --gen-revoke "Hugo Schulz"

sec  1024D/79ECB1F2 2009-01-04 Hugo Schulz (Beispiel)▷
 ◁ <hugo.schulz@example.com>

Ein Widerrufszertifikat für diesen Schlüssel erzeugen? (j/N) j
Grund für den Widerruf:
```

```
  0 = Kein Grund angegeben
  1 = Hinweis: Dieser Schlüssel ist nicht mehr sicher
  2 = Schlüssel ist überholt
  3 = Schlüssel wird nicht mehr benutzt
  Q = Abbruch
(Wahrscheinlich möchten Sie hier 1 auswählen)
Ihre Auswahl? 1
Geben Sie eine optionale Beschreibung ein. Beenden mit einer leeren▷
 ◁ Zeile:
> Mein privater Schlüssel wurde kompromittiert oder ist
> verloren gegangen.
>
Grund für Widerruf: Hinweis: Dieser Schlüssel ist nicht mehr sicher
Mein privater Schlüssel wurde kompromittiert oder ist
verloren gegangen.
Ist das OK? (j/N) j
```

(Danach werden Sie noch nach der *passphrase* gefragt.) Das Widerrufszertifikat steht anschließend in der Datei `revoke.asc`.

Sie bauen hier für den GAU vor, nämlich dass Ihr Schlüsselpaar plötzlich und dauerhaft unbenutzbar wird. Wenn Sie Ihren öffentlichen Schlüssel eines Tages aus anderen Gründen (etwa 2 oder 3 im obigen Menü) widerrufen müssen, dann hält Sie niemand davon ab, zu dieser Gelegenheit ein neues (und passenderes) Widerrufszertifikat zu erzeugen.

Sie sollten das Widerrufszertifikat auf einem Medium speichern, das Sie irgendwo einschließen können (nicht auf Ihrem Computer), denn ein Angreifer kann damit Ihren Schlüssel unbrauchbar machen. Am besten drucken Sie es auch auf Papier aus, damit Sie es notfalls per Hand wieder eintippen können (es ist nicht lang).

12.2.2 Öffentliche Schlüssel publizieren

Schlüssel exportieren

Jetzt können Sie daran gehen, Ihren öffentlichen GnuPG-Schlüssel allen Ihren Verwandten, Freunden, Feinden, … bekannt zu machen. Dazu müssen Sie ihn zuerst »exportieren«. Mit der Option `--list-keys` zeigt GnuPG Ihnen alle öffentlichen Schlüssel an, die Sie derzeit zur Verfügung haben (bisher nur Ihren eigenen):

```
$ gpg --list-keys
/home/hugo/.gnupg/pubring.gpg
------------------------------
pub   1024D/79ECB1F2 2009-01-04
uid                  Hugo Schulz (Beispiel) <hugo.schulz@example.com>
sub   2048g/BBDCE67E 2009-01-04
```

Exportieren können Sie einen öffentlichen Schlüssel mit der Option `--export`. Wie bei der Erzeugung des Widerrufszertifikats können Sie hier mit `--output` einen Dateinamen angeben, unter dem der exportierte Schlüssel abgelegt werden soll. Die Option `--armor` gibt den Schlüssel in einem ASCII-basierten Format aus, das sich dazu eignet, mit elektronischer Post verschickt oder auf einer Web-Seite veröffentlicht zu werden:

```
$ gpg --output hugo.gpg --armor --export "Hugo Schulz"
$ cat hugo.gpg
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.9 (GNU/Linux)

mQGiBElg/q0RBADYvjm+m6f1dwV0Kl6lym2JsmeDYHPRyArtVc5/xLoBkEUR+nLz
```

```
M6stV/2MNcNTSUzIGjAWc4i//vQ09OUR3HdSooEQa/7ZwLd8Wq4G7I+uz74R/txL
cA8uq/fUl4UzBnU+oH4qeRon9tJHrvJeHxvbSf1WMHId5/znpxORu0ZMmwCg1ojA
XYNUoJHZB8BVI3QBSqrWjGcD/jbBUMgyzhzQJvkS3GFD9tL6SxB+knjdTLr9R/8m
=KKKK
```

Eine effizientere Methode zum Publizieren öffentlicher Schlüssel besteht darin, sie auf einen »Keyserver« hochzuladen. Das schaffen Sie mit einem Kommando wie Keyserver

```
$ gpg --keyserver hkp://subkeys.pgp.net --send-keys 79ECB1F2
```

Die meisten Keyserver synchronisieren sich untereinander, so dass es normalerweise genügt, wenn Sie Ihren öffentlichen Schlüssel auf einen davon hochladen. Die magische Kennung `79ECB1F2` ist die »Key-ID«, die Sie zum Beispiel aus der Ausgabe von »`gpg --list-keys` entnehmen können.

12.2.3 Öffentliche Schlüssel importieren und beglaubigen

Einen öffentlichen Schlüssel, den Sie von anderswo bekommen – etwa per Mail oder über eine Web-Seite –, müssen Sie in Ihr »Schlüsselbund« importieren, bevor Sie ihn verwenden können. Das geht so:

```
$ gpg --import /tmp/susi.gpg
gpg: Schlüssel 5526DE34: Öffentlicher Schlüssel "Susi Sorglos>
 < (Kein Kommentar!) <susi.sorglos@example.net>" importiert
gpg: Anzahl insgesamt bearbeiteter Schlüssel: 1
gpg:                              importiert: 1
$ gpg --list-keys
/home/hugo/.gnupg/pubring.gpg
------------------------------
pub   1024D/79ECB1F2 2009-01-04
uid                  Hugo Schulz (Beispiel) <hugo.schulz@example.com>
sub   2048g/BBDCE67E 2009-01-04

pub   1024D/5526DE34 2009-01-04
uid                  Susi Sorglos (Kein Kommentar!)>
 < <susi.sorglos@example.net>
sub   2048g/045B6B4F 2009-01-04
```

Damit steht der Schlüssel in Ihrem Schlüsselbund, aber Sie sind noch nicht fertig – woher wollen Sie wissen, dass das, was Sie gerade importiert haben, wirklich der öffentliche Schlüssel von Susi Sorglos ist? Sie sollten das unbedingt überprüfen. Am besten besorgen Sie sich den »Fingerabdruck« (engl. *fingerprint*) des neuen Schlüssels und fragen Susi, ob er stimmt: Fingerabdruck

```
$ gpg --fingerprint "Susi Sorglos"
pub   1024D/5526DE34 2009-01-04
  Schl.-Fingerabdruck = D533 80FD B930 1BCD 2F6B  4C25 F92F 82B0>
 < 5526 DE34
uid                  Susi Sorglos (Kein Kommentar!)>
 < <susi.sorglos@example.net>
sub   2048g/045B6B4F 2009-01-04
```

Zur Überprüfung können Sie Susi zum Beispiel anrufen (wenn Sie zu 100% sicher sind, dass Sie sie an ihrer Telefonstimme zweifelsfrei erkennen können). Das funktioniert natürlich am besten, wenn Sie ihre Telefonnummer aus einer anderen Quelle haben als der Mail oder Webseite, die auch den Schlüssel enthält, um den es gerade geht …

Sie können sich den Fingerabdruck Ihres GnuPG-Schlüssels auch auf Ihre Visitenkarte drucken (lassen) oder sich Zettel mit den Schlüsseldaten präparieren. Wenn Sie dann jemanden treffen, mit dem Sie Signaturen »tauschen« wollen, können Sie ihm Ihren GnuPG-Fingerabdruck geben und (falls nötig) Ihren Personalausweis zeigen. Damit kann er sich davon überzeugen, dass der öffentliche Schlüssel mit dem betreffenden Fingerabdruck tatsächlich Ihrer ist und später (wenn er wieder am Rechner sitzt) Ihren öffentlichen Schlüssel vom Keyserver holen, beglaubigen und Ihnen schicken, damit Sie die neue Signatur in Ihren »offiziellen« öffentlichen Schlüssel integrieren können.

key signing parties

Ebenfalls populär sind *key signing parties*, bei denen zum Beispiel die Mitglieder von Open-Source-Projekten wie Debian ein gegenseitiges Massen-Überprüfen von GnuPG-Schlüsseln vornehmen. Dafür werden vorher die Schlüsseldaten (User-IDs und Fingerabdrücke) der Beteiligten gesammelt und an alle auf Papier verteilt. Sie lassen sich dann reihum die Personalausweise, Pässe, … zeigen und markieren diejenigen Schlüssel, von deren Authentizität Sie überzeugt sind. (Beglaubigt werden die Schlüssel wieder erst später in Ruhe.)

Schlüssel beglaubigen

Nehmen wir mal an, Sie haben sich zu Ihrer Zufriedenheit von der Authentizität von Susi Sorglos' öffentlichem Schlüssel überzeugt (die Details überlassen wir Ihnen). Jetzt können Sie Susis Schlüssel beglaubigen. Das ist zuerst einmal gut für Susi, da ihr Schlüssel damit von allen Leuten als authentisch akzeptiert wird, die *Ihren* Schlüssel als gültig ansehen (unter den Einschränkungen von Abschnitt 12.1). Umgekehrt profitieren Sie aber auch selbst davon, da Sie an Susis »Netz des Vertrauens« angeschlossen werden – Schlüssel, die Susi beglaubigt hat, können Sie auch als gültig akzeptieren.

Natürlich kommt auch noch das »Eigentümer-Vertrauen« ins Spiel. Sie können in Ihrem Schlüsselbund angeben, wie sehr Sie Susis GnuPG-Kenntnissen und ihrer Gründlichkeit vertrauen wollen, und das wirkt sich auf die Gültigkeit von Schlüsseln aus, die Susi beglaubigt hat. Wenn Sie bezüglich Susi so Ihre Zweifel hegen, dann müssen die von Susi beglaubigten Schlüssel auch noch von anderen Leuten in Ihrem »Netz des Vertrauens« beglaubigt worden sein, damit Sie sie als gültig anerkennen – wenn Sie jedoch eine hohe Meinung von Susi haben, dann reicht Susis Signatur alleine aus.

Susis Schlüssel beglaubigen Sie wie folgt:

```
$ gpg --sign-key "Susi Sorglos"

pub 1024D/5526DE34  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: SC
                    Vertrauen: unbekannt     Gültigkeit: unbekannt
sub 2048g/045B6B4F  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: E
[  unbek.] (1). Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>


pub 1024D/5526DE34  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: SC
                    Vertrauen: unbekannt     Gültigkeit: unbekannt
 Haupt-Fingerabdruck  = D533 80FD B930 1BCD 2F6B  4C25 F92F 82B0 5526▷
 ◁ DE34

     Susi Sorglos (Kein Kommentar!) <susi.sorglos@example.net>

Sind Sie wirklich sicher, daß Sie vorstehenden Schlüssel mit Ihrem
Schlüssel "Hugo Schulz (Beispiel) <hugo.schulz@example.com>"▷
 ◁ (79ECB1F2) beglaubigen wollen
```

```
Wirklich unterschreiben? (j/N) j
```

(Es folgt wieder die Abfrage der *passphrase*.)

Signaturen überprüfen

Die Signaturen auf einem Schlüssel können Sie mit der Option `--check-sigs` anzeigen und überprüfen:

```
$ gpg --check-sigs "Susi Sorglos"
pub   1024D/5526DE34 2009-01-04
uid          Susi Sorglos (Kein Kommentar!) <susi.sorglos@example.net>
sig!3        5526DE34 2009-01-04  Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>
sig!          79ECB1F2 2009-01-04  Hugo Schulz (Beispiel)▷
 ◁ <hugo.schulz@example.com>
sub   2048g/045B6B4F 2009-01-04
sig!          5526DE34 2009-01-04  Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>
```

Sie sehen hier die Signatur von Susi selbst (wird automatisch beim Erstellen des Schlüssels angelegt) und Ihre eigene. Das Ausrufungszeichen hinter dem `sig` am Zeilenanfang heißt, dass die Signatur erfolgreich geprüft werden konnte; bei »schlechten« Signaturen stünde dort ein »-«.

Sie können alle diese Signaturen tatsächlich prüfen, weil Sie die nötigen gültigen öffentlichen Schlüssel zur Verfügung haben (auch wenn die Katze sich da irgendwo in den Schwanz beißt). Fehlt ein öffentlicher Schlüssel für die Überprüfung, dann wird die betreffende Signatur ignoriert.

Mit »`gpg --list-sigs`« können Sie sich alle Signaturen ohne Prüfung anschauen; fehlende öffentliche Schlüssel erscheinen dabei als »`[User-ID nicht gefunden]`«.

Zur Verwaltung von Schlüsseln können Sie auch »`gpg --edit-key`« verwenden:

```
$ gpg --edit-key "Susi Sorglos"
gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


pub 1024D/5526DE34  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: SC
                    Vertrauen: unbekannt     Gültigkeit: vollständig
sub 2048g/045B6B4F  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: E
[ vollst.] (1). Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>

Befehl> _
```

Hier können Sie Befehle eingeben. »`fpr`« zeigt den Fingerabdruck des Schlüssels an und mit »`sign`« können Sie ihn signieren. Es gibt jede Menge andere Befehle, »`help`« zeigt eine Liste an. – Wichtig ist der Befehl »`trust`«, mit dem Sie das Eigentümer-Vertrauen setzen können:

```
Befehl> trust
pub 1024D/5526DE34  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: SC
                    Vertrauen: unbestimmt    Gültigkeit: vollständig
sub 2048g/045B6B4F  erzeugt: 2009-01-04  verfällt: niemals  Aufruf: E
[ vollst.] (1). Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>
```

```
Bitte entscheiden Sie, in wieweit Sie diesem User zutrauen,
Schlüssel anderer User korrekt zu prüfen (durch Vergleich
mit Lichtbildausweisen, Vergleich der Fingerabdrücke aus
unterschiedlichen Quellen ...)?

 1 = Weiß nicht so recht                    Standardfall, wenn nichts anderes gesagt
 2 = Nein, ihm traue ich NICHT                                        Schlampt rum
 3 = Ich vertraue ihm marginal                     So, so, la, la, aber fundamental O. K.
 4 = Ich vertraue ihm vollständig                    Kennt sich aus und passt gut auf
 5 = Ich vertraue ihm absolut                                  Das sind Sie nur selbst
 m = Zurück zum Menü

Ihre Auswahl? 4
```

Der Unterschied zwischen »vollständig« und »absolut« im Auswahlmenü ist der zwischen den Leuten, denen Sie vollständig vertrauen, und Ihnen selbst – »absolut« vertrauen sollten Sie nur Ihren eigenen Schlüsseln.

Die »Vertrauensdatenbank« von GnuPG wird erst beim nächsten Start des Programms auf den neuesten Stand gebracht (das ist ein einigermaßen aufwendiger Prozess). Wenn Sie in großem Stil Eigentümer-Vertrauen anpassen wollen, sollten Sie das also in einem Rutsch machen und dann `gpg` neu starten.

Sollten Sie jemals in die Verlegenheit kommen, Ihren öffentlichen Schlüssel widerrufen zu müssen (Sie haben doch weiter oben ein Widerrufszertifikat angelegt, nicht wahr?), dann müssen Sie damit anfangen, das Widerrufszertifikat in Ihren Schlüsselbund zu importieren:

```
$ gpg --import revoke.asc
```

Anschließend können Sie den widerrufenen Schlüssel an die Keyserver schicken.

Übungen

12.3 [!2] Legen Sie wie in Abschnitt 12.2.1 beschrieben ein Schlüsselpaar an. Vergewissern Sie sich, dass der öffentliche Schlüssel im Schlüsselbund existiert und mit sich selbst beglaubigt ist (das passiert automatisch beim Anlegen).

12.4 [!2] Legen Sie unter einem anderen Benutzerkonto ein weiteres Schlüsselpaar an. Exportieren Sie den öffentlichen Schlüssel und importieren Sie ihn in Ihren Schlüsselbund. Vergewissern Sie sich, dass er dort angekommen ist. (*In Kursen:* Arbeiten Sie mit einem anderen Teilnehmer zusammen und sparen Sie sich die Mühe des Schlüsselpaar-Anlegens.)

12.5 [2] Benutzen Sie »`gpg --sign-key`« oder »`gpg --edit-key`«, um den in Übung 12.4 importierten Schlüssel zu beglaubigen. Prüfen Sie die Signatur(en) auf dem öffentlichen Schlüssel. Exportieren Sie den Schlüssel und importieren Sie ihn in den Schlüsselbund des »Eigentümers« (im anderen Benutzerkonto).

12.6 [3] Benutzen Sie ein drittes Benutzerkonto (ja, ja, es ist wirklich nur zu Ihrem Besten!), um ein drittes GnuPG-Schlüsselpaar anzulegen. Importieren Sie den in Übung 12.5 signierten öffentlichen Schlüssel in den dortigen Schlüsselbund. Was ist die Ausgabe von »`gpg --check-sigs`«? Ist der Schlüssel aus der Sicht des dritten Benutzerkontos »gültig«? Wenn nicht, was müßten Sie tun, um ihn gültig zu machen?

12.7 [2] Weisen Sie im Benutzerkonto aus Übung 12.6 mit »`gpg --edit-key`« dem gerade importierten öffentlichen Schlüssel »volles« Eigentümer-Vertrauen zu. Welche Auswirkungen hat das auf dessen »Gültigkeit«?

12.3 Daten verschlüsseln und entschlüsseln

Um eine Datei mit GnuPG zu verschlüsseln, brauchen Sie die öffentlichen Schlüssel der Empfänger der Datei. Sie können sich einen öffentlichen Schlüssel vorstellen wie einen offenen Tresor: Sie verschlüsseln die Daten, indem Sie sie in den Tresor legen, die Tür zuklappen und das Rad des Kombinationsschlosses ein paar Mal herumdrehen. Anschließend kann nur derjenige, der die Kombination (den privaten Schlüssel) kennt, den Tresor wieder öffnen und die Daten lesen (entschlüsseln).

Wenn Sie also eine Datei verschlüsseln und zum Beispiel an Susi Sorglos schicken wollen, brauchen Sie den öffentlichen Schlüssel von Susi Sorglos. Susi kann die Datei mit ihrem privaten Schlüssel entschlüsseln und braucht wiederum Ihren öffentlichen Schlüssel, wenn sie Ihnen eine Antwort schicken möchte (die Sie dann mit Ihrem privaten Schlüssel entschlüsseln können).

Mit GnuPG geht das so:

```
$ gpg --output text.txt.gpg --encrypt \
>    --recipient "Susi Sorglos" text.txt
```

Dieses Kommando verschlüsselt die Datei `text.txt` so, dass Susi Sorglos sie lesen kann. Die verschlüsselte Version steht in `text.txt.gpg` (dem Parameter der Option `--output`).

Auch hier können Sie die Option `--armor` verwenden, um die verschlüsselte Version in einer ASCII-Fassung zu erhalten, die Sie zum Beispiel mit elektronischer Post verschicken können.

Sie können eine Datei auch so verschlüsseln, dass mehrere Empfänger sie lesen können. Dazu geben Sie einfach die `--recipient`-Option so oft an, wie es Empfänger gibt. Allerdings führt das dazu, dass die Ausgabe für jeden Empfänger eine eigene verschlüsselte Version der Datei enthält.

Susi Sorglos kann die verschlüsselte Datei wie folgt mit ihrem privaten Schlüssel entschlüsseln:

```
susi$ gpg --output text.txt --decrypt text.txt.gpg

Sie benötigen eine Passphrase, um den geheimen Schlüssel zu entsperren.
Benutzer: "Susi Sorglos (Kein Kommentar!) <susi.sorglos@example.net>"
2048-Bit ELG-E Schlüssel, ID 045B6B4F, erzeugt 2009-01-04▷
 ◁ (Hauptschlüssel-ID 5526DE34)

Geben Sie die Passphrase ein: DonauDampfSchiff
gpg: verschlüsselt mit 2048-Bit ELG-E Schlüssel, ID 045B6B4F,▷
 ◁ erzeugt 2009-01-04
      "Susi Sorglos (Kein Kommentar!) <susi.sorglos@example.net>"
```

Wenn Sie eine Datei nur für sich selbst verschlüsseln wollen, brauchen Sie keine asymmetrische Kryptografie. Mit dem Kommando »`gpg --symmetric`« wird ein Schlüssel verwendet, der von einer *passphrase* abgeleitet wird, die Sie als nächstes eingeben. Dieselbe *passphrase* dient auch zum Entschlüsseln:

```
$ echo "Hallo Welt" | gpg --output hw.gpg --symmetric
Geben Sie die Passphrase ein: blafasel
```

```
Geben Sie die Passphrase nochmal ein: blafasel
$ gpg --decrypt hw.gpg
gpg: CAST5 verschlüsselte Daten
Geben Sie die Passphrase ein: blafasel
gpg: Verschlüsselt mit einer Passphrase
Hallo Welt
gpg: WARNUNG: Botschaft wurde nicht integritätsgeschützt▷
 ◁ (integrity protected)
```

(Natürlich sollten Sie nicht dieselbe *passphrase* benutzen, die Sie sonst für Ihre privaten GnuPG-Schlüssel verwenden.)

Dieses Beispiel illustriert auch, dass GnuPG das zu verschlüsselnde Material von der Standardeingabe lesen (das erste Kommando) und die entschlüsselten Daten auf der Standardausgabe liefert (das zweite Kommando).

Übungen

12.8 [!2] Verschlüsseln Sie eine beliebige Datei so, dass Sie sie mit dem Schlüssel aus Übung 12.4 entschlüsseln können. Vergewissern Sie sich, dass das funktioniert.

12.9 [1] Verwenden Sie die »symmetrische« Verschlüsselung, um eine beliebige Datei zu verschlüsseln und wieder zu entschlüsseln. Überzeugen Sie sich, dass das Ergebnis dem Original entspricht.

12.4 Dateien signieren und Signaturen prüfen

Mit Ihrem Schlüsselpaar können Sie eine Datei auch *signieren*. Damit wird die Datei zum einen mit einem Zeitstempel versehen, zum anderen führen Veränderungen an der Datei dazu, dass eine Prüfung der Signatur fehlschlägt. Das ist zum Beispiel sehr wichtig, um sicherzustellen, dass Pakete Ihrer Distribution, die Sie herunterladen und installieren wollen, nicht von übelwollenden Crackern manipuliert worden sind und Viren oder trojanische Pferde enthalten oder bei der nächsten Gelegenheit Ihr Kennwort und Ihre Kreditkartennummer nach Russland schicken. Oder Sie schicken Ihr Buchmanuskript digital signiert an Ihren Verleger, der mit Ihrem öffentlichen Schlüssel prüfen kann, dass das Manuskript tatsächlich von Ihnen kommt und auf dem Weg nicht verändert wurde.

Als Nebeneffekt kommt dazu, dass es für Sie sehr schwierig ist, in Abrede zu stellen, dass Sie ein Dokument mit Ihrer digitalen Signatur *nicht* selber signiert haben – das würde implizieren, dass Ihr privater Schlüssel kompromittiert wurde.

Unterschiede zwischen digitalen Signaturen und Unterschriften

Es gibt eine gewisse Tendenz, digitale Signaturen, so wie Sie sie mit GnuPG anlegen können, auf dieselbe Stufe zu stellen wie Unterschriften, die Sie auf Papier leisten. Das ist unglücklich bis gefährlich, denn es gibt große qualitative Unterschiede zwischen dem Akt einer Unterschrift auf einem Blatt Papier und dem Akt, eine *passphrase* einzugeben und auf ↩ zu drücken. Zum Beispiel unterschreiben Sie auf dem Papier genau das, was weiter oben auf dem Papier steht, und sonst nichts anderes. Im Zweifelsfall wird mit einiger Berechtigung davon ausgegangen, dass Sie das auch gemeint haben. Auf der anderen Seite haben Sie, jedenfalls so wie die entsprechenden Systeme heute ausgelegt sind, nicht die geringste Garantie außer einem warmen pelzigen Gefühl im Bauch, dass Ihr Computer *wirklich* das richtige Dokument signiert (und *nur* dieses Dokument und nicht gleichzeitig auch noch einen Kaufvertrag für eine Industriewaschmaschine). In einem Rechtsstreit wird ein kompetenter Sachverständiger demnach die Behauptung »Aber er hat

es doch digital signiert!« völlig zerpflücken können, ohne dabei in Schweiß zu geraten. Das ist kein wünschenswerter Zustand.

Im einfachsten Fall signieren Sie mit GnuPG eine Datei über das Kommando »`gpg --sign`«:

```
$ gpg --output text.txt.sig --sign text.txt
```

(Da das Signieren Ihren privaten Schlüssel involviert, müssen Sie wie üblich Ihre *passphrase* angeben.) Dabei wird die Datei gleichzeitig auch noch komprimiert und steht anschließend in einem binären Format in der Ausgabedatei.

Prüfen können Sie die Signatur mit »`gpg --verify`«:

```
$ gpg --verify text.txt.sig
gpg: Unterschrift vom Mo 05 Jan 2009 00:52:26 CET mittels▷
 ◁ DSA-Schlüssel ID 5526DE34
gpg: Korrekte Unterschrift von "Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>"
```

Mit »`gpg --decrypt`« wird nicht nur die Signatur geprüft, sondern die Datei auch entkomprimiert und auf der Standardausgabe ausgegeben:

```
$ gpg --decrypt text.txt.sig
Hallo Welt
gpg: Unterschrift vom Mo 05 Jan 2009 00:52:26 CET mittels▷
 ◁ DSA-Schlüssel ID 5526DE34
gpg: Korrekte Unterschrift von "Susi Sorglos (Kein Kommentar!)▷
 ◁ <susi.sorglos@example.net>"
```

Mit der Option `--output` können Sie die entkomprimierten Daten wie gewohnt auch in eine Datei schreiben lassen.

Das komprimierte Binärformat ist unpraktisch, wenn Sie die signierte Datei anschließend verschicken möchten. In diesem Fall verwenden Sie besser »`gpg --clearsign`«, um eine Ausgabedatei zu erhalten, in der Dokument und Signatur im ASCII-Format stehen:

Klartext-Signatur

```
$ gpg --clearsign text.txt
```

Das Ergebnis sieht dann zum Beispiel so aus:

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

Hallo Welt
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (GNU/Linux)

iEYEARECAAYFAklhTWsACgkQ+S+CsFUm3jQWxQCbB2vSOH/lyK55ELrDD6pwbVwP
hTwAoK5pqDNb9GxkENdBk2q0LPtrD1dV
=PExu
-----END PGP SIGNATURE-----
```

Im Klartext signierte Dateien werden zum Beispiel im Debian-Projekt verwendet. Die `.dsc`-Dateien, die den Quellcode eines Pakets begleiten, enthalten MD5-Prüfsummen über den originalen Quellcode (typischerweise eine `.tar.gz`-Datei) und den Patch mit den Debian-spezifischen Anpassungen, und werden vom Paketbetreuer mit GnuPG signiert. Damit können Sie als Empfänger des Paketquellcodes sicherstellen, dass Sie die Version des Paketbetreuers erhalten haben. Ebenso muss eine neue Version eines Pakets,

die auf den Debian-Servern als Teil der Distribution zur Verfügung gestellt werden soll, von einer GnuPG-signierten `.changes`-Datei begleitet werden, die die Dateien des Pakets mit ihren MD5-Prüfsummen auflistet. Dateien, die hochgeladen werden, ohne dass eine `.changes`-Datei dabei ist, die von einem Debian-Entwickler signiert wurde, werden verworfen.

Das »`PGP`« in der ASCII-Ausgabe von GnuPG ist übrigens kein Druckfehler. GnuPG ist eine Implementierung des »OpenPGP«-Standards [RFC4880], der auf ein Programm namens »Pretty Good Privacy« von Phil Zimmermann zurückgeht. »GnuPG«, kurz für »GNU Privacy Guard«, ist ein Wortspiel.

Um Dateien zu signieren, ohne sie dabei zu verändern – etwa wenn Sie ein Paket für eine Distribution einreichen oder die Authentizität eines Quellcode-Archivs sichern wollen, das Sie veröffentlichen, und die üblichen GnuPG-Formate dazu führen würden, dass die Datei nicht mehr mit Programmen wie `tar` verarbeitet werden kann –, können Sie eine »abgetrennte Signatur« (engl. *detached signature*) erzeugen. Dazu verwenden Sie das Kommando »`gpg --detach-sign`«:

abgetrennte Signatur

```
$ gpg --output text.txt.sig --detach-sign text.txt
```

Zum Prüfen einer abgetrennten Signatur brauchen Sie sowohl die signierte Datei als auch die Datei mit der Signatur (das ist logisch, da Sie bei der Prüfung ja nachvollziehen wollen, dass der Inhalt der signierten Datei sich nicht geändert hat). Verwenden Sie dazu »`gpg --verify`«:

```
$ gpg --verify text.txt.sig text.txt
```

Dabei ist der Name der Datei mit der abgetrennten Signatur immer der erste Parameter hinter der Option `--verify`. Sie können den Namen der signierten Datei als zweiten Parameter angeben; GnuPG nimmt sonst an, dass die signierte Datei so heißt wie die Datei mit der abgetrennten Signatur, aber ohne das `.sig` (oder `.asc`, wenn Sie `--armor` benutzen) am Ende.

Übungen

12.10 [!1] Signieren Sie eine beliebige Datei mit GnuPG. Überprüfen Sie anschließend die Gültigkeit der Signatur.

12.11 [!2] Verwenden Sie den Schlüssel aus Übung 12.5, um eine Datei zu signieren. Kopieren Sie sie anschließend in Ihr eigenes Benutzerkonto und verwenden Sie Ihren Schlüsselbund, um die Signatur zu überprüfen. Was ist das Ergebnis?

12.12 [2] Wiederholen Sie Übung 12.11 mit dem Benutzerkonto und dem Schlüsselpaar aus Übung 12.6. Was passiert?

12.5 GnuPG-Konfiguration

`~/.gnupg` GnuPG bewahrt seine Konfigurationsdateien im Verzeichnis `~/.gnupg` auf. Einige typischerweise dort vorhandene Dateien sind:

`pubring.gpg` Das »öffentliche Schlüsselbund«. Hier werden die öffentlichen Schlüssel abgelegt, die Sie importiert (oder erzeugt) haben.

`secring.gpg` Das »geheime Schlüsselbund«. Hier stehen die privaten Schlüssel für Ihre Schlüsselpaare.

trustdb.gpg Diese Datei enthält die Daten für das Eigentümer-Vertrauen für öffentliche Schlüssel. Da es sich dabei um Ihre privaten Ansichten handelt, werden diese Informationen nicht in einer Schlüsselbund-Datei gespeichert, die Sie möglicherweise anderen Personen zugänglich machen wollen.

random_seed In dieser Datei merkt sich GnuPG seine internen Ausgangsdaten für die Zufallszahlengenerierung.

gpg.conf Diese Datei enthält Voreinstellungen für `gpg`. Die Datei ist sehr ausführlich kommentiert.

Einige Einstellungen, die Sie in `gpg.conf` machen könnten, sind zum Beispiel: `gpg.conf`

keyserver Gibt einen URL für Ihren bevorzugten Keyserver an, zum Beispiel

```
keyserver hkp://keys.gnupg.net
```

no-greeting Unterdrückt die Copyright-Notiz, die `gpg` sonst beim Start ausgibt.

group Erlaubt die Definition von »Gruppen«, um Tipparbeit bei der Verschlüsselung zu sparen. Nehmen wir an, Ihre `gpg.conf`-Datei enthält den Eintrag Gruppen

```
group beatles = john paul 0x12345678 ringo
```

Wenn Sie die Option »`--recipient beatles`« angeben, interpretiert GnuPG das so, als ob sie die Option dreimal mit den Namen und einmal mit der Key-ID angegeben hätten.


Das ganze funktioniert nur eine Ebene tief. Sie können also keine Gruppen haben, die andere Gruppen zusammenfassen.

GnuPG ist ein wichtiges, aber auch nicht ganz unkompliziertes Programm. Sie sollten die Dokumentation in `gpg`(1) studieren; außerdem ist ausführliche Dokumentation auf `http://www.gnupg.org/` zu finden.

Übungen


12.13 [2] Definieren Sie in Ihrer Datei `gpg.conf` eine Gruppe, die die Schlüssel aus Übung 12.5 und Übung 12.6 enthält. Verschlüsseln Sie eine Datei, indem Sie den Namen der Gruppe als Empfänger angeben. Vergewissern Sie sich, dass beide Empfänger die Datei tatsächlich entschlüsseln können.


12.14 [5] Lesen Sie [Keysigning-Party-HOWTO] und organisieren Sie eine *key signing party* mit einer geeigneten Gruppe von Leuten (Kursteilnehmer, Kollegen, Bekannte). (Wie Sie das Wort *party* auslegen, ist Ihre Sache, aber es sollte auf jeden Fall auch *key signing* stattfinden.)

Kommandos in diesem Kapitel

gpg Verschlüsselt und signiert Dateien `gpg`(1) 175

Zusammenfassung

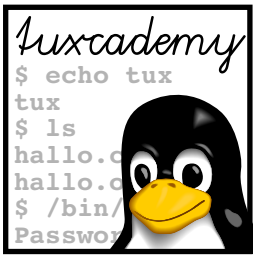
- GnuPG erlaubt das Verschlüsseln und Signieren von Dateien und elektronischer Post.
- Die Authentizität öffentlicher Schlüssel wird bei GnuPG über ein »Netz des Vertrauens« hergestellt.

Literaturverzeichnis

Ash99 John Michael Ashley. »The GNU Privacy Handbook«, 1999.
http://www.gnupg.org/gph/de/manual.pdf

Keysigning-Party-HOWTO V. Alex Brennen. »The Keysigning Party HOWTO«, Januar 2008.
http://www.cryptnet.net/fdp/crypto/keysigning_party/en/keysigning_party.html

RFC4880 J. Callas, L. Donnerhacke, H. Finney, et al. »OpenPGP Message Format«, November 2007. http://www.ietf.org/rfc/rfc4880.txt



13

Linux und Sicherheit: Ein Einstieg

Inhalt

13.1 Einführung 190
13.2 Sicherheit im Dateisystem 190
13.3 Benutzer und Dateien 193
13.4 Ressourcenlimits 197
13.5 Administratorprivilegien mit `sudo` 201
13.6 Grundlegende Netzsicherheit 205

Lernziele

- Das Dateisystem nach sicherheitsrelevanten Einträgen (Gerätedateien, Dateien mit gesetztem SUID/SGID-Bit) absuchen können
- Ressourcenlimits verstehen und einsetzen können
- `sudo` konfigurieren und einsetzen können
- Grundlegende Probleme der Netzsicherheit verstehen

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 4)

13.1 Einführung

»Sicherheit« ist ein wichtiges Thema für alle, die Rechner betreiben – und leider im Zeitalter des Internet nicht beschränkt auf Systemadministratoren bei großen Firmen, Universitäten und Internet-Providern. Selbst Ihr heimischer PC, auf dem nur Sie und Ihre Lieben ein Benutzerkonto haben, kann »Sprungbrett« für Cracker werden, die Ihren schnellen DSL-Anschluss verwenden, um Spam oder Viren in alle Welt zu verschicken oder sich (und damit Sie) anderweitig unpopulär zu machen. Zugegebenermaßen haben Sie mit Ihrer Entscheidung für Linux schon den allergrößten Teil dieser Gefahr ausgemerzt, aber das ist sicherlich kein Grund, sich auf seinen Lorbeeren auszuruhen[1].

Sicherheit unter Linux ist ein vielschichtiges und anspruchsvolles Thema, bei dem hier nur an der Oberfläche kratzen können. Insbesondere wenn Sie im betrieblichen Umfeld ein System verwalten, auf dem Intranet- oder Internet-Dienste erbracht werden, möchten wir Ihnen die Linup-Front-Schulungsunterlage *Linux-Sicherheit* nahelegen, die Sie wesentlich detaillierter in die Problematik einführt. Bis dahin erklären wir Ihnen in diesem Kapitel einige wichtige Anfangsgründe.

13.2 Sicherheit im Dateisystem

Wenn man die Eigenschaften des Linux-Dateisystems in wenigen Sätzen charakterisieren sollte, würden sich unter anderem die folgenden beiden Feststellungen aufdrängen:

1. Alles ist eine Datei (auch »Geräte«).
2. Es gibt ein relativ simples, aber für die meisten Zwecke ausreichendes Rechtesystem.

Daraus folgen diverse sehr nützliche und bequeme Konsequenzen, aber auch Dinge, auf die Sie als sicherheitsbewußter Administrator achten sollten:

Gerätedateien
- Gerätedateien funktionieren irgendwo im Dateisystem, nicht nur unter `/dev`. Wenn es einem Angreifer gelingt, eine für ihn schreibbare Gerätedatei à la `/dev/hda` oder `/dev/kmem` ins System einzuschleppen (sie muss nicht so heißen, sondern nur den richtigen Typ, die richtigen Gerätenummern und die richtigen Zugriffsrechte haben), sind Sie verratzt.

Ausführbare Dateien
- Benutzer können beliebige Dateien ausführbar machen, einfach indem sie das `x`-Bit setzen. Das ist in der Regel kein Problem (es ist sogar ein Vorteil gegenüber anderen Betriebssystemen, die alles ausführen, dessen Name eine bestimmte Endung hat), aber auch kein unüberwindliches Hindernis. Hin und wieder werden Probleme in Linux gefunden, die es einem gewöhnlichen Benutzer ermöglichen, durch die Ausführung eines »normalen« Programms Administratorprivilegien zu erlangen, und dieser Weg steht natürlich auch Programmen offen, die ein Benutzer sich irgendwo aus dem Netz heruntergeladen hat.

Set-UID und Set-GID
- Der Set-UID- und Set-GID-Mechanismus macht es möglich, dass gewöhnliche Benutzer Programme mit den Rechten anderer Benutzer (am liebsten `root`) ausführen. Auch Set-UID- und Set-GID-Programme können irgendwo im Dateisystem liegen.

Als Administrator können Sie einerseits versuchen, das Gefahrenpotential zu minimieren, indem Sie »ungewöhnliche« Dateitypen auf die Bereiche des Systems beschränken, wo Sie sie normalerweise erwarten würden. Andererseits sollten Sie auch proaktiv nach möglicherweise problematischen Dateien suchen. Hier sind ein paar Hinweise:

Wechselmedien
Wechselmedien sind die offensichtliche Möglichkeit für einen Angreifer, un-

[1] Der alte Physiklehrer des Autors dieser Zeilen pflegte zu sagen: »Wer sich auf seinen Lorbeeren ausruht, trägt sie an der falschen Stelle.«

gewöhnliche Dateien ins System einzuschleppen, die er unter seinem normalen Benutzerkonto nicht anlegen kann. Als unprivilegierter Benutzer dürfen Sie ein Kommando wie

```
# mknod /home/hugo/urlaub.jpg c 1 2                    Siehe /dev/kmem
```

nicht ausführen, aber wenn es Ihnen gelingt, einen USB-Stick einzuhängen, der ein ext3-Dateisystem mit einer solchen Datei enthält, die Ihnen gehört, dann haben Sie das große Los gezogen. (So einen USB-Stick können Sie bequem auf Ihrem eigenen Linux-Rechner daheim präparieren.)

Linux adressiert dieses Problem, indem es normalen Benutzern nicht erlaubt, Wechselmedien einzuhängen. Die einzige Methode dafür geht über *mountpoints*, die in `/etc/fstab` explizit mit den Optionen `user` oder `users` versehen wurden. Diese Optionen implizieren aber die Option `nodev`, die dazu führt, dass Gerätedateien auf dem eingehängten Medium nicht beachtet werden. (Pust.) An dieser Stelle könnten wir die Diskussion also beenden; wir möchten das aber nicht tun, ohne Sie daran zu erinnern, dass ein `/etc/fstab`-Eintrag wie

nodev

```
/dev/sda1 /media/usbstick auto noauto,user,dev 0 0
```

sehr gründlich überlegt sein sollte.

Ein ähnliches Problem existiert mit Set-UID-Programmen. Genausowenig wie Gerätedateien dürfen normale Benutzer Programmdateien Set-UID `root` machen (und das ist, um mal Klaus Wowereit zu zitieren, auch gut so). Aber genau wie ein Benutzer eine Gerätedatei auf einem Wechselmedium einschleppen könnte, könnte er auch ein kleines Programm einschleppen, das Set-UID `root` ist und eine Shell startet.

Gängige Shells wie die Bash sträuben sich dagegen, als Set-UID `root` gestartet zu werden. Zumindest ein bisschen. Unseren Angreifer hält das aber nicht auf; er kann auf seinem Wechselmedium auch eine Shell haben, die er sich selbst übersetzt hat, nachdem die entsprechende Prüfung aus dem Quellcode entfernt wurde. (Oder er liest im Bash-Handbuch über die Option `-p` nach.)

Die Strategie, wie Linux mit diesem Problem umgeht, ist sehr ähnlich der gegenüber Gerätedateien: Die `mount`-Optionen `user` und `users` implizieren auch die Option `nosuid`, die die Ausführung von Set-UID-Programmen vom betreffenden Dateisystem aus unterbindet.

nosuid

Genaugenommen unterbindet `nosuid` nicht die Ausführung solcher Programme, es verhindert nur, dass die Set-UID- und Set-GID-Bits beachtet werden.

`nosuid` ist wirkungslos bis gefährlich, falls Sie das Programm `suidperl` installiert haben. Dies ist ein spezieller Interpreter für die Programmiersprache Perl, der für die Ausführung von Set-UID-Perlskripten auf Systemen gedacht ist, die (wie Linux) eigentlich keine Set-UID-Skripte erlauben. `suidperl` ist selber Set-UID `root` und holt für Set-UID-Perlskripte das nach, was bei Set-UID-Maschinenprogrammen der Kernel macht, namentlich die Identität des Dateieigentümers anzunehmen, bevor das Skript gestartet wird. Dumme Idee. Sehr dumme Idee. (Siehe auch [McC08].)

Sie können mit der `mount`-Option `noexec` verhindern, dass von dem betreffenden Dateisystem Programme (oder Skripte) gestartet werden. Auch das wird von `user` und `users` impliziert. Ein großes Hindernis ist es für normale Benutzer aber nicht, da diese ihre Programme immer in ihr Heimatverzeichnis kopieren und von dort starten können. (Früher konnten Sie Programme auf einem `noexec`-Dateisystem auch direkt mit

noexec

```
$ /lib/ld-linux.so.2 /mnt/programm
```

starten – aber seit der Version 2.6 des Linux-Kernels nicht mehr.)

Helfen tut `nosuid` wie auch der `nodev`-Ansatz zunächst aber nur für Wechselmedien. Was auch passieren könnte, ist, dass ein findiger Cracker über irgendeine Sicherheitslücke in einem Netzwerkdienst ins System einbricht und sich eine bequeme Hintertür hinterlassen möchte. Eine Set-UID-Shell unter einem unverfänglichen Namen wie `.foobarrc` ist da gerade recht.

Grundsätzlich hindert Sie niemand daran, die Optionen `nosuid` und `nodev` explizit für *alle* Dateisysteme in Kraft zu setzen, wo Benutzer Dateien anlegen können. Außer für Wechselmedien gilt das vor allem für `/home`, `/tmp` und `/var/tmp`.

Bei aller Prävention machen Sie als Systemadministrator trotzdem keinen Fehler, wenn Sie Ihre Platten in gewissen Abständen nach verdächtigen Dateien abklopfen (am besten irgendwann nachts, wenn nicht so viel Betrieb ist). Das naheliegende Werkzeug dafür ist `find`.

Set-UID-Dateien finden

Um alle Set-UID- oder Set-GID-Dateien im System zu finden, können Sie die Option `-perm` von `find` verwenden. Ein entsprechendes Kommando sieht ungefähr so aus:

```
# find / -perm /06000 -type f -print
```

(Denken Sie daran, dass `find` verschiedene Kriterien implizit UND-verknüpft.)

»`-perm /06000`« können Sie bei GNU `find` sagen, das bei Linux Standard ist; andere Implementierungen von `find` verlangen das umständlichere

```
\( -perm -04000 -o -perm -02000 \)
```

Ohne das »`-type f`« bekommen Sie auch Verzeichnisse mit gesetztem Set-GID-Bit angezeigt. Ein gesetztes Set-GID-Bit auf einem Verzeichnis hat jedoch keine sicherheitsrelevanten Konsequenzen.

Möglicherweise können Sie sich Arbeit sparen, indem Sie nicht nur den Namen einer gefundenen Datei ausgeben, sondern gleich die kompletten Dateiinformationen à la »`ls -l`«. Verwenden Sie dafür die Aktion `-ls` statt `-print`.

Sie sollten einen Überblick über die Set-UID- und Set-GID-Programme im Sys-

Liste

tem haben – legen Sie am besten eine Liste an, die Sie manuell prüfen und später (etwa per `cron`) automatisch mit der Wirklichkeit vergleichen. Lassen Sie sich benachrichtigen, wenn ein Programm im System fehlt, das in der Liste steht oder – wichtiger – ein Programm dazugekommen ist, das vorher nicht in der Liste stand. Manche Distributionen haben so etwas schon von sich aus.

Gerätedateien finden

Gerätedateien können Sie auf ähnliche Weise aufspüren: Fragen Sie nach, ob eine Datei einem block- oder zeichenorientierte Gerät entspricht:

```
# find / \( -type b -o -type c \) -ls
```

Das Verzeichnis `/dev` müssen Sie eigentlich nicht durchsuchen, allerdings gibt es keine direkte Methode, es gezielt von der Suche auszunehmen. Allenfalls können Sie `find` sagen, dass es seine Bemühungen einstellen soll, wenn es auf das Verzeichnis `/dev` gestoßen ist:

```
# find / -path /dev -prune -o \( -type b -o -type c \) -ls
```

Oder Sie können natürlich die Ausgabe von `find` durch "`grep -v ^/dev`" leiten, um die `/dev`-Einträge aus dem Ergebnis zu entfernen.

Sie können das Auditing Ihrer Dateisysteme auch noch wesentlich exakter betreiben: Programme wie Tripwire oder AIDE dienen dazu, Veränderungen an wichtigen Systemdateien (Programmdateien oder Konfiguration) zu erkennen und zu melden, wie sie zum Beispiel vorkommen können, wenn sich ein »Rootkit« im System einnistet. Dies im Detail auszuführen ist an dieser Stelle nicht möglich; das Thema wird in der Linup-Front-Schulungsunterlage *Linux-Sicherheit* wieder aufgegriffen.

Tripwire
AIDE

Übungen

13.1 [!1] Warum ist es ein Problem, wenn ein normaler Benutzer direkten Zugriff auf `/dev/hda` (oder dieselbe Gerätedatei unter einem anderen Namen) hat?

13.2 [!2] Vergewissern Sie sich, dass die `mount`-Optionen `nodev`, `nosuid` und `noexec` das tun, was sie sollen. Sie können das zum Beispiel auf `/tmp` testen, falls `/tmp` bei Ihnen ein eigenes Dateisystem ist; am bequemsten setzen Sie die Optionen über etwas wie

```
# mount -o remount,nodev /tmp
```

in Kraft. (Ist `/tmp` bei Ihnen kein eigenes Dateisystem, können Sie mit etwas wie

```
# dd if=/dev/zero of=/tmp/testfs bs=1M count=32
# mke2fs -F /tmp/testfs
# mount -o loop,nodev /tmp/testfs /mnt
```

ein Dateisystem in einer Datei zum Testen anlegen.)

13.3 [2] Stellen Sie eine komplette Liste aller Set-UID- und Set-GID-Programme auf Ihrem System zusammen. (*Für Bonuspunkte:* Überlegen Sie sich, warum das jeweilige Programm Set-UID bzw. Set-GID sein muss.)

13.4 [2] Prüfen Sie, ob die oben angegebene Methode zum Finden von Gerätedateien außerhalb von `/dev` funktioniert. Legen Sie dazu irgendwo im System ein paar Gerätedateien an (und entfernen sie hinterher wieder).

13.3 Benutzer und Dateien

Manchmal ist es nützlich, herauszufinden, wer gerade auf einem Rechner angemeldet ist. Das einfachste Kommando dafür heißt `who`:

who

```
$ who
hugo     :0           2015-07-27 13:39 (:0)
hugo     pts/0        2015-07-27 13:56 (red.example.com)
```

Die erste Spalte nennt den Benutzernamen und die zweite das Terminal, auf dem der Benutzer angemeldet ist. »`:0`« steht dabei für den X11-Server mit diesem Displaynamen; andere Namen (etwa `pts/0`) beziehen sich auf eine Gerätedatei unter `/dev`.

Terminalnamen der Form `pts/`… bezeichnen »Pseudo-Terminals«, typischerweise Terminalfenster in einer grafischen Umgebung oder Sitzungen mit der Secure Shell.

Der Rest der Zeile gibt Datum und Uhrzeit der Anmeldung an. Anschließend folgt möglicherweise noch in Klammern die Gegenstelle der Sitzung; der auf `pts/0` angemeldete Benutzer `hugo` zum Beispiel kommt vom Rechner `red.example.com`.

Wenn Sie mehrere Terminalfenster oder ein Terminalfenster mit mehreren Unterfenstern haben, dann ist es höchstwahrscheinlich, dass alle diese Sitzungen als separate »Benutzer« gezählt werden. Lassen Sie sich davon nicht verwirren.

`who` unterstützt ein paar Kommandozeilenoptionen von weitergehendem Interesse: `-H` sorgt dafür, dass eine Kopfzeile mit Spaltentiteln angegeben wird; `-b` gibt den Zeitpunkt des letzten Systemstarts aus und `-r` den aktuellen Runlevel (falls angebracht). `-a` liefert alles, was `who` ausgeben kann. Mit `-m` wird nur derjenige Benutzer angezeigt, dessen Terminal mit der Standardeingabe von `who` verbunden ist.

Wenn Sie `who` mit zwei Parametern aufrufen, die keine Optionen sind, dann wird das als »`who -m`« interpretiert. Der tatsächliche Parametertext ist bedeutungslos. Damit funktioniert das klassische

```
$ who am i
hugo     pts/0        2015-07-27 13:56 (red.example.com)
```

aber auch alles Mögliche andere:

```
$ who is cool
$ who wants icecream
```

Nicht zu verwechseln ist das mit dem Kommando `whoami` (ohne Leerzeichen), das den effektiven Benutzernamen ausgibt:

```
$ whoami
hugo
```

Kommando `w`

Das Kommando `w` ist ein naher Verwandter von `who`:

```
# w
 18:06:32 up  4:29,  2 users,  load average: 0,01, 0,02, 0,05
USER  TTY   FROM            LOGIN@ IDLE   JCPU  PCPU WHAT
hugo  :0    :0              13:39 ?xdm?  40.21s 0.06s /usr/bin▷
 ◁/lxsession -s
hugo  pts/0 red.example.com 13:56  0.00s  0.11s 0.00s w
```

Die erste Zeile liefert die aktuelle Uhrzeit und die *uptime*, also die Zeit, die der Rechner bisher in Betrieb ist, die Anzahl der aktiven Benutzer und die Systemlast. Anschließend kommt eine Zeile für jeden angemeldeten Benutzer, so wie `who` das auch machen würde. Dabei gibt `LOGIN@` an, wann der Benutzer sich angemeldet hat (als Uhrzeit oder – wenn das Anmelden länger zurückliegt – als Datumsangabe). `IDLE` ist die Zeitspanne, seit der der Benutzer nichts getan hat. `JCPU` ist die bisher von allen Prozessen in dieser Sitzung insgesamgt verbrauchte Rechenzeit (exklusive abgeschlossener Hintergrundjobs, aber mit Hintergrundjobs, die gerade laufen) und `PCPU` ist die bisher vom aktuell laufenden Prozess verbrauchte Rechenzeit. `WHAT` ist die aktuelle Kommandozeile.

Die Optionen von `w` sind nicht so interessant: `-h` unterdrückt die Kopfzeile und `-s` die Spalte `LOGIN`. Die anderen Optionen sind noch langweiliger.

Wenn Sie einen Benutzernamen angeben, werden nur Informationen über diesen Benutzer ausgegeben.

`last`

Während `who` und `w` sich um die Gegenwart kümmern, können Sie mit `last` die Vergangenheit betrachten und schauen, wer wann angemeldet war:

Tabelle 13.1: Zugriffscodes für Prozesse bei `fuser`

Code	Bedeutung
`c`	Als aktuelles Verzeichnis benutzt
`e`	Als Programmdatei ausgeführt
`f`	Als Datei geöffnet (zum Lesen)*
`F`	Als Datei geöffnet (zum Schreiben)*
`r`	Als Wurzelverzeichnis benutzt
`m`	Als *shared library* benutzt oder in den Speicher eingeblendet

* = Die Codes `f` und `F` werden in der normalen Ausgabe nicht angezeigt.

```
# last
hugo   pts/0       red.example.com Mon Jul 27 13:56   still logged in
hugo   :0          :0              Mon Jul 27 13:39   still logged in
reboot system boot 3.16.0-4-amd64  Mon Jul 27 13:37 - 18:20  (04:43)
hugo   :0          :0              Sun Jul 26 17:43 - crash  (19:53)
reboot system boot 3.16.0-4-amd64  Sun Jul 26 17:43 - 18:20 (1+00:37)
hugo   pts/2       red.example.com Fri Jul 24 20:00 - crash (1+21:42)
hugo   :0          :0              Fri Jul 24 19:15 - crash (1+22:27)
hugo   :0          :0              Fri Jul 10 14:19 - 19:15 (14+04:55)
hugo   :0          :0              Fri Jul 10 13:22 - 14:19  (00:56)
reboot system boot 3.16.0-4-amd64  Fri Jul 10 12:46 - 18:20 (17+05:33)
```

Auch hier haben Sie für jeden Anmeldevorgang den Benutzer, das Terminal (oder `:0` für die grafische Umgebung), die Gegenstelle, das Anmeldedatum, die Abmeldezeit und die Sitzungsdauer. Steht bei der Abmeldezeit `crash`, dann hatte der Rechner keine Zeit mehr, das Sitzungsende zu vermerken (kann bei virtuellen Maschinen mal vorkommen). Die `reboot`-Zeilen verweisen auf Systemneustarts, wobei dann in der dritten Spalte (wo sonst die Gegenstelle steht) der Name des gestarteten Systemkerns auftaucht.

`last` betrachtet die Datei `/var/log/wtmp` (jedenfalls solange Sie nicht mit der Option `-f` eine andere Datei benennen). Wenn Sie einen oder mehrere Benutzer- oder Terminalnamen angeben, dann bekommen Sie nur die Aktivitäten der betreffenden Benutzer (oder Terminals) gezeigt

Die Informationen, die `last` ausgibt, könnten Sie mit einer gewissen Berechtigung als sensibel ansehen. Immerhin erlauben sie Ihnen eine rückwirkende Kontrolle, wer wann wo angemeldet war – was für die Fehlersuche mitunter nützlich ist, aber auch einen betrieblichen Datenschutzbeauftragten oder die Personalvertretung gegen Sie aufbringen könnte. Seien Sie daher umsichtig und heben Sie die betreffenden Daten nicht *zu* lange auf, ohne mit jemandem Verantwortlichen darüber diskutiert zu haben.

Mitunter möchten Sie herausfinden, welcher Prozess oder Benutzer gerade auf irgendeine Ressource im System zugreift (das typische Beispiel ist, dass Sie ein Dateisystem aushängen wollen, aber das System Sie nicht läßt, weil irgendein Prozess noch eine Datei oder ein Verzeichnis auf dem Dateisystem benutzt). Dabei hilft Ihnen das Kommando `fuser`[2] – es übernimmt als Parameter den Namen einer Datei oder eines Dateisystems (oder mehrerer Dateien/Dateisysteme) und listet alle Prozesse auf, die diese Objekte »benutzen«:

`fuser`

```
$ cd /home/hugo
$ (echo Hallo; sleep 600) >test.txt
# In einem anderen Fenster:
```

[2] Wir lesen das als »eff-juser«, nicht »fjuser« oder gar »fuser«.

```
$ fuser /home/hugo /home/hugo/test.txt
/home/hugo:            767c  1360c  1465c  1467c  1488c  1489c
/home/hugo/test.txt:  1488  1489
```

Die Ausgabe von `fuser` ist im Standardfall eine Reihe von PIDs, gefolgt von einem Buchstaben, der angibt, wie der betreffende Prozess auf die Ressource zugreift. Die möglichen Buchstaben stehen in Tabelle 13.1.

Mit der Option `-v` bekommen Sie eine ausführlichere Ausgabe:

```
# fuser -v /home/hugo/test.txt
                      BEN.        PID ZUGR.  BEFEHL
  /home/hugo/test.txt: hugo       1488 F.... bash
                       hugo       1489 F.... sleep
```

(Unter `ZUGR.` sehen Sie wieder die Zugriffscodes aus Tabelle 13.1.)

Die Option `-m` erweitert den Suchradius von »die genannte Datei oder das genannte Verzeichnis« auf »irgendeine Datei oder ein Verzeichnis auf demselben Dateisystem wie die genannte Datei oder das genannte Verzeichnis«. Das Kommando

```
# fuser -m /srv
```

zum Beispiel listet alle Prozesse auf, die auf irgendeine Datei (oder ein Verzeichnis) auf dem Dateisystem zugreifen, wo `/srv` liegt. Das ist natürlich extrem nützlich, wenn `/srv` bei Ihnen ein eigenes Dateisystem ist und Sie es gerne aushängen möchten.

Um den Fall »Ich möchte ein Dateisystem aushängen und Linux läßt mich nicht« noch effizienter loswerden, können Sie mit der Option `-k` gleich ein Signal (`SIGKILL`, wenn Sie nichts Anderes sagen) an die betreffenden Prozesse schicken:

```
# fuser -mk -TERM /srv; sleep 10; fuser -mk /srv
```

läßt den Prozessen 10 Sekunden Zeit, um ihre Angelegenheiten zu ordnen, und dann geht es ihnen (oder denen, die danach noch übrig sind) an den Kragen. Wenn Sie außerdem die Option `-w` angeben, werden nur solche Prozesse beendet, die schreibend auf die Ressource(n) zugreifen. Mit `-i` werden Sie bei jedem Prozess gefragt, ob Sie ihn wirklich beenden möchten.

`fuser` kann nicht nur die Benutzer von Dateien identifizieren, sondern auch die von TCP- und UDP-Ports:

```
# fuser -n tcp ssh
ssh/tcp:     428   912   914
```

Mit der Option `-n` wird der »Namensraum« gewählt (`file` – der Standard –, `tcp` oder `udp`), und danach können Sie dann Portnummern oder die symbolischen Namen von Ports (siehe `/etc/services`) angeben.

Solange keine Verwechslungsgefahr gegeben ist, können Sie Portnummern auch in der Form `ssh/tcp` angeben und die `-n`-Option weglassen.

`fuser` für TCP- oder UDP-Ports funktioniert am besten als `root`. Normale Benutzer haben auf die entsprechenden Datenstrukturen in der Regel keinen Zugriff.

Übungen

13.5 [!2] Probieren Sie die Kommandos `who` und `w` aus. Überzeugen Sie sich, dass Terminalsitzungen in der Grafikoberfläche als separate Benutzersitzungen gezählt werden.

13.6 [1] Benutzen Sie `who`, um den Zeitpunkt des letzten Systemstarts anzuzeigen.

13.7 [3] (Für Ihr `sed`- und `awk`-Fu.) Schreiben Sie ein Skript, das berechnet, wie lange ein Benutzer *insgesamt* angemeldet war (innerhalb des Horizonts von `last`).

13.4 Ressourcenlimits

Wenn mehrere Benutzer sich denselben Rechner teilen (sei es über Terminals oder Anmelden über das Netz mit der `ssh`), ist es wichtig, dass keiner von ihnen die Möglichkeit hat, die Ressourcen des Rechners zu Ungunsten der anderen zu monopolisieren. Selbst wenn Sie einen Rechner ganz für sich alleine haben, kann es nützlich sein, einzelne Prozesse in ihre Schranken zu weisen, wenn sie sich allzu gierig gebärden. Um das zu erleichtern, sieht Linux den Mechanismus der »Ressourcenlimits« (engl. *resource limits*) vor.

Über Ressourcenlimits können Sie Obergrenzen für verschiedene Ressourcen vorgeben, die Benutzer (oder einzelne Prozesse dieser Benutzer) ausnutzen dürfen. Mit dem Shell-Kommando `ulimit` können Sie sich einen Überblick darüber verschaffen, welche Ressourcen das sind – die Option `-a` zeigt alle Ressourcenlimits und ihre aktuellen Werte an:

```
$ ulimit -a
core file size          (blocks, -c) 0                    Größe von Core-Dumps
data seg size           (kbytes, -d) unlimited            Dynamischer Speicher
scheduling priority             (-e) 0                    Nice-Wert, in etwa
file size               (blocks, -f) unlimited            Dateigröße
pending signals                 (-i) 8181                 Ausstehende Signale
max locked memory       (kbytes, -l) 32                   Gesperrter Speicher
max memory size         (kbytes, -m) unlimited            RAM-Verbrauch, unbeachtet
open files                      (-n) 1024                 Offene Dateien
pipe size            (512 bytes, -p) 8                    Puffer für Pipes
POSIX message queues     (bytes, -q) 819200               Warteschlangengröße
real-time priority              (-r) 0                    Maximale Echtzeit-Priorität
stack size              (kbytes, -s) 8192                 Größe des Stapels
cpu time               (seconds, -t) unlimited            Verbrauchte Rechenzeit
max user processes              (-u) 8181                 Prozesse (pro Benutzer)
virtual memory          (kbytes, -v) unlimited            Adressraum
file locks                      (-x) unlimited            Dateisperren
```

Für jedes dieser Ressourcenlimits gibt es eine »weiche« und eine »harte« Grenze. Der Unterschied ist, dass Sie Administratorprivilegien brauchen, um die harte Grenze zu erhöhen (verringern dürfen Sie sie immer). Die weiche Grenze können Benutzer beliebig wählen, allerdings nur bis zum aktuellen Wert der harten Grenze als Maximum. In der Shell setzen Sie harte Grenzen mit »`ulimit -H`« und weiche Grenzen mit »`ulimit -S`«; wenn Sie weder das eine noch das andere angeben, werden die harte und die weiche Grenze auf denselben Wert gesetzt.

weiche Grenze
harte Grenze

Wenn Sie keinen neuen Wert für ein Ressourcenlimit angeben, wird nur der aktuelle Wert angezeigt:

```
$ ulimit -n 512                                   512 offene Dateien pro Prozess
$ ulimit -n                                       Wie viele waren es noch gleich?
512
```

Ressourcenlimits sind Prozessattribute

Die Ressourcenlimits sind Prozessattribute (ähnlich wie das aktuelle Verzeichnis oder die Prozessumgebung) und vererben sich darum von einem Prozess auf seine Kindprozesse.

Zu einigen der Ressourcenlimits müssen wir noch ein paar Anmerkungen machen:

core file size Dies ist vor allem interessant für Programmentwickler – Core-Dumps entstehen, wenn ein Programm unvorhergesehenerweise durch ein Signal zum Absturz gebracht wird, und erlauben eine Untersuchung der Ursachen. Beim Standardwert 0 werden gar keine Core-Dumps geschrieben; wenn Sie welche brauchen, sollten Sie etwas wie

```
$ ulimit -c unlimited
```

sagen.

data seg size Wenn die weiche Grenze dieses Limits erreicht ist, bekommt der Prozess keinen Speicher mehr vom Betriebssystem. Er kann sich wahrscheinlich über die Runden retten, wenn er schon benutzten Speicher wieder freigibt.

scheduling priority Gibt eine Untergrenze (!) für den Nice-Wert eines Prozesses an. Da `ulimit` keine negativen Parameter zuläßt, ist der tatsächliche Wert, der angenommen wird, wenn Sie »`ulimit -e` *n*« sagen, 20 – *n*. Das Ganze ist etwas verdreht, aber läuft darauf hinaus, dass Sie es normalen Benutzern erlauben können, den Nice-Wert eines Prozesses unter 0 zu senken:

```
# ulimit -e 30                                    Effektiv –10
# /bin/su - hugo
$ nice --5 /bin/sleep 10                          Funktioniert; Nice-Wert –5
$ nice --15 /bin/sleep 10
nice: cannot set niceness: Permission denied
```

file size Wenn ein Prozess versucht, eine Datei größer zu machen, als dieses Limit angibt, bekommt er das Signal `SIGXFSZ`. Normalerweise bricht das den Prozess ab, aber wenn der Prozess es abfängt, führt nur der Schreibvorgang zu einer Fehlermeldung, und der Prozess kann sich etwas Anderes überlegen.

max memory size Dieses Ressourcenlimit klingt sehr verführerisch, wird aber seit Linux 2.4.30 nicht mehr beachtet und tat auch da nicht das, was Sie vielleicht meinen.

open files Anzahl der Dateien, die der Prozess gleichzeitig offen haben kann. Wird das Limit erreicht und der Prozess versucht noch eine Datei zu öffnen, schlägt der Versuch fehl.

stack size Gibt die Größe des Prozess-Stapels (oder, falls Sie ein Diplom in Informatik haben, des »Kellers«) an, in Kibibytes.

Außerdem, seit Linux 2.6.23, die Größe des Platzes für Kommandozeilenargumente und Umgebungsvariable: Linux stellt dem Prozess 1/4 dieser Größe zur Verfügung, aber minimal 32 Speicherseiten, was auf Architekturen wie dem x86-PC, wo Linux Speicherseiten von 4 KiB verwendet, auf 128 KiB (den Wert vor Linux 2.6.23) hinausläuft. Bei dem Standardwert für das Limit ist der übliche Wert jetzt aber 2 MiB.

cpu time Wenn ein Prozess an die weiche Grenze dieses Limits kommt, bekommt er das Signal `SIGXCPU` geschickt. Dies wiederholt sich einmal pro Sekunde, bis die harte Grenze erreicht ist; dann setzt es `SIGKILL`.

max user processes Dieses Limit unterscheidet sich von den anderen darin, dass es pro Benutzer gilt und nicht pro Prozess oder Datei. Es gibt die maximale Anzahl von parallelen Prozessen an, die der Benutzer gleichzeitig haben darf. Wird es erreicht, schlagen Versuche, mehr Prozesse anzulegen, fehl.

Sie können Ressourcenlimits immer interaktiv mit `ulimit` setzen. Sie gelten dann für die Shell, in der das Kommando ausgeführt wurde, und vererben sich an die nächsten Kommandos (und deren Kindprozesse). ulimit

Ein Klassiker des Genres ist die »Fork-Bombe« à la Fork-Bombe

```
:(){ :|:& };:
```

oder (etwas leserlicher hingeschrieben)

```
f() {
  f | f &
}
f
```

Diese kleine Shell-Kommandozeile startet einen Prozess, der zwei Kindprozesse erzeugt und sich dann in den Hintergrund schickt. Jeder dieser Kindprozesse erzeugt wieder zwei Kindprozesse und so weiter … Sie erkennen, was hier gespielt wird. Das Problem ist, dass diese Sorte Programm, wenn Sie sie unkontrolliert laufen lassen, die komplette Prozesstabelle des Kernels monopolisiert und Sie als Systemadministrator keine neuen Prozesse starten können, um der Lage Herr zu werden. (Immer wenn sich irgendein Prozess beendet, ist die Fork-Bombe vermutlich eher da als Sie, um die freigewordene Position in der Prozesstabelle zu usurpieren.) – Sie können das Problem aber gar nicht erst aufkommen lassen, indem Sie Benutzer nicht genug Prozesse erzeugen lassen, um die Prozesstabelle zu füllen. Etwas wie

```
ulimit -u 128
```

(gegenüber dem Standardwert von 8191) sollte solchen Machenschaften einen Riegel vorschieben, ohne den Stil Ihrer Benutzer allzusehr einzuschränken. (Die Größe der Prozesstabelle können Sie übrigens der Datei `/proc/sys/kernel/threads-max` entnehmen.)

Sollte Ihr System mal von einer Fork-Bombe befallen sein und Sie haben eine `root`-Shell zur Hand, können Sie versuchen, die Prozesse zu beenden. Sie müssen dabei ein bisschen aufpassen, da, wenn Sie einfach nur `kill` sagen, jeder beendete Prozess ja gleich wieder durch einen neuen ersetzt wird. Versuchen Sie statt dessen, zuerst alle Fork-Bomben-Prozesse anzuhalten (nicht zu beenden). Damit können sie sich nicht mehr vermehren. Anschließend werden sie dann beendet:

```
# killall -STOP prozessname
# killall -KILL prozessname
```

(Versuchen Sie das nicht auf einem Rechner mit Solaris oder BSD.)

Wenn Ihre Windows benutzenden Kumpels sich darüber lustig machen, wie leicht ein Linux-Rechner mit einer einzigen Kommandozeile zum Stehen gebracht werden kann: Eine Batch-Datei mit dem Inhalt Windows

```
%0|%0
```

funktioniert unter Windows gerade genauso gut – und man kann noch weniger dagegen unternehmen. Fünf Zeichen versus dreizehn.

Ressourcenlimits für Benutzer

Um Benutzern tatsächlich restriktive Ressourcenlimits aufzubrummen, müssen Sie diese setzen, bevor die Benutzer ans Ruder kommen und auch so, dass sie die entsprechenden `ulimits` nicht selber ändern können (die Datei `~/.bash_profile` scheidet also aus). Ein naheliegender Platz wäre `/etc/profile`, jedenfalls solange alle Ihre Benutzer die Bash als Login-Shell verwenden.

pam_limits

Die elegantere Methode besteht darin, das PAM-Modul `pam_limits` zu verwenden. Dieses Modul wird in den Anmeldeprozess integriert und gestattet die Definition differenzierter Ressourcenlimits für einzelne Benutzer und Gruppen in der Datei `/etc/security/limits.conf`. Die Einzelheiten finden sich in den Handbuchseiten `pam_limits`(8) und `limits.conf`(5).

Übungen

13.8 [!2] Suchen Sie sich ein paar interessante Ressourcenlimits aus (`file size` und `open files` liegen nahe) und schreiben Sie Shellskripte, die prüfen, ob diese Ressourcenlimits tatsächlich greifen. (Sie können die Ressourcenlimits am Anfang eines Skripts auf einen lächerlich niedrigen Wert setzen, um die Effekte früher zu sehen.)

13.9 [!2] Bauen Sie in `/etc/profile` ein paar interessante Ressourcenlimits ein und vergewissern Sie sich, dass diese Limits für neu angemeldete Benutzer greifen. Wie können Sie dafür sorgen, dass Ihre Ressourcenlimits nur für bestimmte Benutzer gelten?

13.10 [2] Verwenden Sie die Datei `/etc/security/limits.conf`, um für ein Benutzerkonto ein paar interessante Ressourcenlimits zu setzen, und vergewissern Sie sich, dass diese Limits greifen. Melden Sie sich dazu auf einer Textkonsole als der betreffende Benutzer an. (Damit das funktioniert, muss `pam_limits` bei Ihnen konfiguriert sein. Prüfen Sie, ob in der Datei `/etc/pam.d/login` eine Zeile vorkommt, die ungefähr so aussieht:

```
session required pam_limits.so
```

In diesem Fall steht `pam_limits` bei Ihnen zur Verfügung.)

13.11 [2] Experimentieren Sie mit der Fork-Bombe, indem Sie eine »entschärfte« Version starten, die ungefähr so aussieht:

```
#!/bin/sh
# forkbomb.sh
f() {
  sleep 5
  f | f &
}
f
```

Verfolgen Sie die Anzahl der Fork-Bomben-Prozesse mit einem Kommando wie

```
# watch 'ps auxw | grep forkbomb.sh | grep -v "\(watch\|egrep\)«
```

Beenden Sie die Fork-Bombe wie oben erklärt mit `killall`.

13.12 [2] (Für Wagemutige.) Lassen Sie die Fork-Bombe *ohne* das sleep-Kommando laufen. Um den ansonsten wahrscheinlich nötigen Systemneustart zu vermeiden, können Sie *vorher* ein Kommando wie

```
# (sleep 15 && exec /usr/bin/killall -STOP forkbomb.sh) &
```

absetzen. (Oder versuchen Sie das Ganze in einer virtuellen Maschine.)

13.5 Administratorprivilegien mit sudo

Mit dem Kommando su können normale Benutzer die Identität von root annehmen (wenn sie das richtige Kennwort wissen). Zwar protokolliert das System die Anwendung von su[3], aber anschließend hat der betreffende Benutzer komplette Administratorprivilegien – wenn es sich nur um den Praktikanten handelt, der das wöchentliche Backup anstoßen soll, vielleicht etwas zuviel des Guten. Es wäre schöner, wenn Sie einzelnen Benutzern gezielt erlauben könnten, *bestimmte* Kommandos als root auszuführen.

Genau dafür ist sudo gedacht. Der Sinn dahinter ist, dass das root-Benutzerkonto eigentlich überhaupt nicht mehr direkt verwendet wird. Statt dessen können entsprechend privilegierte Benutzer mit etwas wie

```
$ sudo passwd susi
```

Kommandos als root ausführen. sudo fragt dabei das Kennwort des *Benutzers* (nicht das von root) ab – wobei eine richtige Antwort für eine gewisse Zeit vorhält – oder kann auf eine Kennwortabfrage ganz verzichten.

Gewisse Distributionen – Ubuntu, Debian GNU/Linux auf Wunsch, und übrigens auch MacOS X – privilegieren den ersten bei der Installation angelegten Benutzer automatisch dafür, sudo benutzen zu dürfen, und erlauben direktes Anmelden als root überhaupt nicht mehr. Das hilft immerhin dagegen, dass naive Benutzer immer als root arbeiten, »weil man so ja alles darf«.

Tatsächlich können Sie sich bei sudo in der Konfiguration aussuchen, ob ein Benutzer sein eigenes Kennwort eingeben soll oder das von root. Letzteres ist im Großen und Ganzen aber eine dumme Idee. (Was die Entwickler der Novell/SUSE-Distributionen anscheinend nicht davon abgehalten hat, dies zur Voreinstellung zu machen.) Siehe Übung 13.14.

Normalerweise gilt eine erfolgreiche Kennworteingabe 15 Minuten lang (ist konfigurierbar). Mit dem Kommando »sudo -k« (das keine Kennworteingabe erfordert) können Sie die Kennworteingabe aber zurücksetzen, so dass sudo beim nächsten Kommando wieder nachfragt.

Das komplette Verzichten auf die Kennwortabfrage ist ein zweischneidiges Schwert. Es ist schön bequem, aber macht die Konten der Benutzer, die das dürfen, effektiv zu root-Konten, die entsprechend abgesichert werden müssen.

Alle per sudo abgeschickten Kommandos werden mit dem Syslog-Mechanismus (Kapitel 1) protokolliert. Das sieht dann ungefähr so aus: Protokoll

```
Feb 05 22:13:15 red sudo:  hugo : TTY=pts/5 ; PWD=/home/hugo ;▷
 ◁ USER=root ; COMMAND=/usr/bin/passwd
```

/etc/sudoers

Die Datei /etc/sudoers regelt, wer sudo benutzen darf und wofür. Das Format der Datei ist relativ komplex und die Möglichkeiten spotten jeder Beschreibung. Wir erklären hier nur den allereinfachsten Gebrauch.

visudo

Zum Editieren von /etc/sudoers sollten Sie das Kommando visudo verwenden. Es ruft die Datei mit dem Editor Ihrer Wahl auf (es schaut erst in der Umgebungsvariable VISUAL, dann in der Umgebungsvariable EDITOR und schließlich in der editor-Einstellung in /etc/sudoers, bevor es auf den guten alten vi zurückfällt) und macht, nachdem Sie Ihre Änderungen abgespeichert haben, gleich eine Syntaxüberprüfung, bevor es Ihre neue Konfiguration in Kraft setzt. Dies ist sehr wichtig, damit Sie im Fehlerfall keine Konfigurationsdatei bekommen, an der sudo später verzweifelt, so dass Sie sich den Ast abgesägt haben, auf dem Sie sitzen. Außerdem stellt es sicher, dass immer nur ein Benutzer die Datei ändern darf.

Regeln

Herzstück von /etc/sudoers sind Zeilen mit Regeln in der Form

```
hugo  ALL = /usr/bin/cancel [a-z]*
```

Diese Regel erlaubt es dem Benutzer hugo, das Kommando cancel in der Form

```
$ sudo cancel lp-123
```

Parameter

aufzurufen. Das »[a-z]*« verlangt, dass der Parameter von cancel mit einem Buchstaben beginnt – Optionen können also nicht eingeschmuggelt werden.

ALL

Das »ALL« heißt, dass die Regel auf allen Rechnern gilt. Das macht es in einem Netz einfacher, sudo in einer einzigen Datei für alle Rechner zu konfigurieren: Wenn hugo nur auf dem Rechner red Druckaufträge löschen darf, dann schreiben Sie

```
hugo  red = /usr/bin/cancel [a-z]*
```

Auf dem Rechner blue zum Beispiel wird der Befehl dann abgewiesen, auch wenn er mit sudo aufgerufen wird.

Dass eine sudoers-Datei potentiell Regeln für alle Rechner im Netz enthalten kann, heißt noch lange nicht, dass Sie die Datei nur auf einem einzigen Rechner vorhalten müssen. Weit gefehlt: Sie als Administrator sind dafür verantwortlich, die Datei auf jedem Rechner verfügbar zu machen, wo sie gelten soll.

beliebige Parameter

Wenn Sie einfach nur einen Kommandonamen angeben (ohne Parameter), dann sind beliebige Parameter erlaubt:

```
hugo ALL = /usr/bin/cancel
```

keine Parameter

Wenn gar keine Parameter erlaubt sein sollen, dann müssen Sie als einzigen Parameter in der sudoers-Datei eine leere Zeichenkette angeben: Mit

```
hugo ALL = /usr/bin/passwd ""
```

dürfte hugo *nur* das root-Kennwort ändern. (O. K., O. K. …)

Seien Sie vorsichtig mit Kommandos wie vi, die es dem Benutzer erlauben, Shellkommandos zu starten. Diese Kommandos werden dann auch mit root-Rechten ausgeführt! Um das zu vermeiden, können Sie etwas sagen wie

[3] … was Ihnen natürlich nur hilft, wenn das Protokoll auf einem anderen Rechner geführt wird, weil der Benutzer nach dem erfolgreichen su ein lokales Protokoll ändern oder löschen könnte.

```
hugo ALL = NOEXEC: /usr/bin/vi
```

Dadurch wird vi daran gehindert, andere Programme aufzurufen. (Am Ende dieses Abschnitts wird noch eine bessere Strategie für den Umgang mit Editoren erklärt.)

Sie können auch mehrere Kommandos angeben: mehrere Kommandos

```
hugo ALL = /usr/bin/cancel [a-z]*, /usr/bin/cupsenable [a-z]*
```

erlaubt »sudo cancel« und »sudo cupsenable«, jeweils mit (mindestens) einem Parameter.

Wenn Sie einen Verzeichnisnamen (als absoluten Pfadnamen mit Schrägstrich am Schluss) angeben, steht das für alle ausführbaren Dateien in diesem Verzeichnis (aber nicht in Unterverzeichnissen):

```
hugo ALL = /usr/bin/                                    Alles in /usr/bin
```

Oft ist es bequemer, Kommandos zu Gruppen zusammenzufassen. Hierfür verwenden Sie Kommando-Aliasnamen: Kommando-Aliasnamen

```
Cmnd_Alias PRINTING = /usr/bin/cancel [a-z]*, \
                      /usr/bin/cupsenable [a-z]* \
                      /usr/bin/cupsdisable [a-z]*
◁◁◁◁◁
hugo ALL = PRINTING
```

Sie können Kommando-Aliasnamen und direkt angegebene Kommandos auch mischen:

```
hugo ALL = PRINTING, /usr/bin/accept [a-z]*
```

Der Kommando-Aliasname »ALL« steht für »alle Kommandos«:

```
hugo ALL = ALL
```

gibt hugo unter dem Strich volle Administratorprivilegien.

Mit einem davorgesetzten »!« können Sie Kommandos ausschließen, die sonst von der Konfiguration erlaubt würden: Kommandos ausschließen

```
hugo ALL = /usr/bin/passwd [a-z]*, !/usr/bin/passwd root
```

erlaubt es hugo, alle Kennwörter zu ändern bis auf das von root.

Sie sollten mit dem »!« nicht zu trickreich werden. Halten Sie insbesondere Abstand von Konstruktionen wie

```
hugo ALL = /bin/, !/bin/sh, !/bin/bash
```

– wenn hugo alle Kommandos in /bin als root ausführen darf außer »sh«, dann hält ihn auch keiner davon ab, etwas wie

```
$ sudo cp /bin/sh /bin/mysh
$ sudo mysh
```

zu sagen. Um solche Machenschaften zu erkennen, ist sudo nicht zynisch genug. Achten Sie auch auf Kombinationen mit ALL.

Endgültig alle Schranken fallen läßt eine Regel wie

```
hugo ALL = NOPASSWD: ALL
```

die es hugo erlaubt, alle Kommandos als root auszuführen und dabei nicht einmal nach einem Kennwort gefragt zu werden.

Andere Benutzer

Normalerweise führt sudo Kommandos als root aus. Sie können aber auch einen anderen Benutzer angeben: Mit

```
hugo ALL = (mysql) /usr/bin/mysqladmin
```

kann hugo das Kommando mysqladmin als Benutzer mysql ausführen. Aufrufen muss er das wie

```
$ sudo -u mysql mysqladmin flush-privileges
```

Benutzer- und Rechner-Aliasnamen

Aliasnamen gibt es nicht nur für Kommandos, sondern auch für Benutzer und für Rechner. Sie können also Sachen sagen wie

```
User_Alias WEBMASTERS = hugo susi
Host_Alias WEBHOSTS = www1 www2
◁◁◁◁◁
WEBMASTERS WEBHOSTS = NOPASSWD: /usr/sbin/apache2ctl graceful
```

Rechner-Aliasnamen können Sie auch über IP-Adressen und Netzadressen (mit Netzmaske) definieren:

```
Host_Alias FILESERVERS = red, 192.168.17.1
Host_Alias DEVNET      = 192.168.17.0/255.255.255.0
Host_Alias FINANCENET  = 192.168.18.0/24
```

Überall, wo ein Benutzername stehen darf, können Sie auch auf eine Gruppe verweisen. Geben Sie dafür einfach den Gruppennamen mit einem davorgesetzten Prozentzeichen an:

```
%operators ALL = /usr/local/bin/do-backup
```

Optionen

In /etc/sudoers können Sie zusätzlich zu den Aliasnamen und Regeln noch eine Vielzahl von Optionen angeben, die die Funktion von sudo beeinflussen. Lesen Sie sudo(8) und sudoers(5).

sudoedit

Ein nettes Extra-Gimmick ist »sudo -e« (oder sudoedit). Das Kommando

```
$ sudo -e /etc/hosts
```

ist im wesentlichen äquivalent zu

```
$ sudo cp /etc/hosts /tmp/hosts.$$
$ sudo chown $USER /tmp/hosts.$$
$ vi /tmp/hosts.$$                                    oder siehe VISUAL
$ if ! sudo cmp --quiet /etc/hosts /tmp/hosts.$$ \
> then \
>   sudo cp /tmp/hosts.$$ /etc/hosts && rm -f /tmp/hosts.$$ \
> fi
```

Das heißt, das Kommando legt eine temporäre Kopie der zu editierenden Datei an und ruft mit der Kopie einen Editor auf. Anschließend wird geprüft, ob die Kopie gegenüber dem Original verändert wurde, und falls ja, wird die Kopie über das Original kopiert. Diese Methode hat einige nette Vorteile, nicht zuletzt den, dass der Editor nur mit den Privilegien des normalen Benutzers läuft und dieser deshalb nicht einfach aus dem Editor eine `root`-Shell aufrufen kann. – In der `/etc/sudoers`-Datei wird übrigens nach `sudoedit` gesucht, wenn es darum geht, ob ein Benutzer dieses Kommando ausführen darf.

Übungen

13.13 [!1] Konfigurieren Sie eine `sudo`-Regel, die es Ihnen unter Ihrem normalen Benutzerkonto gestattet, mit `useradd` einen neuen Benutzer anzulegen.

13.14 [2] (»[!2]« für Anwender von Novell/SUSE-Distributionen.) Am Anfang des Abschnitts haben wir erwähnt, dass `sudo` statt des Kennworts des Benutzers auch das Kennwort von `root` abfragen kann. Dazu sagten wir: »Letzteres ist im Großen und Ganzen aber eine dumme Idee.« Welche Gründe könnten uns zu dieser Wertung veranlasst haben?

13.15 [2] Die folgende Konfiguration soll es den Benutzern in `ADMINS` ermöglichen, alle Kommandos als `root` auszuführen, bis darauf, dass sie das `root`-Kennwort nicht ändern können sollen. Warum funktioniert sie nicht wie beabsichtigt?

```
ADMINS   ALL = NOPASSWD: ALL, !/usr/bin/passwd, \
               /usr/bin/passwd [A-z]*, \
               !/usr/bin/passwd root
```

13.6 Grundlegende Netzsicherheit

Für Rechner, die ohne einen Netzwerkanschluss betrieben werden, ist Sicherheit ein wichtiges, aber im Großen und Ganzen überschaubares Gebiet. Leider gilt diese Einschränkung heute nur noch für die wenigsten Rechner, insbesondere wenn wir ein Betriebssystem wie Linux betrachten, das quasi auf dem Internet groß geworden ist und den »Stand der Kunst« für netzwerkfähige Systeme mit definiert.

Als Administrator eines Rechners, der ans Netz angebunden ist, haben Sie also eine besondere Verantwortung. Sie müssen nicht nur dafür sorgen, dass die Dienste, die Ihr Rechner anbietet (egal ob beschränkt auf das lokale Netz Ihres Unternehmens oder frei für das Internet zugänglich) in für Ihre Benutzer verläßlicher Form erbracht werden, sondern Sie müssen auch dafür sorgen, dass Ihr Rechner nicht als Sprungbrett dient, damit von ihm aus andere Teilnehmer am Internet geschädigt werden – sei es durch Spam-Versand oder verteilte *denial-of-service*-Angriffe. Sie können sich nicht hinter Entschuldigungen verstecken wie »Ich bin doch so klein und unbedeutend, niemand sieht ausgerechnet *mich*«, denn diese sind nachweislich falsch – viele Cracker machen sich einen Spaß daraus, systematisch die IP-Adressen abzuklopfen, die Zugangsprovider für ihre Kunden verwenden, und nach unsicheren Rechnern zu suchen.

Wie werden Sie aber dieser Verantwortung gerecht? Die erste Grundregel besteht darin, eine möglichst geringe Angriffsfläche zu bieten. Stellen Sie sicher, dass Ihr Rechner keine Dienste im Internet anbietet, von denen Sie vielleicht gar nicht wissen, dass sie existieren, geschweige denn aktiviert sind. Das können Sie am einfachsten mit einem Kommando wie

```
# netstat -tulp
```

erreichen, das Ihnen eine Liste aller »offenen Ports« präsentiert, also der Dienste, die Ihr Rechner zur Verfügung stellt, mitsamt den IP-Adressen, auf denen er das tut und den verantwortlichen Prozessen. (Siehe hierzu auch Abschnitt 5.5.) Wenn Sie Zugang zu einem entsprechend ausgerüsteten Rechner außerhalb Ihres lokalen Netzes haben, können Sie auch mit `nmap` überprüfen, wie Ihr Rechner sich dem Internet präsentiert.

Früher gab es eine Tendenz bei einigen Linux-Distributoren, alle möglichen Dienste in nur kursorisch konfigurierter Form standardmäßig einzuschalten. Heutzutage ist das zum Glück nicht mehr so, aber die Leninsche Maxime »Vertrauen ist gut, Kontrolle ist besser« ist nach wie vor eine sehr empfehlenswerte Haltung, wenn es um Netzsicherheit geht.

Sie sollten in der Lage sein, jede Zeile der Ausgabe von `netstat` oder `nmap` erklären zu können. Taucht dort etwas auf, das Ihnen nichts sagt, dann gehen Sie der Sache nach.

Dienste deaktivieren

Deaktivieren Sie alle Dienste, die Sie nicht brauchen, indem Sie das betreffende Programm aus der Liste der beim Start hochzufahrenden Daemons entfernen oder in der `/etc/inetd.conf`-Datei auskommentieren (siehe dazu Abschnitt 6.1.2). Wenn Sie den `xinetd` verwenden, dann setzen Sie im Konfigurationsabschnitt des betreffenden Dienstes den Parameter

```
disable = yes
```

(Details siehe Abschnitt 6.3.2.)

Loopback-Schnittstelle

Bei Diensten, die nicht offensichtlich überflüssig sind, ist es oft möglich, ihre Verfügbarkeit so einzuschränken, dass sie nur lokal oder von bestimmten Rechnern aus zugänglich sind. Insbesondere die Möglichkeit, einen Dienst nur auf der Loopback-Schnittstelle (`localhost` oder `127.0.0.1`) zu erbringen, sollten Sie ausnutzen, wo Sie nur können. Beispielsweise ist es sinnvoll, lokalen Programmen eine Möglichkeit einzuräumen, elektronische Post über SMTP einzureichen, aber das heißt nicht, dass Sie das dem ganzen LAN oder Internet erlauben müssen – die gängigen MTAs lassen sich alle so konfigurieren, dass sie den SMTP-Port 25 nur auf `127.0.0.1` öffnen.

Sie können auch den TCP-Wrapper verwenden, um Dienste zum Beispiel nur für Rechner im LAN zur Verfügung zu stellen.

Authentisierung über IP-Adressen

Beachten Sie, dass eine Authentisierung über IP-Adressen im Allgemeinen nicht sicher ist – es ist für einen Angreifer einfach, Datagramme mit beliebigen IP-Adressen zu verschicken (vor allem, wenn es nicht wirklich darauf ankommt, die Ergebnisse zurück zu erhalten). Was einigermaßen zuverlässig funktioniert, ist, mit dem TCP-Wrapper einen Dienst auf den IP-Adressbereich des lokalen Netzes zu beschränken, wenn Sie gleichzeitig etwa durch eine geeignete Konfiguration des Paketfilters Ihres Firewalls dafür sorgen, dass Datagramme von außerhalb Ihres Netzes, die Absenderadressen innerhalb Ihres Netzes behaupten, verworfen werden.

Über die sichere Konfiguration verschiedener Netzwerkdienste wird an anderer Stelle im Linup-Front-Unterlagenzyklus noch viel zu sagen sein – bis hierher waren Netzwerkdienste nicht wirklich unser Thema. Mit dem Thema »Sicherheit« überhaupt beschäftigt sich die Linup-Front-Schulungsunterlage *Linux-Sicherheit*, wo Sie zum Beispiel auch die Konfiguration von Linux als Paketfilter und Firewall erklärt bekommen.

Übungen

13.16 [2] Prüfen Sie, welche Dienste Ihr System nach außen anbietet. Sind alle diese Dienste notwendig?

Kommandos in diesem Kapitel

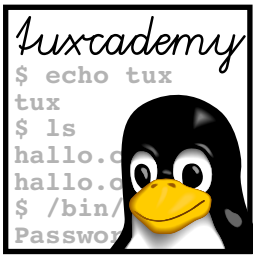
fuser	Identifiziert Prozesse über Dateien oder Sockets	fuser(8)	195
last	Zeige zuletzt angemeldete Benutzer an	last(1)	194
sudo	Erlaubt normalen Benutzern das Aufrufen bestimmter Kommandos mit Administratorprivilegien	sudo(8)	201
sudoedit	Erlaubt normalen Benutzern das Ändern beliebiger Dateien (entspricht „sudo -e“)	sudo(8)	204
ulimit	Legt Ressourcenbegrenzungen für Prozesse fest	bash(1)	197
visudo	Ruft die Datei /etc/sudoers exklusiv zum Ändern auf, mit anschließender Syntaxprüfung	visudo(8)	201
w	Listet die aktuell angemeldeten Benutzer auf (und mehr)	w(1)	194
who	Gibt die Namen der gerade angemeldeten Benutzer aus	who(1)	193
whoami	Gibt den aktuellen (effektiven) Benutzernamen aus	whoami(1)	194

Zusammenfassung

- Sicherheit ist ein wichtiges Thema für alle, die Rechner betreiben.
- Benutzer sollten keine beliebigen Wechselmedien ohne Einschränkungen ins Dateisystem einhängen dürfen. Die mount-Optionen user und users sorgen dafür, dass Gerätedateien und SUID/SGID-Dateien auf einem Medium nicht als solche funktionieren.
- Als Systemverwalter sollten Sie Ihr System auch proaktiv nach Gerätedateien und SUID/SGID-Dateien absuchen, die dort nicht hingehören.
- Die Kommandos who und w erlauben Ihnen, die gerade am System angemeldeten Benutzer anzuschauen, während das Kommando last historische Sitzungsinformationen anzeigt.
- Mit fuser können Sie die Benutzer von Dateien, Verzeichnissen, Dateisystemen oder TCP- und UDP-Ports identifizieren.
- Ressourcenlimits können verwendet werden, um Benutzer oder Systemprozesse davon abzuhalten, Ressourcen zu monopolisieren, die dann anderen Benutzern fehlen.
- Mit sudo können einzelne Benutzer gezielt bestimmte Kommandos mit Administratorprivilegien ausführen. Die Konfiguration des Programms ist allerdings nicht ganz einfach.
- Grundlegende Netzsicherheit beinhaltet, keine überflüssigen Dienste anzubieten und die Dienste, die angeboten werden, auf den kleinstmöglichen Benutzerkreis einzuschränken.

Literaturverzeichnis

McC08 Matt McCutchen. »The suidperl Story«, Mârz 2008.
http://mattmccutchen.net/suidperl.html



A

Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

1.1 Solche Ereignisse werden standardmäßig von `syslogd` in die Datei `/var/log/messages` geschrieben. Am elegantesten lösen Sie die Aufgabe so:

```
# grep 'su: (to root)' /var/log/messages
```

1.2 Tragen Sie in `/etc/syslog.conf` an irgendeiner Stelle die Zeile:

```
*.*                -/var/log/test
```

ein. Danach signalisieren Sie dem `syslogd` mit »`kill -HUP`«, seine Konfigurationsdatei neu zu lesen. Wenn Sie dann in `/var/log` schauen, sollte die neue Datei samt ersten Inhalten (welchen?) bereits vorhanden sein.

1.3 Auf dem empfangenden Rechner muss der `syslogd` mit dem Parameter `-r` gestartet werden (siehe S. 16). Der sendende Rechner braucht eine Konfigurationszeile der Form

```
local0.*     @blue.example.com
```

(wenn der empfangende Rechner den Namen »`blue.example.com`« hat).

1.4 Die einzige sichere Methode besteht darin, das Protokoll dem Zugriff des Angreifers zu entziehen. Sie müssen die Nachrichten darum an einen anderen Rechner schicken. Wenn Sie nicht möchten, dass der Angreifer diesen Rechner auch noch kompromittiert, dann schließen Sie den protokollierenden Rechner über eine serielle Schnittstelle an den Rechner an, der das Protokoll speichert, und konfigurieren Sie `syslogd` so, dass er die Nachrichten an das entsprechende Gerät (`/dev/ttyS0` oder so) schickt. Auf dem Protokollrechner kann ein einfaches Programm die Nachrichten an der seriellen Schnittstelle entgegennehmen und speichern oder weiterverarbeiten. Alternativ können Sie natürlich auch einen (altmodischen) Matrix-Drucker mit Endlospapier verwenden.

1.5 Sie können unter anderem mit Informationen über die Speichermenge und -zuordnung, die vorhandenen Prozessoren, angeschlossene Platten und andere Massenspeicher (IDE und SCSI), USB-Geräte und Netzwerkkarten rechnen. Die Details hängen natürlich von Ihrem System und der Linux-Installation ab.

1.10 Versuchen Sie etwas wie

```
# Die Definition der Quelle setzen wir mal voraus
filter login_hugo {
    facility(authpriv)
      and (match("session opened") or match("session closed"))
      and match("user hugo");
};
destination d_root { usertty("root"); };
log { source(...);
    filter(login_hugo);
    destination(d_root);
};
```

1.12 Schauen Sie in `/etc/logrotate.conf` nach (und gegebenenfalls in `logrotate`(8)).

1.13 Erstellen Sie in `/etc/logrotate.d` eine Datei beliebigen Namens mit folgendem Inhalt:

```
/var/log/test {
    compress
    dateext
    rotate 10
    size 100
    create
}
```

2.1 Textdateien sind grundsätzlich den Standard-Unix-Werkzeugen (`grep` & Co.) zugänglich und darum ideologisch reiner. Man kann sie auch ohne spezialisierte Software anschauen. Außerdem ist das Prinzip sehr gut verstanden und es gibt jede Menge Werkzeuge, die sich um die Auswertung von traditionellen Protokolldateien kümmern. Als Nachteile kann man anführen, dass Textdateien umständlich zu durchsuchen sind und jede Form von gezielter Auswertung entweder extrem mühselig ist oder zusätzliche (nicht standardisierte) Software benötigt. Es gibt keine Form von kryptografischer Absicherung gegen die Manipulation von Protokolleinträgen, und der Umfang der Informationen, die ins Protokoll geschrieben werden können, ist beschränkt.

3.2 ISO/OSI-Schicht 2 beschreibt die Interaktion zwischen zwei direkt miteinander verbundenen Rechnern (z. B. über Ethernet). Schicht 3 beschreibt die Interaktion zwischen Rechnern, die nicht direkt miteinander verbunden sind, umfasst also Routing und medienunabhängige Adressierung (z. B. IP über Ethernet oder Token-Ring oder …).

3.3 Sie können sich Ideen in den Dateien `/etc/services` und (unter Umständen) `/etc/protocols` holen. Die Einordnung müssen Sie allerdings selbst vornehmen. *Tipp:* Praktisch alles, was in `/etc/services` erwähnt wird, kann der Anwendungsschicht zugeordnet werden.

3.4 Eine genaue Antwort läßt sich natürlich nicht geben, aber normalerweise sollte eine TTL von 30–40 mehr als ausreichen. (Der Standardwert ist 64.) Die minimale TTL können Sie ermitteln, indem Sie beginnend bei einer TTL von 1 sukzessive Pakete mit immer höherer TTL an das gewünschte Ziel schicken. Wenn Sie eine ICMP-Fehlermeldung eines Routers bekommen, dass das Paket verworfen wurde, ist die TTL noch nicht groß genug. (Das Programm `traceroute` automatisiert diesen Vorgang.) Diese »minimale« TTL ist natürlich keine Konstante, da IP keinen eindeutigen Weg für die Paketzustellung garantiert.

3.5

1. Die Adresse `172.55.10.3` kann nicht als Stationsadresse verwendet werden, denn sie ist die Broadcast-Adresse dieses Netzes.
2. Die Adresse `138.44.33.12` kann als Stationsadresse benutzt werden.
3. Die Adresse `10.84.13.160` ist die Netzwerkadresse des betreffenden Netzes und steht darum als Stationsadresse nicht zur Verfügung.

3.6 Beispielsweise um bestimmte Netzwerktopologien umzusetzen oder Teile des Adressbereichs an Rechner an unterschiedlichen Standorten zu vergeben (wenn der Provider mitmacht).

3.7 Es gibt insgesamt 16 Subnetze (was man auch daran sehen kann, dass die Subnetzmaske vier Einsen mehr hat als die ursprüngliche Netzmaske). Weitere Subnetze sind `145.2.0.0`, `145.2.16.0`, `145.2.48.0`, `145.2.80.0`, `145.2.96.0`, `145.2.112.0`, `145.2.144.0`, `145.2.176.0`, `145.2.208.0`, `145.2.224.0` und `145.2.240.0`. Die Station mit der IP-Adresse `145.2.195.13` liegt im Subnetz `145.2.192.0`.

4.1 `lsmod` zeigt alle geladenen Module an. »`rmmod` ⟨*Modulname*⟩« entlädt ein Modul, was aber fehlschlägt, wenn das Modul noch verwendet wird.

4.2 Das geht am einfachsten mit `ifconfig`.

4.3 Verwenden Sie »`ifconfig` ⟨*Interface*⟩ ⟨*IP-Adresse*⟩«. Um die Erreichbarkeit der anderen Rechner zu testen, verwenden Sie den Befehl `ping`.

5.2 Erstere sollte größenordnungsmäßig im Zehn-Mikrosekunden-Bereich liegen, letztere je nach Netzwerkinfrastruktur in der Gegend von Millisekunden.

5.3 Probieren Sie etwas wie

```
# ping -f -c 1000000 localhost
```

Die Laufzeit finden Sie am Ende der vorletzten Ausgabezeile von `ping`. (Auf dem System des Autors dauert es knapp 13 Sekunden.)

5.4 Zum Beispiel:

```
$ ping6 ff02::2%eth0
PING ff02::2%eth0(ff02::2) 56 data bytes
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=1 ttl=64 time=12.4 ms
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=2 ttl=64 time=5.27 ms
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=3 ttl=64 time=4.53 ms
Strg+C
--- ff02::2%eth0 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 4.531/7.425/12.471/3.581 ms
```

6.2 FTP ist ein Protokoll, das relativ langlebige »Sitzungen« verwendet: Sie melden sich an, schicken diverse FTP-Kommandos, bekommen Dateien zurückgeschickt (oder schicken selber welche an den Server) und melden sich schließlich wieder ab. Für alle diese Aktionen wird dieselbe TCP-Verbindung benutzt; beim Aufbau der Verbindung kann der `inetd` den FTP-Server starten, der dann bis zum Abbau der Verbindung am Ende der Sitzung weiterläuft. Das dem WWW zugrundeliegende Protokoll HTTP dagegen arbeitet nach dem »Anfrage-Antwort-

Schema«: Ein Browser stellt eine Anfrage nach einer Ressource, etwa `http://www.example.com/test.html`. Hierfür baut er eine TCP-Verbindung zum WWW-Server auf. Der Server schickt ihm die gewünschte Ressource (oder eine Fehlermeldung) über dieselbe TCP-Verbindung und baut dann die TCP-Verbindung wieder ab. Das heißt, der `inetd` würde nur für diese einzelne Ressourcenanfrage einen WWW-Server starten. Stellen Sie sich nun vor, was es bedeutet, eine längere HTML-Seite mit 50 eingebetteten Bildern abzurufen: Der `inetd` müsste 51 WWW-Server-Prozesse erzeugen, die jeweils unabhängig voneinander eine mehr oder weniger umfangreiche Initialisierungsprozedur durchlaufen, bevor sie vielleicht nur 100 Bytes GIF-Daten an den Client schicken. Dies ist sogar für Testzwecke viel zu ineffizient. – Der `inetd` eignet sich nur für Dienste, die entweder ziemlich trivial sind (so dass auch keine langen Initialisierungen anfallen) oder langlebige Sitzungen verwenden. Schon bei SMTP ist es meist sinnvoller, einen freistehenden Server zu verwenden, anstatt bei jeder eingehenden Mailnachricht einen Mailserver vom `inetd` aus zu starten.

6.3 Für die Lösung sollten Sie sich zuerst eine geeignete Portnummer aussuchen und diese in `/etc/services` aufnehmen, zum Beispiel

```
caesar   9999/tcp
```

für den Port 9999. Anschließend genügt im Prinzip eine Zeile der Form

```
caesar  stream  tcp  nowait  root  /usr/bin/tr tr A-Z D-ZA-C
```

in `/etc/inetd.conf`. Das Problem mit dieser genial einfachen Lösung ist nur, dass das `tr`-Programm seine Ausgabe gepuffert schreibt – Sie kriegen den Geheimtext also erst am Programmende zu sehen. Ein einfaches C-Programm wie

```
#include <stdio.h>
#include <ctype.h>

const char code[] = "DEFGHIJKLMNOPQRSTUVWXYZABC";

int
main (void)
{
    int c;
    setvbuf(stdout, (char *)NULL, _IOLBF, 0);
    while ((c = getchar()) != EOF) {
        putchar('A' <= c && c <= 'Z' ? code[c - 'A'] : c);
    }
    return 1;
}
```

kann die Ausgabepufferung dagegen auf »zeilenweise« umstellen, was mehr Spaß macht.

6.4 Ersetzen Sie die Zeile in `/etc/inetd.conf` durch etwas wie

```
ps  stream  tcp  nowait  root  /usr/sbin/tcpd /bin/ps auxw
```

(die tatsächlichen Pfadnamen können abweichen). Anschließend müssen Sie den `tcpd` passend konfigurieren, etwa mit einer Zeile wie

```
ps : ALL EXCEPT 127.0.0.1
```

in der Datei `/etc/hosts.deny`. Alternativ könnten Sie eine Zeile der Form

```
ps : 127.0.0.1
```

nach `/etc/hosts.allow` und einen Eintrag

```
ps : ALL
```

nach `/etc/hosts.deny` schreiben, was aufwändiger ist. Wer wirklich paranoid ist, schreibt »`ALL : ALL`« nach `/etc/hosts.deny` und schaltet nur bestimmte Rechner und Dienste explizit in `/etc/hosts.allow` frei.

6.5 Hierzu können Sie den `/etc/hosts.allow`-Eintrag wie folgt erweitern:

```
ps : 127.0.0.1 : spawn /usr/bin/logger -p local0.info -t ps "%c"
```

6.7 Sie können etwas definieren wie

```
service ps
{
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = root
    server          = /bin/ps
    server_args     = auxw
    only_from       = localhost
    no_access       =
}
```

6.8 Hierzu ist das Attribut `interface` nützlich.

6.9 Die letztere Methode hat den Vorteil, dass definitiv nur lokale Prozesse auf den Dienst zugreifen können, während bei der ersteren prinzipiell Manipulationen an der Quelladresse von Paketen möglich sind, damit andere Rechner Dienste zumindest anstoßen können (und damit vielleicht einen *denial of service* provozieren). In diesem Sinne ist die `127.0.0.1`-Methode sicherer. Natürlich müssen Sie bei den lokalen Prozessen dann darauf achten, dass diese tatsächlich versuchen, mit `localhost` zu reden statt mit dem FQDN des Rechners. Unbequem, aber Sicherheit hat nun mal ihren Preis …

7.4 Hier ist `daytime.socket`:

```
[Unit]
Description=DAYTIME service socket

[Socket]
ListenStream=13
Accept=yes

[Install]
WantedBy=sockets.target
```

Und hier `daytime@.service`:

```
[Unit]
Description=DAYTIME service (per-connection server)
```

```
[Service]
ExecStart=-/bin/date
StandardInput=socket
```

Achten Sie darauf, dass diese Datei eine »Schablone« ist, also `daytime@.service` heißen muss – sonst findet systemd sie nicht, wenn `daytime.socket` aktiviert wird.

Wie würden Sie gegebenenfalls dafür sorgen, dass das Datum im standardisierten englischsprachigen (nicht eingedeutschten) Format erscheint?

8.1 Versuchen Sie etwas wie

```
for tz in America/New_York Europe/Berlin Asia/Tokyo
do
    TZ=$tz xclock -title "$(basename $tz | tr _ ' ')" -update 1 &
done
```

8.2 Laut »`zdump -v /usr/share/zoneinfo/Europe/Berlin`« in den Jahren 1916–18, 1940–49 (mit »doppelter« Sommerzeit in 1945 und 1947) und von 1980 bis heute.

8.3 Bei der Verwendung eines Zeitservers im Internet haben Sie den Vorteil, keine zusätzliche Hardware zu benötigen, allerdings kann eine nicht unerhebliche Netzlast entstehen. Bei zeitbasiert abgerechneten Internetzugängen ist von der Verwendung eines Zeitservers abzuraten, da *de facto* fast permanent eine Verbindung ins Internet gehalten wird. Ebenso ist es notwendig, dass Sie Ihren Firewall so konfigurieren, dass Ihr NTP-Client (der dann innerhalb Ihres Netzes als Zeitserver auf dem nächsthöheren Stratum auftreten kann/sollte) mit dem Zeitserver im Internet Kontakt aufnehmen kann. Dies stellt einen weiteren potenziellen Angriffspunkt für Cracker dar. – Die Funkuhr ist ein weiteres Peripheriegerät, das natürlich Fehler entwickeln kann und gewartet werden muss; zum Glück sind Funkuhren sehr preisgünstig, so dass Sie aus Redundanzgründen zwei oder drei betreiben können … In einem sicherheitskritischen Umfeld ist die Funkuhr an einem Zeitserver in der DMZ die Methode der Wahl, da dadurch die Sicherheit des Firewalls nicht kompromittiert wird. Das DCF77-Signal ist vermutlich auch weniger leicht zu fälschen als NTP-Pakete.

8.6 Beachten Sie, dass Sie zur Virtualisierung hier kein containerbasiertes System verwenden können, da die Uhr dabei normalerweise nicht virtualisiert wird (Container können nicht ihre eigene Uhr unabhängig von der des System-Kerns stellen und schon gar nicht unabhängig schneller oder langsamer laufen lassen, so wie `ntpd` das gerne machen möchte).

8.7 Der Broadcast-Server versendet standardmäßig alle 64 Sekunden ein Datagramm, das die aktuelle Zeit bekanntgibt. Empfängt ein frisch gestarteter `ntpd` auf einem Broadcast-Client ein solches Datagramm, wartet er eine zufällige (kurze) Zeitspanne und startet dann eine Reihe von direkten Anfragen an den Broadcast-Server, um seine Uhr zu stellen und die Verbindung zu kalibrieren. Danach lauscht er nur noch auf weitere Broadcast-Datagramme und läßt die Uhr kontrolliert langsamer oder schneller laufen, um Unterschiede auszugleichen. Wenn die Uhr des Rechners im laufenden Betrieb zu sehr von der Zeit abweicht, die er über NTP bekommt, dann macht er lieber gar nichts als die Uhr zu sprunghaft zu stellen; Sie müssen `ntpd` neu starten, um den automatischen Stellmechanismus zu aktivieren.

9.2 Die Voreinstellung können Sie mit etwas wie

```
$ lpoptions -d lp -o number-up=2
```

machen. Anschließend drucken Kommandos wie `lp` standardmäßig verkleinert. Wenn Sie doch einmal etwas in der Originalgröße drucken wollen, können Sie das mit

```
$ lp -o number-up=1 foo.txt
```

erreichen.

9.7 CUPS erlaubt zwar die Spezifikation diverser Druckoptionen, aber manche Optionen setzen eine manuelle Intervention am Drucker voraus. Beispielsweise könnte es sich für einen einfachen Farbdrucker mit einem Papierschacht lohnen, getrennte Warteschlangen für »normale« Aufträge und für solche einzurichten, die auf hochwertigem Fotopapier gedruckt werden sollen. Im normalen Betrieb ist der Drucker mit einfachem Papier ausgestattet, Aufträge in der »normalen« Warteschlange werden sofort gedruckt, und Foto-Aufträge stauen sich auf. Später können Sie die »normale« Warteschlange anhalten (Aufträge werden zwar angenommen, aber nicht mehr gedruckt), das Papier auswechseln und die »Foto«-Warteschlange freigeben. Wenn alle Fotos gedruckt sind, halten Sie die »Foto«-Warteschlange wieder an, wechseln das Papier zurück und geben die »normale« Warteschlange frei.

10.1 Beim ersten Anmelden sollte `ssh` Sie um die Bestätigung des öffentlichen Rechner-Schlüssels des entfernten Rechners bitten. Beim zweiten Anmeldevorgang ist der öffentliche Rechner-Schlüssel in der Datei `~/.ssh/known_hosts` gespeichert und muss nicht mehr bestätigt werden.

10.2 Im ersten Fall wird die Sitzung abgewiesen, da kein öffentlicher Rechner-Schlüssel für den entfernten Rechner in `~/.ssh/known_hosts` steht. Im zweiten Fall wird die Sitzung ohne weitere Rückfrage aufgebaut.

10.3 Die Datei enthält ohnehin nur öffentliche Schlüssel und bedarf darum keiner besonderen Geheimhaltung.

10.6 Benutzen Sie etwas wie

```
$ ssh-keygen -l -f ~/.ssh/id_rsa.pub
```

(Macht es einen Unterschied, ob Sie `id_rsa.pub` oder `id_rsa` angeben?)

10.7 Nichtinteraktive Programme, die eine SSH-Verbindung nutzen wollen, können in der Regel keine *passphrase* eingeben. Für solche eingeschränkten Fälle ist es denkbar, einen privaten Schlüssel ohne *passphrase* zu verwenden. Dann sollten Sie aber von der Möglichkeit Gebrauch machen, den öffentlichen Schlüssel auf dem entfernten Rechner nur für bestimmte Kommandos nutzbar zu machen (nämlich die, die das nichtinteraktive Programm aufrufen muss). Details dazu stehen in `sshd`(8).

10.8 Versuchen Sie mal »`ssh -X root@localhost`«.

10.9 Eine mögliche Kommandozeile wäre

```
# ssh -L 4711:localhost:7 user@remote.example.com
```

Beachten Sie hierbei, dass das `localhost` aus der Sicht des Rechners `remote` betrachtet wird. `ssh` läßt leider keine symbolischen Namen von *well-known ports* zu, so wie sie in `/etc/services` stehen.

12.2

1. Alice hat selbst die Schlüssel von Bob und von Carla beglaubigt. Diese sind demnach gültig. Doris' Schlüssel ist von Bob beglaubigt, dem Alice »voll« vertraut und dessen Schlüssel sie beglaubigt hat, und ist entsprechend ebenso gültig. Das »etwas« Vertrauen, das Alice in Carla hat, reicht leider nicht aus, damit Erics Schlüssel »gültig« wird.

2. Wenn Alice Carla und Doris »etwas« vertraut, dann wird dadurch Erics Schlüssel »gültig«, da Carla und Doris ihn beide beglaubigt haben. Es existiert eine Kette von beglaubigten Schlüsseln der Länge 2 von Alice zu Carla und von da zu Eric, sowie eine Kette der Länge 3 von Alice über Bob und Doris zu Eric.

3. Erics Schlüssel ist »gültig« und dadurch ist Fionas Schlüssel am Ende einer Kette der Länge 3 (von Alice über Carla und Eric). Die erste Bedingung ist damit erfüllt. Problematisch ist allerdings das Eigentümer-Vertrauen – da alle Pfade von Alice zu Fiona über Eric laufen, kann nur »volles« Vertrauen in Eric Fionas Schlüssel »gültig« machen, jedenfalls solange nicht Bob oder aber Carla *und* Doris Fionas Schlüssel beglaubigen. Wenn also keine zusätzlichen Beglaubigungen stattfinden, sollte Alice sich von Eric überzeugen lassen, dass er kompetent und gewissenhaft mit GnuPG umgehen kann, und ihm »voll« vertrauen.

13.1 Mit direktem Lesezugriff auf die Gerätedatei eines Dateisystems (oder gar einer ganzen Platte) können Sie alle darauf gespeicherten Daten lesen, egal was die Zugriffsrechte der einzelnen Dateien sagen. Dazu brauchen Sie nur ein paar Kenntnisse über die Struktur des verwendeten Dateisystems oder einen großen USB-Stick und einen anderen Linux-Rechner. – Mit direktem Schreibzugriff auf die Gerätedatei eines Dateisystems haben Sie auf dem betreffenden Rechner effektiv Administratorprivilegien, weil Sie beliebige Programme Set-UID `root` machen können (um nur eine naheliegende Möglichkeit zu nennen).

13.6 Verwenden Sie »`who -b`«.

13.7 Eine Lösung (nicht die einzige denkbare) wäre etwas wie

```
#!/bin/sh

last $1 \
    | sed -n '/(/{s/^.*(//; s/).*$//; p}' \
    | tr -c '[0-9\n]' ' ' \
    | awk 'NF == 2 { m += $1; s += $2 }
           NF == 3 { h += $1; m += $2; s += $3 }
           END     { m += int(s/60); s %= 60;
                     h += int(m/60); m %= 60;
                     printf "%d:%02d:%02d\n", h, m, s }'
```

13.9 Verwenden Sie eine Konstruktion wie

```
if [ "$USER" = "hugo" ]; then
  ulimit ...
fi
```

oder

```
if [ $(id -g) = 1000 ]; then
  ulimit ...
fi
```

Bei mehreren verschiedenen Benutzernamen geht auch

```
if grep "^$USER$" /etc/limitusers; then
  ulimit ...
fi
```

(wenn `/etc/limitusers` einen Benutzernamen pro Zeile enthält.)

Natürlich können Sie auch `pam_limits` neu erfinden, indem Sie eine Datei `/etc/userlimits` anlegen, die Zeilen der Form

```
hugo -c unlimited -s 16384
susi -u 128
```

enthält. Sie brauchen dann nur etwas wie

```
eval ulimit $(grep "^$USER$" /etc/userlimits \
  | while read U LIMITS; do echo $LIMITS; done)
```

um die Ressourcenlimits in Kraft zu setzen.

13.14 Gemeinsam verwendete Kennwörter sind prinzipiell ungeschickt – ein Kennwort, das nur Ihnen bekannt ist, können Sie ändern, wann und warum auch immer Sie wollen, aber das Ändern des gemeinsamen `root`-Kennworts ist tendenziell eine größere Sache. Dies vor allem vor dem Hintergrund, dass die Wahrscheinlichkeit dafür, dass ein gemeinsames Kennwort kompromittiert wird, exponentiell in der Anzahl der Leute steigt, die es kennen. Einer der Vorteile von `sudo` ist, dass Sie zu 95% (oder so) ohne ein allgemein bekanntes `root`-Kennwort auskommen können; es genügt, bei der Installation des Rechners eines zu vergeben, das nur für diesen Rechner gilt, und es an einem sicheren Ort zu hinterlegen.

Der qualitative Unterschied ist: Während die Abfrage des `root`-Kennworts sicherstellen soll, dass der betreffende Benutzer `sudo` benutzen darf, soll die Abfrage des *eigenen* Kennworts sicherstellen, dass tatsächlich der berechtigte Benutzer `sudo` aufruft und nicht jemand, der mal eben kurz einen unbeaufsichtigten PC erspäht hat. Das heißt, es wird ein völlig anderes Problem gelöst, nämlich das, dass Sie sich auf das `sudo`-Protokoll halbwegs verlassen können. (Wenn Sie nur das `root`-Kennwort abfragen, kann der `sudo`-berechtigte Benutzer X immer noch den unbeaufsichtigten Rechner des `sudo`-berechtigten Benutzers Y verwenden, um verfängliche Kommandos im Namen von Y abzuschicken, die im Protokoll auch so vermerkt werden.) *Ob* der Benutzer überhaupt `sudo` benutzen darf, ist eine Entscheidung, die Sie in dem Moment treffen, wo Sie den Benutzer in die `sudoers`-Datei eintragen und die eigentlich nicht von Fall zu Fall dadurch überprüft werden müssen sollte, dass der Benutzer das `root`-Kennwort wissen muss.

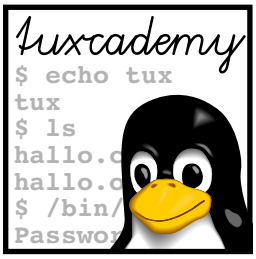
Ferner gilt: Wenn Benutzer das `root`-Kennwort kennen müssen, um `sudo` benutzen zu können, dann müssen Sie Schritte einleiten, damit diese Benutzer nicht auch gleich `su` benutzen oder sich direkt als `root` anmelden und so einerseits die Protokollierung von `sudo` komplett unterlaufen und andererseits von dort aus Dinge tun dürfen, die die `sudo`-Konfiguration ihnen nicht erlaubt hätte.

Sollten Sie als Anwender einer SUSE-Distribution mit dieser Fehlkonfiguration gestraft sein, dann suchen Sie in Ihrer `sudoers`-Datei nach der Zeichenkette

```
Defaults rootpw
```

(oder so ähnlich) und setzen Sie ein Ausrufungszeichen vor das »`rootpw`«, um auf das normale Verhalten umzuschalten. (Die SUSE redet sich damit heraus, dass das voreingestellte Verhalten nötig sei, damit nach der Installation der neue Benutzer auf `sudo` zugreifen könne. Wenn das tatsächlich so wäre, dann müßte man sich fragen, wie Ubuntu, Debian und Apple es anders hinkriegen. Außerdem wäre es kein Fehler seitens der SUSE, die Benutzer auf diesen Umstand an einer sichtbareren Stelle aufmerksam zu machen als in einem Kommentar in `/etc/sudoers`.)

13.15 Alle Bemühungen um den korrekten Aufruf von `passwd` sind irrelevant, weil die Benutzer einfach `/usr/bin/passwd` unter einem anderen Namen kopieren und jenes neue Kommando aufrufen können. (Dieses Beispiel wurde übrigens früher in exakt der dargestellten Form in offiziellen SUSE-Schulungsunterlagen zur Nachahmung empfohlen.)



B

LPIC-1-Zertifizierung

B.1 Überblick

Das *Linux Professional Institute* (LPI) ist eine herstellerunabhängige, nicht profitorientierte Organisation, die sich der Förderung des professionellen Einsatzes von Linux widmet. Ein Aspekt der Arbeit des LPI ist die Erstellung und Durchführung weltweit anerkannter, distributionsunabhängiger Zertifizierungsprüfungen beispielsweise für Linux-Systemadministratoren.

Mit der „LPIC-1"-Zertifizierung des LPI können Sie nachweisen, dass Sie über grundlegende Linux-Kenntnisse verfügen, wie sie etwa für Systemadministratoren, Entwickler, Berater oder Mitarbeiter bei der Anwenderunterstützung sinnvoll sind. Die Zertifizierung richtet sich an Kandidaten mit etwa 1 bis 3 Jahren Erfahrung und besteht aus zwei Prüfungen, LPI-101 und LPI-102. Diese werden in Form von computerorientierten Multiple-Choice- und Kurzantworttests über die Prüfungszentren von Pearson VUE und Thomson Prometric angeboten oder können auf Veranstaltungen wie dem LinuxTag oder der CeBIT zu vergünstigten Preisen auf Papier abgelegt werden. Das LPI veröffentlicht auf seinen Web-Seiten unter `http://www.lpi.org/` die **Prüfungsziele**, die den Inhalt der Prüfungen umreißen.

Prüfungsziele

Die vorliegende Unterlage ist Teil eines Kurskonzepts der Linup Front GmbH zur Vorbereitung auf die Prüfung LPI-101 und deckt damit einen Teil der offiziellen Prüfungsziele ab. Details können Sie den folgenden Tabellen entnehmen. Eine wichtige Beobachtung in diesem Zusammenhang ist, dass die LPIC-1-Prüfungsziele nicht dazu geeignet oder vorgesehen sind, einen Einführungskurs in Linux didaktisch zu strukturieren. Aus diesem Grund verfolgt unser Kurskonzept keine strikte Ausrichtung auf die Prüfungen oder Prüfungsziele in der Form „Belegen Sie Kurs x und y, machen Sie Prüfung p, dann belegen Sie Kurs a und b und machen Sie Prüfung q". Ein solcher Ansatz verleitet viele Kurs-Interessenten zu der Annahme, sie könnten als absolute Linux-Einsteiger n Kurstage absolvieren (mit möglichst minimalem n) und wären anschließend fit für die LPIC-1-Prüfungen. Die Erfahrung lehrt, dass das in der Praxis nicht funktioniert, da die LPI-Prüfungen geschickt so angelegt sind, dass Intensivkurse und prüfungsorientiertes „Büffeln" nicht wirklich helfen.

Entsprechend ist unser Kurskonzept darauf ausgerichtet, Ihnen in didaktisch sinnvoller Form ein solides Linux-Basiswissen zu vermitteln und Sie als Teilnehmer in die Lage zu versetzen, selbständig mit dem System zu arbeiten. Die LPIC-1-Zertifizierung ist nicht primäres Ziel oder Selbstzweck, sondern natürliche Folge aus Ihren neuerworbenen Kenntnissen und Ihrer Erfahrung.

B.2 Prüfung LPI-102

Die folgende Tabelle zeigt die Prüfungsziele der Prüfung LPI-102 (Version 4.0) und die Unterlagen, die diese Prüfungsziele abdecken. Die Zahlen in den Spalten für die einzelnen Unterlagen verweisen auf die Kapitel, die das entsprechende Material enthalten.

Nr	Gew	Titel	ADM1	GRD2	ADM2
105.1	4	Die Shell-Umgebung anpassen und verwenden	–	1–2	–
105.2	4	Einfache Skripte anpassen oder schreiben	–	2–5	–
105.3	2	SQL-Datenverwaltung	–	8	–
106.1	2	X11 installieren und konfigurieren	–	11	–
106.2	1	Einen Display-Manager einrichten	–	11	–
106.3	1	Hilfen für Behinderte	–	12	–
107.1	5	Benutzer- und Gruppenkonten und dazugehörige Systemdateien verwalten	2	–	–
107.2	4	Systemadministrationsaufgaben durch Einplanen von Jobs automatisieren	–	9	–
107.3	3	Lokalisierung und Internationalisierung	–	10	–
108.1	3	Die Systemzeit verwalten	–	–	8
108.2	3	Systemprotokollierung	–	–	1–2
108.3	3	Grundlagen von Mail Transfer Agents (MTAs)	–	–	11
108.4	2	Drucker und Druckvorgänge verwalten	–	–	9
109.1	4	Grundlagen von Internet-Protokollen	–	–	3–4
109.2	4	Grundlegende Netz-Konfiguration	–	–	4–5, 7
109.3	4	Grundlegende Netz-Fehlersuche	–	–	4–5, 7
109.4	2	Clientseitiges DNS konfigurieren	–	–	4
110.1	3	Administrationsaufgaben für Sicherheit durchführen	2	–	4–5, 13
110.2	3	Einen Rechner absichern	2	–	4, 6–7, 13
110.3	3	Daten durch Verschlüsselung schützen	–	–	10, 12

B.3 LPI-Prüfungsziele in dieser Schulungsunterlage

108.1 Die Systemzeit verwalten

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, die Systemzeit korrekt zu halten und die Uhr mit NTP zu synchronisieren.

Wichtigste Wissensgebiete

- Systemzeit und -datum setzen
- Die Hardwareuhr auf die korrekte Zeit in UTC setzen
- Die korrekte Zeitzone einstellen
- Grundlegende NTP-Konfiguration
- Wissen über den Gebrauch des `pool.ntp.org`-Diensts
- Wissen von der Existenz des `ntpq`-Kommandos

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/usr/share/zoneinfo/`
- `/etc/timezone`
- `/etc/localtime`
- `/etc/ntp.conf`
- `date`
- `hwclock`
- `ntpd`
- `ntpdate`
- `pool.ntp.org`

108.2 Systemprotokollierung

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, den Syslog-Daemon zu konfigurieren. Dieses Lernziel umfasst auch die Konfiguration des Syslog-Daemons für den Versand von Ausgabe an einen zentralen Protokollserver oder das Annehmen von Ausgabe als zentraler Protokollserver. Der Gebrauch des journal-Subsystems von systemd ist abgedeckt. Ferner wird Wissen von der Existenz von rsyslog und syslog-ng als alternativen Protokollsystemen verlangt.

Wichtigste Wissensgebiete

- Konfiguration des Syslog-Daemons
- Standard-Facilities, -Prioritäten und -Aktionen verstehen
- Konfiguration von logrotate beherrschen
- Wissen von der Existenz von rsyslog und syslog-ng

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `syslog.conf`
- `syslogd`
- `klogd`
- `/var/log/`
- `logger`
- `logrotate`
- `/etc/logrotate.conf`
- `/etc/logrotate.d`
- `journalctl`
- `/etc/systemd/journald.conf`
- `/var/log/journal/`

108.3 Grundlagen von Mail Transfer Agents (MTAs)

Gewicht 3

Beschreibung Kandidaten sollten von der Existenz der verbreiteten MTA-Programme wissen und einfache Weiterleitungs- und Aliaskonfigurationen auf einem Client-Rechner einstellen können. Andere Konfigurationsdateien werden nicht abgedeckt.

Wichtigste Wissensgebiete

- E-Mail-Aliase anlegen
- E-Mail-Weiterleitung konfigurieren
- Wissen von allgemein verfügbaren MTA-Programmen (postfix, sendmail, qmail, exim) (keine Konfiguration)

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `~/.forward`
- Kommandos in der Sendmail-Emulationsschicht
- `newaliases`
- `mail`
- `mailq`
- postfix
- sendmail
- exim
- qmail

108.4 Drucker und Druckvorgänge verwalten

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, Druckerwarteschlangen und Druckaufträge von Benutzern mit CUPS und der LPD-Kompatibilitätsschnittstelle zu verwalten.

Wichtigste Wissensgebiete

- Grundlegende CUPS-Konfiguration (für lokale und entfernte Drucker)
- Benutzer-Druckerwarteschlangen verwalten
- Allgemeine Druckprobleme lösen
- Druckaufträge zu konfigurierten Druckerwarteschlangen hinzufügen und daraus löschen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- CUPS-Konfigurationsdateien, -Werkzeuge und -Hilfsprogramme
- `/etc/cups/`
- LPD-Kompatibilitätsschnittstelle (`lpr`, `lprm`, `lpq`)

109.1 Grundlagen von Internet-Protokollen

Gewicht 4

Beschreibung Kandidaten sollten ein angemessenes Verständnis der Grundlagen von TCP/IP-Netzen demonstrieren.

Wichtigste Wissensgebiete

- Verständnis von Netzmasken und CIDR-Notation demonstrieren
- Wissen über die Unterschiede zwischen privaten und öffentlichen IP-Adressen als »dotted quads«
- Wissen über gängige TCP- und UDP-Ports (20, 21, 22, 23, 25, 53, 80, 110, 123, 139, 143, 161, 162, 389, 443, 465, 514, 636, 993, 995)
- Wissen über die Unterschiede und wesentlichen Eigenschaften von UDP, TCP und ICMP
- Wissen über die wesentlichen Unterschiede zwischen IPv4 und IPv6
- Wissen über die grundlegenden Eigenschaften von IPv6

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/services`
- IPv4, IPv6
- Subnetting
- TCP, UDP, ICMP

109.2 Grundlegende Netz-Konfiguration

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, Konfigurationseinstellungen auf Client-Rechnern anzuschauen, zu verändern und zu überprüfen.

Wichtigste Wissensgebiete

- Netzschnittstellen manuell und automatisch konfigurieren
- Grundlegende TCP/IP-Rechnerkonfiguration
- Eine Default-Route setzen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/hostname`
- `/etc/hosts`
- `/etc/nsswitch.conf`
- `ifconfig`
- `ifup`
- `ifdown`
- `ip`
- `route`
- `ping`

109.3 Grundlegende Netz-Fehlersuche

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, Netzprobleme auf Client-Rechnern zu lösen.

Wichtigste Wissensgebiete

- Netzschnittstellen und Routingtabellen manuell und automatisch konfigurieren; dies umfasst das Hinzufügen, Starten, Stoppen, neu Starten, Löschen oder Umkonfigurieren von Netzschnittstellen
- Die Routingtabelle ändern, anschauen oder konfigurieren und eine falsch gesetzte Default-Route manuell richtigstellen.
- Probleme mit der Netzkonfiguration finden und lösen.

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `ifconfig`
- `ip`
- `ifup`
- `ifdown`
- `route`
- `host`
- `hostname`
- `dig`
- `netstat`
- `ping`
- `ping6`
- `traceroute`
- `traceroute6`
- `tracepath`
- `tracepath6`
- `netcat`

109.4 Clientseitiges DNS konfigurieren

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, DNS auf einem Client-Rechner einzurichten.

Wichtigste Wissensgebiete

- Anfragen an entfernte DNS-Server stellen
- Lokale Namensauflösung konfigurieren und entfernte DNS-Server verwenden
- Die Reihenfolge der Namensauflösung ändern

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- `host`
- `dig`
- `getent`

110.1 Administrationsaufgaben für Sicherheit durchführen

Gewicht 3

Beschreibung Kandidaten sollten wissen, wie sie die Systemkonfiguration prüfen, um die Sicherheit des Rechners in Übereinstimmung mit örtlichen Sicherheitsrichtlinien zu gewährleisten.

Wichtigste Wissensgebiete

- Ein System nach Dateien mit gesetztem SUID/SGID-Bit durchsuchen
- Benutzerkennwörter und die Informationen über das Altern von Kennwörtern setzen oder ändern
- Mit nmap und netstat offene Ports auf einem System finden können
- Grenzen für Benutzeranmeldungen, Prozesse und Speicherverbrauch setzen
- Feststellen, welche Benutzer sich angemeldet haben oder gerade angemeldet sind
- Grundlegende Konfiguration und Gebrauch von sudo

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `find`
- `passwd`
- `fuser`
- `lsof`
- `nmap`
- `chage`
- `netstat`
- `sudo`
- `/etc/sudoers`
- `su`
- `usermod`
- `ulimit`
- `who`, `w`, `last`

110.2 Einen Rechner absichern

Gewicht 3

Beschreibung Kandidaten sollten wissen, wie sie grundlegende Rechnersicherheit konfigurieren können.

Wichtigste Wissensgebiete

- Kenntnisse über Shadow-Kennwörter und wie sie funktionieren
- Nicht verwendete Netzdienste abschalten
- Die Rolle der TCP-Wrapper verstehen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/nologin`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/xinetd.d/`
- `/etc/xinetd.conf`
- `/etc/inet.d/`
- `/etc/inetd.conf`
- `/etc/inittab`
- `/etc/init.d/`
- `/etc/hosts.allow`
- `/etc/hosts.deny`

110.3 Daten durch Verschlüsselung schützen

Gewicht 3

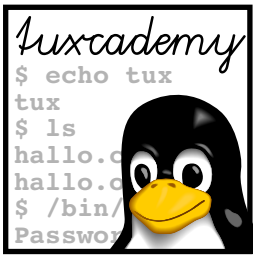
Beschreibung Der Kandidat sollte in der Lage sein, Public-Key-Techniken zum Schutz von Daten und Kommunikation einzusetzen.

Wichtigste Wissensgebiete

- Einen OpenSSH-2-Client grundlegend konfigurieren und verwenden
- Die Rolle von OpenSSH-2-Rechnerschlüsseln verstehen
- GnuPG grundlegend konfigurieren und verwenden, inklusive Schlüsselrückruf
- SSH-Port-Tunnel (auch X11-Tunnel) verstehen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `ssh`
- `ssh-keygen`
- `ssh-agent`
- `ssh-add`
- `~/.ssh/id_rsa` und `id_rsa.pub`
- `~/.ssh/id_dsa` und `id_dsa.pub`
- `/etc/ssh/ssh_host_rsa_key` und `ssh_host_rsa_key.pub`
- `/etc/ssh/ssh_host_dsa_key` und `ssh_host_dsa_key.pub`
- `~/.ssh/authorized_keys`
- `/etc/ssh_known_hosts`
- `gpg`
- `~/.gnupg/`



C

Kommando-Index

Dieser Anhang fasst alle im Text erklärten Kommandos zusammen und verweist auf deren Dokumentation sowie die Stellen im Text, wo die Kommandos eingeführt werden.

arp Erlaubt Zugriff auf den ARP-Cache (Abbildung von IP- auf MAC-Adressen) arp(8) 53

cancel Storniert einen Druckauftrag cancel(1) 141

date Gibt Datum und Uhrzeit aus date(1) 125

dig Sucht Informationen im DNS (sehr komfortabel) dig(1) 96

dmesg Gibt den Inhalt des Kernel-Nachrichtenpuffers aus dmesg(8) 19

dnsmasq Ein einfacher DHCP- und cachender DNS-Server für kleine Installationen dnsmasq(8) 82

fuser Identifiziert Prozesse über Dateien oder Sockets fuser(8) 195

getent Ruft Einträge aus administrativen Datenbanken ab getent(1) 97

gpg Verschlüsselt und signiert Dateien gpg(1) 175

host Sucht Informationen im DNS host(1) 95

hwclock Steuert die CMOS-Uhr eines PCs hwclock(8) 124

ifconfig Konfiguriert Netzwerk-Schnittstellen ifconfig(8) 71

ifdown Schaltet eine Netzwerk-Schnittstelle aus (Debian) ifdown(8) 76

ifup Schaltet eine Netzwerk-Schnittstelle ein (Debian) ifup(8) 76

inetd Internet-Superserver, überwacht Ports und startet ggf. Dienste inetd(8) 104, 57

ip Verwaltet Netzwerkschnittstellen und Routing ip(8) 75

ipv6calc Hilfsprogramm für Berechnungen mit IPv6-Adressen ipv6calc(8) 66

klogd Akzeptiert Protokollnachrichten des Systemkerns klogd(8) 14, 18

last Zeige zuletzt angemeldete Benutzer an last(1) 194

logger Macht Einträge ins Systemprotokoll logger(1) 17

logrotate Verwaltet, kürzt und „rotiert“ Protokolldateien logrotate(8) 27

logsurfer Programm zum Durchsuchen des Systemprotokolls nach wichtigen Ereignissen www.cert.dfn.de/eng/logsurf/ 18

lp Reicht einen Druckauftrag ein lp(1) 139

lpadmin Dient zur Verwaltung von Druckerwarteschlangen lpadmin(8) 146

lpinfo Zeigt verfügbare Drucker-Geräte und -Treiber an lpinfo(8) 146

lpoptions Verwaltet Standardeinstellungen für Druckerwarteschlangen lpoptions(1) 141

lpq Gibt den Status einer Druckerwarteschlange aus lpq(1) 140

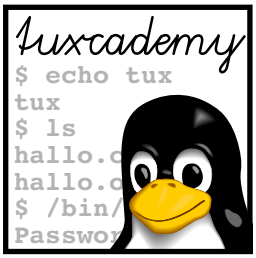
lpr Reicht einen Druckauftrag ein lpr(1) 138

lprm Storniert einen Druckauftrag lprm(1) 141

mailq Zeigt den Zustand der Mail-Warteschlange an (Sendmail & Co.) sendmail(1) 168

netcat Sehr allgemeines Netzwerk-Client-Programm nc(8) 99
newaliases Aktualisiert die Alias-Datenbank des Mailservers (Sendmail/Postfix) newaliases(1) 169
nmap Netzwerk-Portscanner, analysiert offene Ports auf Rechnern nmap(1) 94
ntp-keygen Erzeugt Schlüsselmaterial für ntpd ntp-keygen(8) 129
ntpq Steuert NTP-Server ntpq(8) 130
ping Prüft grundlegende Netzwerk-Konnektivität mit ICMP ping(8) 87
ping6 Prüft grundlegende Netzwerk-Konnektivität (für IPv6) ping(8) 88
pstops Bereitet PostScript-Druckaufträge für CUPS auf 143
route Verwaltet die statische Routing-Tabelle im Linux-Kern route(8) 72
scp Sicheres Dateikopierprogramm auf SSH-Basis scp(1) 155
sendmail MTA-Administrationskommando (Sendmail, aber auch andere – kompatible – MTAs) sendmail(1) 168
sftp Sicheres FTP-artiges Programm auf SSH-Basis sftp(1) 156
ssh „Secure Shell“, erlaubt sichere interaktive Sitzungen auf anderen Rechnern ssh(1) 152
ssh-add Akkreditiert private Schlüssel beim ssh-agent ssh-add(1) 158
ssh-agent Verwaltet private Schlüssel und Kennwörter für die SSH ssh-agent(1) 158
ssh-copy-id Kopiert öffentliche SSH-Schlüssel auf andere Rechner ssh-copy-id(1) 158
ssh-keygen Generiert und verwaltet Schlüssel für die SSH ssh-keygen(1) 157
sshd Server für das SSH-Protokoll (sicherer interaktiver Fernzugriff) sshd(8) 152
sudo Erlaubt normalen Benutzern das Aufrufen bestimmter Kommandos mit Administratorprivilegien sudo(8) 201
sudoedit Erlaubt normalen Benutzern das Ändern beliebiger Dateien (entspricht „sudo -e“) sudo(8) 204
syslog-ng Bearbeitet Systemprotokoll-Meldungen (neuer und besser) syslog-ng(8) 23
syslogd Bearbeitet Systemprotokoll-Meldungen syslogd(8) 14
tail Zeigt das Ende einer Datei an tail(1) 18
tcpd „TCP-Wrapper“, erlaubt oder verbietet Zugriff auf Dienste in Abhängigkeit von der IP-Adresse des Clients tcpd(8) 106
tcpdump Netzwerk-Sniffer, protokolliert und analysiert Netzwerkverkehr tcpdump(1) 99
telnet Erlaubt Verbindungen zu beliebigen TCP-Diensten, insbesondere TELNET (Fernzugriff) telnet(1) 98
tracepath Prüft die Wegleitung zu einer anderen Station, mit Pfad-MTU tracepath(8) 91
tracepath6 Entspricht tracepath, aber für IPv6 tracepath(8) 92
traceroute Prüft die Wegleitung zu einer anderen Station traceroute(8) 89
ulimit Legt Ressourcenbegrenzungen für Prozesse fest bash(1) 197
vacation Automatischer E-Mail-Beantworter, etwa für längere Abwesenheiten vacation(1) 170
visudo Ruft die Datei /etc/sudoers exklusiv zum Ändern auf, mit anschließender Syntaxprüfung visudo(8) 201
w Listet die aktuell angemeldeten Benutzer auf (und mehr) w(1) 194
who Gibt die Namen der gerade angemeldeten Benutzer aus who(1) 193
whoami Gibt den aktuellen (effektiven) Benutzernamen aus whoami(1) 194
wireshark Paket-Sniffer, liest und analysiert Netzwerkverkehr (Ex-ethereal) wireshark(1) 100
xconsole Zeigt Systemmeldungen in einem X-Fenster an xconsole(1) 14
xinetd Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste xinetd(8) 108, 57
xlogmaster X11-basiertes Systembeobachtungsprogramm xlogmaster(1), www.gnu.org/software/xlogmaster/ 18

zdump Gibt die aktuelle Zeit oder die Zeitzonendefinitionen für verschiedene Zeitzonen aus zdump(1) 125
zic Compiler für Zeitzonen-Dateien zic(8) 125



Index

Dieser Index verweist auf die wichtigsten Stichwörter in der Schulungsunterlage. Besonders wichtige Stellen für die einzelnen Stichwörter sind durch **fette** Seitenzahlen gekennzeichnet. Sortiert wird nur nach den Buchstaben im Indexeintrag; „`~/.bashrc`“ wird also unter „B“ eingeordnet.

Allman, Eric, 166
`apt`, 172
`apt-cache`, 144
`arp`, 53
`awk`, 197

Backends, **137**
`bash`, 158
    `-p` (Option), 191
`~/.bash_profile`, 200
Berkeley-LPD, **136**
Bernstein, Dan J., 166
Bernstein, Daniel J., 154
Broadcast-Adresse, **59**

`cancel`, 141
`chrony`, 130
Common Unix Printing System, **137**
`cp`, 156
`cron`, 28, 126, 130, 157, 160, 168, 192
CUPS, **137**

Datagramme, **51**
`date`, 125–126, 132
    `-u` (Option), 125
Definitionen, **12**
`/dev`, 70, 190, 192
`/dev/hda`, 193
`/dev/klog`, 41
`/dev/kmem`, 191
`/dev/log`, 14, 24, 34
`/dev/lp0`, 136
`/dev/null`, 107
`/dev/xconsole`, 16
`dig`, 95–97
    `-x` (Option), 96
`DISPLAY` (Umgebungsvariable), 159–160
`dmesg`, 19
    `-c` (Option), 19
    `-n` (Option), 19
`dnsmasq`, 82
`domain` (`/etc/resolv.conf`), 81
Druckoptionen, 141

`echo`, 74
`EDITOR` (Umgebungsvariable), 201
Elgamal, Taher, 176
Empfang von Nachrichten, **167**
`/etc/adjtime`, 126
`/etc/aliases`, 169
`/etc/cups/cupsd.conf`, 146, 148
`/etc/cups/mime.convs`, 142
`/etc/cups/mime.types`, 142
`/etc/cups/ppd`, 146
`/etc/cups/printers.conf`, 146
`/etc/fstab`, 117, 191
`/etc/hosts`, 82, 97
`/etc/hosts.allow`, 106–108, 212–213
`/etc/hosts.deny`, 106, 108, 212–213
`/etc/inetd.conf`, 104, 106–109, 111, 206, 212
`/etc/init.d/network`, 77
`/etc/init.d/networking`, 76
`/etc/init.d/xinetd`, 111
`/etc/localtime`, 125
`/etc/logrotate.conf`, 28–29
`/etc/logrotate.d`, 28, 210
`/etc/machine-id`, 41
`/etc/mail/aliases`, 169
`/etc/modules.conf`, 70
`/etc/network/interfaces`, 76, 79, 92
`/etc/network/options`, 74
`/etc/nsswitch.conf`, 83, 97
`/etc/ntp.conf`, 127
`/etc/pam.d/login`, 200
`/etc/passwd`, 98, 156
`/etc/printcap`, **137**
`/etc/printcap`, 137, 229
`/etc/profile`, 200
`/etc/protocols`, 104, 109, 210
`/etc/resolv.conf`, 81
    `domain`, 81
    `nameserver`, 81

options, 82
search, 81
sortlist, 81
/etc/rsyslog.conf, 14, 20, 22
/etc/security/limits.conf, 200
/etc/services, 57, 98, 104, 109, 119, 196, 210, 215
/etc/shadow, 156
/etc/ssh, 153
/etc/ssh/ssh_config, 159
/etc/ssh/sshd_config, 158–159
/etc/sudoers, 201–202, 204, 217
/etc/sysconfig, 76–77
/etc/sysconfig/network, 76
/etc/sysconfig/network-scripts, 77–78
/etc/sysconfig/network-scripts/ifcfg-eth0, 77
/etc/sysconfig/network/config, 77
/etc/sysconfig/network/routes, 77
/etc/sysconfig/static-routes, 78
/etc/sysconfig/sysctl, 74
/etc/sysctl.conf, 75, 81
/etc/syslog-ng/syslog-ng.conf, 23
/etc/syslog.conf, 14, 17, 20, 26, 209
/etc/systemd/journald.conf, 35–37
/etc/timezone, 125
/etc/udev/rules.d, 70
/etc/udev/rules.d/70-persistent-net.rules, 86
/etc/xinetd.conf, 109, 111
/etc/xinetd.d, 111
ethereal, 100–101, 226
Ethernet, 44
Exim, 166
extended Internet daemon, **108**

fg, 99
Filter, **137**
find, 192
-ls (Option), 192
-perm (Option), 192
-print (Option), 192
-type (Option), 192
Flags, **55**
~/.forward, 170
Fragmentierung, **51**
fuser, 195–196, 207
-i (Option), 196
-k (Option), 196
-m (Option), 196
-n (Option), 196
-v (Option), 196
-w (Option), 196

gated, 72–73
Gerhards, Rainer, 19
getent, 97
getmail_fetch, 161
~/.gnupg, 186
gpg, 175, 179, 181–187
--armor (Option), 178, 183, 186
--check-sigs (Option), 181
--clearsign (Option), 185
--decrypt (Option), 185
--detach-sign (Option), 186
--edit-key (Option), 181
--export (Option), 178
--gen-key (Option), 175
--list-keys (Option), 178–179
--list-sigs (Option), 181
--output (Option), 178, 183, 185
--recipient (Option), 183, 187
--sign (Option), 184
--symmetric (Option), 183
--verify (Option), 185–186
gpg.conf, 187
grep, 192, 210
-v (Option), 192
gzip, 30
-6 (Option), 30

Hazel, Philip, 166
host, 95–97
-a (Option), 96
-l (Option), 96
-t (Option), 96
hugo, 203
hwclock, 124–126
--date (Option), 126
--systohc (Option), 126

IANA, 57
id_ed25519, 158
id_ed25519.pub, 158
id_rsa, 157
id_rsa.pub, 157–158
ifconfig, 71, 73, 75, 81, 86–87, 211
-a (Option), 86
ifdown, 76–78, 92
-a (Option), 76
ifup, 76–78, 92
-a (Option), 76
inetd, 57, 104–106, 108, 111–112, 116, 118–120, 167, 211
inetd.conf, 105
Internet Printing Protocol, **137**
ip, 75–76, 81, 86, 89
addr (Option), 75
addr add (Option), 75
brd + (Option), 75
help (Option), 75
link (Option), 75
local (Option), 75
route (Option), 75
ipv6calc, 66–67
itox, 112

journalctl, 36–42

-b (Option), 39
-f (Option), 38
-k (Option), 39
--list-boots (Option), 39
-n (Option), 38
--no-pager (Option), 36
--output=verbose (Option), 40
-p (Option), 38–39
--since (Option), 39
-u (Option), 38–39
--until (Option), 39
journald, 39, 42

kill, 199
killall, 200
klogd, 14, 18–19, 23, 25
kprinter, 142

last, 194–195, 197, 207
-f (Option), 195
less, 18, 36
logger, 17, 23, 27, 37
logrotate, 28–31, 37
-f (Option), 28
--force (Option), 28
logsurfer, 18
Lokale Netze, **47**
lp, 137–139, 141, 214
-o (Option), 141
lpadmin, 146
-D (Option), 146
-E (Option), 146
-L (Option), 146
-m (Option), 146
-p (Option), 146
-v (Option), 146
-x (Option), 146
lpinfo, 146
-m (Option), 146
-v (Option), 146
~/.lpoptions, 141
lpoptions, 141
-l (Option), 141
-o (Option), 141
-p (Option), 141
lpq, 137, 140–141
-a (Option), 140
-l (Option), 140
-P (Option), 140
lpr, 137–141
-o (Option), 141
-P (Option), 138
lprm, 137, 141
-P (Option), 141
lpstat, 137, 140
ls, 192
-l (Option), 192
lsmod, 86, 211
lspci, 86
-k (Option), 86

mailq, 168
mime.types, 142
mount, 191, 193
nodev (Option), 193
noexec (Option), 193
nosuid (Option), 193
mpage, 138
mysqladmin, 204

nameserver (/etc/resolv.conf), 81
NAT, 63
nc, 99
netcat, 99
netdate, 126
netstat, 92–94, 206
-l (Option), 93
-t (Option), 93
-u (Option), 93
Netzklassen, **61**
Netzmaske, **59**
Netzwerkadresse, **59**
newaliases, 169
nmap, 92, 94–95, 100, 205–206
-A (Option), 95
NSA, 153
nslookup, 95
ntp-keygen, 129
ntpd, 127–130, 132, 214
ntpdate, 130, 132
ntpq, 130–131
-p (Option), 130

OpenSSH, **152**
options (/etc/resolv.conf), 82

PAGER (Umgebungsvariable), 36
passwd, 217
ping, 52, 87–90, 211
-a (Option), 88
-b (Option), 88
-c (Option), 88
-f (Option), 88
-I (Option), 88
-i (Option), 88
-n (Option), 88
-s (Option), 88
ping6, 88–89
Portnummern, **55**
Ports, 56
Portscanner, **94**
postalias, 169
Postfix, 166
PPD-Dateien, **143**
pppd, 77
PRINTER (Umgebungsvariable), 138, 140
/proc/kmsg, 18–19
/proc/sys/kernel/threads-max, 199

Prüfungsziele, **219**
pstops, 143

Qmail, 166
qmail-qread, 168
qmail-qstat, 168
qmail-send, 168

rcnetwork, 77
registered ports, **57**
rm, 18
rmmod, 211
route, 72, 74, 76–77, 89
    -host (Option), 74
    -net (Option), 74
routed, 72
Routing, **54**
/run/log/journal, 35

Scheidler, Balazs, 23
scp, 155–156, 158
    -r (Option), 156
search (/etc/resolv.conf), 81
sed, 197
Sendmail, 166
sendmail, 168–169
    -bi (Option), 169
    -bp (Option), 168
    -q (Option), 168
sftp, 156, 158
sh, 203
sleep, 161, 200
Snowden, Edward, 153
sort, 55
sortlist (/etc/resolv.conf), 81
Spooler, **136**
~/.ssh, 157
ssh, 55, 98, 120, 152–156, 158–162, 173, 215
    -f (Option), 161
    -KR (Option), 161
    -L (Option), 160–161
    -N (Option), 160
    -R (Option), 160–161
    -X (Option), 159
ssh-add, 158–159
    -D (Option), 158
ssh-agent, 158–159
ssh-copy-id, 158
ssh-keygen, 153, 157–159
    -f (Option), 153
    -l (Option), 153
    -t ed25519 (Option), 158
~/.ssh/authorized_keys, 158
~/.ssh/config, 154, 159
~/.ssh/known_hosts, 154–155, 215
~/.ssh/ssh_config, 155
sshd, 98, 118, 152, 159
    -i (Option), 118
su, 17, 201
Subnetting, **62**
sudo, 201–204, 207, 217, 226
    -e (Option), 204
    -k (Option), 201
sudo cupsenable, 203
sudoedit, 204
suidperl, 191
SuSEconfig, 76
sysctl, 80
Syslog-NG, 23
syslog-ng, 23
syslog.conf, 17
syslogd, 14, 16–21, 23–24, 26–28, 37, 209
    -r (Option), 16, 24, 209
systemctl, 34–35, 116, 120
    -l (Option), 35
systemd-journald, 35–37
SYSTEMD_LESS (Umgebungsvariable), 36
SYSTEMD_PAGER (Umgebungsvariable), 36

tail, 18, 38
    -f (Option), 18, 38
tar, 186
tcpd, 106, 108, 120, 212
tcpdump, 99–100, 162
telnet, 98–99, 106
tr, 212
tracepath, 89, 91–92
tracepath6, 92
traceroute, 89–92, 210
    -6 (Option), 91
    -I (Option), 90
    -M tcp (Option), 90
    -p (Option), 90
    -T (Option), 90
traceroute6, 91–92
TZ (Umgebungsvariable), 125

ulimit, 197–199
    -a (Option), 197
    -e (Option), 198
    -H (Option), 197
    -S (Option), 197
ulimits, 200
umask, 107
Umgebungsvariable
    DISPLAY, 159–160
    EDITOR, 201
    PAGER, 36
    PRINTER, 138, 140
    SYSTEMD_LESS, 36
    SYSTEMD_PAGER, 36
    TZ, 125
    VISUAL, 201, 204
    XINETD_BIN, 111
useradd, 205
/usr/bin, 203
/usr/lib/cups/backend, 143

/usr/share/cups/model, 146
/usr/share/zoneinfo, 125
UTC, 124

vacation, 170
~/.vacation.msg, 170
/var/log, 17, 35, 42
/var/log/cups/error_log, 148
/var/log/journal, 35
/var/log/messages, 36, 209
/var/log/wtmp, 195
/var/mail, 167–169
/var/qmail, 168
/var/qmail/alias, 169
/var/run/xinetd.dump, 111
/var/spool/exim, 168
/var/spool/mail, 167
/var/spool/mqueue, 168
/var/spool/postfix, 168
Venema, Wietse, 166
verbindungsloses Protokoll, **51**
Versand von Nachrichten, **167**
vi, 201–202
VISUAL (Umgebungsvariable), 201, 204
visudo, 201

w, 194, 196, 207
    -h (Option), 194
    -s (Option), 194
Warteschlange, **136**
Weitverkehrsnetze, **47**
well-known ports, **57**
who, 193–194, 196, 207, 216
    -a (Option), 194
    -b (Option), 194, 216
    -H (Option), 194
    -m (Option), 194
    -r (Option), 194
whoami, 194
wireshark, 100, 162
Wowereit, Klaus, 191

.Xauthority, 160
xclock, 126
xconsole, 14
xinetd, 57, 108–109, 111–112, 116, 118, 120, 167, 206
xinetd.conf, 109
XINETD_BIN (Umgebungsvariable), 111
xlogmaster, 18

zdump, 125–126
Zertifikat, **173**
zic, 125
Zimmermann, Phil, 186