Debian GNU/Linux-Server im Eigenbau

Michael Kaaden

dsp@galileo.franken.de

10. Januar 2006

Version 1.17

Build 378

Diese Anleitung wurde mit LaTeX 2_ε gesetzt.

Aktuelle Version siehe http://www.kaaden.franken.de/papers/DebianServer/.

Für Tobias,

der seinen Papa schon vor der Geburt
mit Füßen tritt . . .

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I

Einführung

KAPITEL 1

Vorwort

Spätestens seit den ziemlich genialen IBM-Werbespots weiß selbst der durchschnittliche *Bild*-Zeitungs-Leser, dass es a) Server gibt und b) die etwas mit Computern und Rechenzentren zu tun haben. Besser Informierte wissen, dass Server heutzutage in nahezu allen Unternehmen stehen (bis vielleicht auf die ganz kleinen Klitschen, die noch mit Dampffax arbeiten) und dort unterschiedlichste, aber immer äußerst nützliche Dienste verrichten.

Schnelle Prozessoren, große Speicherkapazitäten sowie Netzwerkhardware sind auch für Privatanwender mehr als nur erschwinglich geworden. Stabile und für den Serverbetrieb geeignete Betriebssysteme sind kostenlos verfügbar. Es gibt damit keinen Grund mehr, sich die Segnungen eines Servers zu Hause zu versagen, der Druckdienste anbietet, IP-Adressen an angeschlossene Rechner verteilt, Namensdienste bereitstellt, das MP3-Archiv vorhält, E-Mail verteilt, nachdem er daraus daraus Spam und Viren gefiltert hat, Webseiten cachet, Faxe empfängt, Anrufbeantworter spielt und noch viele weitere Aufgaben erfüllt. Vielleicht sind auch Deine *Carrera*-Rennbahn oder *Märklin*-Modelleisenbahn von Deiner besseren Hälfte dauerhaft in Kisten auf den Dachboden verbannt worden, so dass Du einfach ein neues interessantes Spielzeug „brauchst“ …

Diese Anleitung beschäftigt sich mit der Einrichtung eines Servers, der unter *Debian GNU/Linux* in Version 3.1 (*Sarge*[*] genannt) läuft. Ich halte sie in einem lockeren Plauderton, als hättest Du mich zu einem leckeren Abendessen eingeladen, um Dir einiges von meinem Wissen abzuzapfen (weshalb ich Dich auch duze). Der Grund für diesen für Dich vielleicht etwas ungewöhnlich wirkenden Schreibstil ist einfach der, dass ich der Meinung bin, dass das Ganze so weniger trocken auf Dich wirkt. Und einschlafen sollst Du dabei wirklich nicht, da Du sonst nur Fehler machst. Außerdem hat man sich als Informatiker tatsächlich daran gewöhnt, von Leuten angerufen oder gar eingeladen zu werden, die man zwar überhaupt nicht kennt, die aber der beste Freund oder die beste Freundin des Bruders des Schwagers des Vorbesitzers des Goldfischs Deiner Tante dritten Grades sind. Aber man ist ja höflich (und einem guten Essen selten abgeneigt) und lässt das über sich ergehen.[†]

[*] zu den Codenamen siehe http://www.debian.de/releases/

[†] Nur dann, wenn diese Leute, die man eigentlich gar nicht kennt, dauernd wegen irgendwelcher ach so wichtiger Probleme anrufen, während man gerade mit den Kindern spielt, sie ins Bett bringen will oder

1.1 README.1ST

Ich beschreibe im Folgenden, wie ich *meinen* Server unter Debian GNU/Linux 3.1 (auch bekannt als *Sarge*) eingerichtet habe. Diese Anleitung dient primär mir selbst als Gedächtnisstütze, wenn ich das System einmal neu einrichten muss oder mich jemand fragt, wie ich dieses oder jenes Problem gelöst habe. Daher ist diese Anleitung keine vollständige Einführung in die Einrichtung und Benutzung eines *beliebigen* Debian GNU/Linux-Systems – und will auch keine sein.

1.1.1 Haftungsausschluß

Es wäre ziemlich uncool von Dir, wolltest Du mich für Schäden oder sonstige negative Folgen aus dem Befolgen bzw. Nichtbefolgen meiner Ausführungen in dieser Anleitung verantwortlich machen. Weder mittelbar noch unmittelbar. Nix. Nada. Es gilt für Dich: Erst denken, dann handeln und schließlich freuen oder ärgern – aber ärgere Dich über Dich selbst, nicht über mich. Gibt sonst schlechtes Karma. Wenn Du Deine Kiste nicht eigenverantwortlich einrichten kannst oder willst, ist für Dich jetzt der Zeitpunkt gekommen, mit dem Lesen – und vor allem *Befolgen* – dieser Anleitung aufzuhören. Das ist aber kein Drama, denn es gibt genug Dienstleister, die sich freuen, Dir die Einrichtung eines solchen Servers für einen Sack voll Gold abzunehmen.

1.1.2 Einschränkungen

Wenn Du, der geneigte Leser, die im weiteren Verlauf erläuterten Schritte auf Deinem eigenen System nachvollziehen möchtest, so sei mir willkommen. Aber Deine Software, Hardware und Bedürfnisse sind möglicherweise andere als meine. Das ist dann schade für Dich, denn dann wirst Du Deine kleinen grauen Zellen etwas anstrengen müssen. Aber Du verstehst sicher, dass ich keine allgemeingültige Anleitung liefern kann, wie Du in jedem möglichen Fall vorzugehen hast. Was meine Installation angeht, werde ich allerdings oft Begründungen dafür angeben, warum ich etwas so gelöst habe, wie ich es in dieser Anleitung beschreibe.

1.1.3 Nervausschluß

Ganz wichtig: Für die meisten Leser und ganz besonders für Dich ist es bestimmt selbstverständlich, aber lass es mich sicherheitshalber noch etwas präziser ausdrücken, um meine Freizeit zu schützen (und meine Nerven zu schonen): Auch wenn es Dir noch so verlockend erscheinen mag, widerstehe der Versuchung, mir Fragen zu Hard- oder Software zu stellen, die in meiner Anleitung gar nicht vorkommen oder anders verwendet werden, als

sie gerade ins Bett gebracht hat und jetzt etwas Ruhe haben könnte, wird es lästig. Allerdings habe ich die Erfahrung gemacht, dass es reicht, ein paar wohlformulierte Bemerkungen darüber fallen zu lassen, wie lächerlich dieses Problem doch eigentlich sei, dass die weniger Selbstbewussten sich *nie* mehr melden. Bei den ganz Hartnäckigen reicht es, € 25 pro angefangene zehn Minuten Telefonberatung zu verlangen, dann lassen die das auch ganz schnell bleiben …

Du es Dir vorstellst. Sorry, aber an dieser Stelle musst Du ein richtiger Mann bzw. eine richtige Frau sein und Dir selbst helfen. Entsprechende E-Mails gedenke ich auch nicht zu beantworten. Du musst verstehen, dass ich kein Ersatz für die in Deiner Umgebung bestimmt vorhandene LINUX-User-Group und die zahlreichen anderen Informationsquellen sein kann (und will!), die ich zum Teil in Abschnitt 5.1 auf Seite 25 aufgezählt habe. Grundlagenfragen („Kann ich auch eine ISDN- statt einer Netzwerkkarte verwenden?“, „Wie installiere ich dieses seltsame LINUX eigentlich unter Windows XP?“ etc.) scheiden völlig aus.

1.1.4 Voraussetzungen zum Befolgen dieser Anleitung

Voraussetzung dafür, dass Du Deinen eigenen Server aufzusetzen vermagst, ist, dass Du entweder Ahnung von UNIX/LINUX hast oder aber eine gesunde Kombination aus Neugierde, informationstechnischem Verständnis, etwas Fleiß sowie Selbstständigkeit mitbringst. Denn Du wirst sicher auf das eine oder andere Problem stoßen, das Du irgendwie wirst lösen müssen. Es wäre ziemlich uncool, bei kleinsten Problemchen immer andere fragen zu müssen (insbesondere mich). Du solltest übrigens nicht weiter lesen, falls Du die gerade genannten Voraussetzungen nicht erfüllst.

1.2 Überblick

Im Folgenden werde ich zunächst auf die Installation sowie Konfiguration der Kiste eingehen und abschließend deren Pflege abhandeln. Einfach hinstellen und fertig ginge zwar, wäre aber nicht besonders clever – sonst sind plötzlich all Deine E-Mails ins Nirwana eingegangen, wenn Deine Festplatte hopps geht, und das wäre doch ziemlich uncool. Oder? Also hör nicht auf halbem Wege auf, sondern bleib bis zum Schluss dabei.

1.3 Konventionen

Es ist immer praktisch, zu wissen, worüber gerade geredet (bzw. geschrieben) wird. Daher will ich mich mit Dir zunächst über ein paar Konventionen verständigen.

1.3.1 Beispielbenutzer

Ich werde im Folgenden einen Beispielbenutzer namens „OBI-WAN KENOBI“ verwenden. Dieser verwendet den Login-Namen „`obiwan`“.

Und nein, ich habe keinen Sprung in der Schüssel (zumindest keinen, auf den Du sicher aus der Verwendung dieses Namens schließen könntest). Ich wollte einfach einen Namen verwenden, den Du sofort als Platzhalter erkennst und im Kopf jeweils durch Deine Daten ersetzt. Und das war der erste, der mir ein- und gefiel ...

1.3.2 Beispieldomain

Außerdem werde ich in dieser Anleitung zwei unterschiedliche Domainnamen nutzen. Die eine Domain namens „`meins.de`“ sei die, die unser Benutzer Obi-Wan Kenobi *offiziell* besitzt, die er also bei einem Dienstleister beantragt hat. Damit ist dieser Domain eine offizielle IP-Adresse aus dem öffentlichen Adressraum zugeteilt.

Ich gedenke, auch einen internen Nameserver aufzusetzen. Dieser wird mit privaten IP-Adressen arbeiten und soll nicht von außen ansprechbar sein. Für mein lokales Netz verwende ich eine eigene Domain namens „`arda.lan`“.*

Wieso diese zusätzliche interne Domain? Ganz einfach: Um eine Überdeckung des Namensraums zu vermeiden. Überleg mal: Soll `www.meins.de` die öffentliche IP-Adresse oder die interne bezeichnen? Das musst Du entscheiden können. Und – Firewalling sei dank – wirst Du über die internen IP-Adressen wesentlich mehr in Deinem Netz dürfen als über die offiziellen.

Zusammenfassend: Besitzt Du eine offizielle Domain, setzt Du für „`meins.de`“ jeweils Deine offizielle Domain ein. Hast Du keine, verwendest Du durchgängig den internen Domainnamen „`arda.lan`“. Und falls Du doch durchgängig Deine offizielle Domain nutzen möchtest, bitte. Aber sag nicht, ich hätte Dich nicht gewarnt, falls Du Probleme mit der Namensauflösung bekommst …

1.3.3 Beispielnetzwerk

Damit Du weißt, wie mein lokales Netzwerk aussieht, das vor allem im Abschnitt zur Einrichtung eines Nameservers erwähnt wird, habe ich Dir ein Bildchen (vgl. Abb. 1.1 auf der nächsten Seite) gemalt, das die Topologie samt der Rechnernamen veranschaulicht.

1.3.4 Typographische Konventionen

In der Anleitung verwende ich einige typographische Konventionen, die Du so oder so ähnlich aus diversen Fachbüchern kennen dürftest. Entsprechende Beispiele findest Du in Tabelle 1.1 auf der nächsten Seite. Aber keine Sorge, schon aus dem Kontext heraus dürfte Dir jeweils klar sein, wovon ich gerade schreibe.

- Als KAPITÄLCHEN gesetzt werden neu eingeführte Namen
- *Kursiv gesetzt* werden
  - Namen von Software
  - Namen von Programmen
- *Schräggestellt* werden
  - neu eingeführte Begriffe

* Die Erklärung für das Zustandekommen diesen Namens findest Du im Abschnitt 7.3 auf Seite 40 ausgeführt.

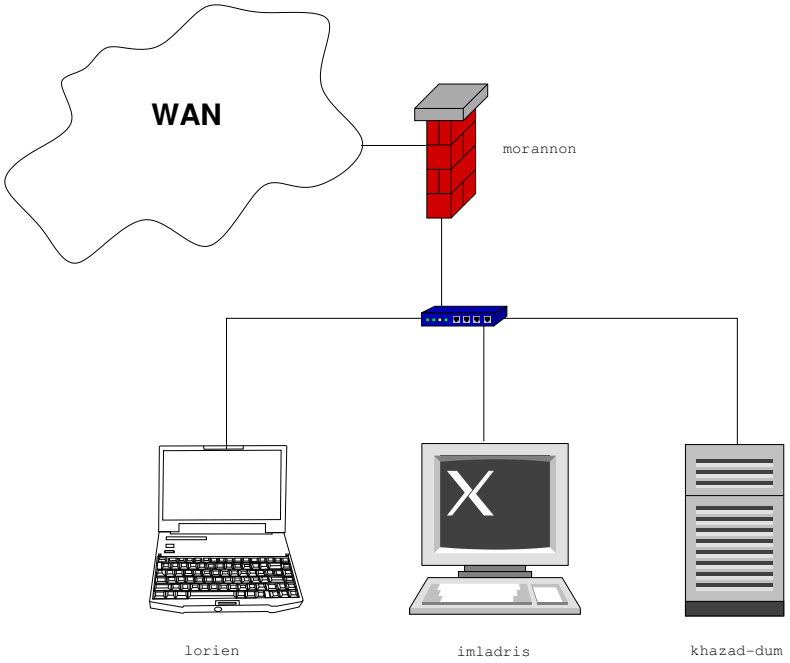

Abbildung 1.1: *mein lokales Netz*

Verwendung	**Darstellung**
neu eingeführte Namen	KONRAD ZUSE
Namen von Software	*Emacs*
Namen von Programmen	*emacs*
neu eingeführter Begriff	*Kernel*
Namen von Paketen	*emacs21-bin-common*
Benutzernamen	`obiwan`
Dateinamen	`/etc/samba/smb.conf`
Domainnamen	`arda.lan`
URLs	`http://www.de.debian.org/`
Eingaben (als `root`)	`coruscant:/tmp # dpkg -i foo.deb`
Eingaben (als User)	`obiwan@coruscant:~ $ ls -l /etc`

Tabelle 1.1: *typographische Konventionen*

  - neu eingeführte Namen
  - Namen von Paketen

- `Schrift mit fester Breite` wird verwendet für
  - Benutzernamen
  - Dateinamen
  - Domainnamen
  - URLs
  - Eingaben auf der Kommandozeile
  - Beispielsitzungen
  - Inhalte von Konfigurationsdateien

In meinen Beispielsitzungen, teilweise auch in den abgedruckten Konfigurationsdateien, wirst Du an manchem Zeilenende ein „\“ vorfinden. Bei diesem sog. *Backslash* handelt es sich um eine legale Syntax in der Shell sowie in vielen Konfigurationsdateien. In der Ausgabe eines Programms wird Dir das Symbol jedoch nie in dieser Rolle begegnen. Es bedeutet jeweils, dass ich aus satztechnischen Gründen *händisch* einen Zeilenumbruch einfügen musste. Stell Dir einfach vor, die nachfolgende Zeile stünde an der Stelle des „\“.

Genauer möchte ich kurz auf den Paketbegriff eingehen, auch wenn es Dir pedantisch erscheinen mag. Aber ohne die nun folgende Erläuterung wunderst Du Dich früher oder später darüber, dass beispielsweise die gesamte Dokumentation eines Pakets fehlt, auch wenn diese normalerweise mit der Software mitgeliefert wird und Du auf der zugehörigen Web-Site schon einiges Interessantes in dieser Dokumentation gelesen hast. Am besten erläutere ich Dir die Problematik an einem Beispiel:

Etliche Leute basteln an *Emacs*, einem freien Editor, der sich mittels der Programmiersprache LISP erweitern lässt und dadurch zu wesentlich mehr nütze ist als zum schnöden Bearbeiten von Texten aller Art. So gibt es z. B. Mail- und Newsreader, Spiele und vieles mehr für den *Emacs*. Bei einer Software wie dieser spreche ich von einem *Programmpaket*. Es enthält u. a. die Dokumentation, Beispiele etc.

Ein *Debian-Paket* ist dagegen eine installierbare Einheit, die die *Packager* des Debian-Teams aus dem eigentlichen Programmpaket zusammengestellt haben. Ein solches Debian-Paket kann ein komplettes Paket wie *Emacs* umfassen, tut dies aber zumindest bei größeren Programmpaketen typischerweise nicht. Stattdessen werden mehrere kleine Einheiten geschnürt, so dass man beispielsweise die Dokumentation weglassen kann, wenn man diese vielleicht aus Platzgründen nicht auf dem Rechner haben möchte. Im Fall von *Emacs* ist es auch tatsächlich so, dass das Programmpaket, so wie es seine Entwickler auffassen, durch die Debian-Pakete *emacs21-bin-common*, *emacs21-common*, *emacs21-el* und *emacsen-common* gebildet wird.

Ein Debian-Paket umfasst also u. U. nur *Teile* eines Programmpakets, so dass Du ggf. *mehrere* Debian-Pakete einspielen musst, um all das auf Deinem System zu haben, was normalerweise in einem *einzigen* Programmpaket enthalten ist.

Ach so: Wenn Dir das Ganze zu kompliziert klingt, kann man es auf eine einfache Formel herunterbrechen: Programmpakete haben einen Namen, der normalerweise Groß- und Kleinbuchstaben beinhaltet, während Namen von Debian-Pakete *ausschließlich* in Kleinbuchstaben gehalten sind.

1.4 Aktuelle Version dieser Anleitung

Auch wenn mir mein Server nicht als *Carrera*-Substitut dient: Solange ich meinen Server einsetze, nehme ich daran Veränderungen vor. Die neuen Erkenntnisse, die ich dabei gewinne, aber auch Verbesserungsvorschläge, die man mir zukommen lässt und die ich für interessant und sinnvoll halte, fließen in neue Versionen dieser Anleitung ein.

Damit Du weißt, *welche* Neuerungen sich hinter den jeweiligen Versionsnummern verbergen, pflege ich auf `http://www.kaaden.franken.de/papers/DebianServer/` ein Änderungsprotokoll. Dort findest Du neben dem Link auf die PDF-Version dieser Anleitung auch einen sog. *Tar-Ball* mit den langen Konfigurationsdateien, die ich im weiteren Verlauf abgedruckt habe. Die zahlreichen kleinen Schnipsel habe ich dabei weggelassen. Selber in Konfigurationsdateien gucken und darin etwas eintragen macht ja bekanntlich schlau.

1.5 Feedback

Ich sage es zwar nicht gern, aber nicht mal ich bin vollkommen. :-)

Vielleicht stolperst Du über den einen oder anderen Fehler, egal, ob fachlich oder sprachlich. Vielleicht hast Du auch Anregungen, die Deiner Meinung nach unbedingt in einer späteren Version dieser Anleitung berücksichtigt werden sollten. Oder Du bist einfach nur super mit meiner Anleitung klargekommen und dementsprechend glücklich.

In allen Fällen bin ich an Feedback von Dir interessiert. Die E-Mail-Adresse dafür lautet dsp@galileo.franken.de.

1.6 Über den Autor

MICHAEL KAADEN ist Diplom-Informatiker. Seit Abschluss des Studiums an der Universität Erlangen-Nürnberg ist er als Software-Ingenieur für ein mittelständisches Softwareunternehmen tätig. Dort hatte er neben seiner eigentlichen Tätigkeit jahrelang die Systemadministration unter seinen Fittichen. Derzeit beschäftigt er sich mit der Konzeption und Entwicklung von Teilen eines CASE-Tools. Aktuelle Schwerpunkte sind dabei Servlets und allgemein XML-Technologien.

LINUX setzt er seit 1993 ein. Damals war das noch eine *Softlanding System*-Linux-Distribution, die er u. a. zur Erstellung seiner Studienarbeit nutzte. Sein erstes SuSE Linux

hat er 1995 gekauft, das letzte 2004. Seitdem ist Debian GNU/Linux in den Versionen *Sarge* und *Sid*[*] in Verwendung.

Große Meriten verdient er sich im Freundes- und Bekanntenkreis sowohl durch die Beseitigung akuter Computerprobleme als auch durch Beratung in Hard- und Softwarefragen. Irgendwann, so hat er beschlossen, wird er dafür mal Geld verlangen und dadurch reich werden.

[*] zu den Codenamen siehe http://www.debian.de/releases/

KAPITEL 2

Systemvoraussetzungen

Du fragst, welche Hardware Du brauchst, um eine für Dich geeignete Kiste aufsetzen zu können. Du kannst im laufenden Betrieb auf jeden Fall auf Tastatur, Maus und Monitor verzichten. Zur Installation allerdings brauchst Du sie natürlich. Ansonsten kommt es darauf an, was Du mit dem Teil anfangen willst.

Brauchst Du eine Maschine, die nebenbei einen computeranimierten Film rendern soll, der PIXAR das Fürchten lehren soll, brauchst Du vor allem CPU-Leistung (am besten die von mehreren tausend Rechnern). Willst Du einen Medienserver, der alle Lieder all Deiner 2 000 CDs in MP3-Form vorhält und auch alle Hauptfilme Deiner Lieblings-DVDs, brauchst Du vor allem Plattenplatz und schnelle Netzwerkhardware.

Ich gebe Dir einfach mal ein paar konkrete Beispiele aus meinem Erfahrungsschatz:

- Für meinen Server unter SuSE Linux mit Kernel 2.4 hatte ich einen ausgedienten Intel Pentium II mit 350 MHz und 256 MB Arbeitsspeicher und einer einfachen Grafikkarte verwendet. Die Maschine hatte rund 120 GB Plattenplatz und verfügte über ein CD-ROM-Laufwerk. Es waren drei Netzwerkkarten eingebaut, die das interne Netz, das DSL-Modem und eine Demilitarized Zone versorgten. Letztere war für eine WLAN-Basisstation gedacht. Außerdem steckte ein ISDN-Adapter drin.

  Das Ding ächzte etwas, wenn eine Menge E-Mails auf Viren und Spam zu untersuchen waren, tat ansonsten aber prima.

- Aktuell verwende ich einen in „meiner“ Firma ausgemusterten und mir freundlicherweise überlassenen Intel Pentium III mit 600 MHz und 1 GB Arbeitsspeicher. Die defekte 10 GB-Festplatte habe ich durch eine Neue mit einer Kapazität von 200 GB ersetzt, das ebenfalls defekte CD-ROM- durch ein neues DVD-ROM-Laufwerk. Und da ich die Maschine nicht mehr als Router und Firewall verwende (dafür habe ich inzwischen eigene Hardware), reicht mir eine einzige Netzwerkkarte.

  Mit der Kiste bin ich äußerst zufrieden (bis auf den lärmenden CPU-Lüfter, den ich demnächst mal ersetzen sollte, was mir aber die Uptime versaute).

- Ich habe auch schon mal eine entsprechende Kiste mit einem Pentium 100 mit 9 GB Plattenkapazität und 48 MB RAM aufgesetzt.

> Auch das tat wunderbar. Es muss aber klar sein, dass eine solche Maschine enorm in die Knie geht, wenn viele E-Mails eingehen und alle nach Viren untersucht werden müssen. Die Routing- und Firewalling-Funktionalität dagegen lastet die Maschine bei weitem nicht aus.

Fazit: Nimm, was Du hast. Hauptspeicher ist sicher wichtiger als eine schnelle CPU, außer, Du brauchst ordentlich Rechenleistung auf Deinem Heimserver. Achte auch auf den Stromverbrauch. Ein aktueller Intel Pentium 4 braucht unter Volllast über 100 Watt, und da ist die Peripherie wie Festplatten etc. noch nicht eingerechnet. Der Aldi-PC von Weihnachten 2004 nimmt sich beispielsweise unbelastet 138, unter Volllast sogar satte 223 Watt.* Und das Ding läuft 24h am Tag, und das 365 Tage im Jahr. Ökologisch gesehen nicht unbedingt korrekt. Rechne Dir mal aus, was da auf Deiner Stromrechnung stehen wird. Ein alter Rechner ist vom Stromverbrauch her wesentlich genügsamer. Es lohnt sich im Normalfall also sicher nicht, extra einen brandneuen Rechner zu kaufen, nur um den als Server in den Keller zu verfrachten. Notfalls kannst Du immer noch aufrüsten – neu installieren musst Du nicht, allenfalls einen neuen Kernel vor der Aufrüstung einspielen, falls Du von einer Intel P<irgendwas>-CPU zu einem AMD Athlon wechseln solltest (zumindest solange, wie Du nicht von einer 32- auf eine 64-Bit-Plattform übergehst).

* siehe http://www.heise.de/newsticker/meldung/53318

KAPITEL 3

Auswahl einer Linux-Distribution

Dir stellt sich natürlich die Frage, welche LINUX-Distribution Du verwenden sollst. Tja, das ist nicht so leicht zu beantworten, denn es gibt sie wie Sand am Meer. Ich kann nur für mich sprechen, und da der Titel dieser Anleitung schon den Namen *Debian GNU/Linux* beinhaltet, weißt Du bereits, was ich einsetze. Warum ich mich dafür entschieden habe, werde ich Dir kurz erläutern.

Ich habe neun Jahre lang SuSE Linux verwendet, nicht zuletzt deshalb, weil ich einen Großteil der damaligen Mitarbeiter samt Chefs noch von der Uni her kannte und die Distribution einfach prima war. Daher hat sich bisherige Version dieser Anleitung mit der Installation von SuSE Linux befasst. Inzwischen bin ich es einfach endgültig leid, spätestens alle zwei Jahre[*] statt früher DM 40 heutzutage € 60 bzw. € 90 (mit Handbuch) für eine neue SuSE Linux-Version ausgeben zu müssen, die dann von Anfang an dicke Bugs enthält[†], für deren Behebung man gerne Einblick in die Supportdatenbank hätte, aber nicht bekommt, weil man keine Lust hat, die Zwangsregistrierung[‡] durchzuführen.

Nachdem mich schon länger auch einige andere Sachen an SuSE Linux gestört haben[§], habe ich mir dann einige andere Distributionen angesehen. Am besten gefielen mir Debian GNU/Linux[¶] und GENTOO[‖].

---

[*] Weil danach keine Fehlerbehebungen im Falle von sicherheitsrelevanten Fehlern mehr erfolgen.

[†] In Version 9.1 stürzte der Kernel ab, sobald man ein XFS-Dateisystem einbinden wollte.

[‡] Keine Ahnung, ob SuSE diese mit SuSE Linux 9.0 eingeführte Praxis inzwischen aufgegeben hat. Und ich gebe zu: Es interessiert mich auch nicht mehr, obwohl ich jetzt fast zehn Jahre treuer Kunde war, nicht nur privat, sondern auch im Unternehmenseinsatz. Aber da meine Bekannten aus der Studienzeit, die bei SuSE arbeiteten, zwischenzeitlich alle das Unternehmen verlassen haben, belastet mich das nicht weiter.

[§] Beispielsweise die Tatsache, dass dort keinerlei Reaktion auf meine Bug-Reports oder Verbesserungsvorschläge erfolgt und keine neuen Pakete geschnürt sowie an die Kunden geliefert werden, wenn eine neue Version einer Software erscheint. Und ja, ich habe verstanden, *warum* SuSE letzteres so macht. Aber gefallen muss es mir trotzdem nicht, oder?

[¶] http://www.debian.org

[‖] http://www.gentoo.org

Debian GNU/Linux kann in meinen Augen vor allem die folgenden großen Pluspunkte gegenüber SuSE Linux verbuchen:

- hohe Qualität der mitgelieferten Software
- Distribution kostenlos verfügbar
- genialer Paketmanager *APT*
- sehr aktive Entwicklergemeinschaft, die auf Fehlermeldungen schnell reagiert und sehr hilfsbereit ist
- drei parallel gepflegte Zweige, die man beliebig (!) mischen kann:
  - *Stable* als stabile Distribution (Codename bis 6. Juni 2005 *Woody*, seitdem *Sarge*)
  - *Unstable* als stets aktuelle Distribution (Codename *Sid*)
  - *Testing* als Mittelweg zwischen den beiden zuerst genannten (Codename bis 6. Juni 2005 *Sarge*, seitdem *Etch*)
- Upgrade-Pfade zwischen den Entwicklungszweigen immer vorhanden
- Verwendung eigener Kernel ist unproblematisch – es gibt sogar das Paket *kernel-package*, das die Erstellung eigener Kernel-Pakete besonders einfach macht

Ich erwähnte bereits Gentoo Linux. Warum habe ich das nicht genommen? Nun, Gentoos größte Stärke ist für mich gleichzeitig die größte Schwäche: Man kann keine Binärpakete herunterladen, sondern nur Quellpakete, die man dann auf seiner eigenen Maschine übersetzen lassen muss. Dadurch kann man über zentral hinterlegte Parameter sehr genau bestimmen, was man möchte. So kann man Optimierungen für die eigene CPU aktivieren oder aber Anwendungen, die man sowohl für *KDE* als auch *Gnome* übersetzen kann, wahlweise nur für *eine* dieser Oberflächen compilieren. Das ist eine tolle Sache, nur: Mein Server ist keine superschnelle Kiste. Das Ding übersetzt schon mal mehrere Tage an ein paar größeren Paketen, was derweil ordentlich Rechenzeit frisst und damit Performance kostet. Und das muss nun wirklich nicht sein, zumal es in meinem Fall den Vorteil der auf meine CPU optimierten Software *deutlich* aufwiegt. Abgesehen davon kann ich mir auch mal rasch ein auf meine Maschine optimiertes Debian-Paket schnüren, wenn ich das für eine Anwendung benötige.

Überhaupt, der sog. Paketmanager PORTAGE von Gentoo: Man kann problemlos Pakete deinstallieren, die von anderen Paketen vorausgesetzt werden:

> *Portage will not check if the package you want to remove is required by another package. It will however warn you when you want to remove an important package that breaks your system if you unmerge it.*[*]

[*] http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=2&chap=1#doc_chap3

Äußerst unschön. Debian GNU/Linux lässt so etwas nicht zu, außer, man erzwingt es explizit.

An Gentoo hat mich zudem irritiert, dass beim Update von Paketen ganz nebenbei Warnungen der Art *„Wenn Sie ein Update von globo-2.2 auf globo-2.4 durchführen, sollten Sie anschließend `globo-config --update-old-config` aufrufen.“* vorbeihuschen. Eine Aktualisierung des Systems erfolgt damit in zwei Schritten: Zuerst muss man die Programmpakete auf den neuesten Stand bringen und die Ausgaben in eine Protokolldatei bugsieren, etwa mittels

```
coruscant:~ # emerge --update world 2>&1 | tee -a /tmp/emerge.log
```

Anschließend muss man die Protokolldatei nach Meldungen durchsuchen, die einem mitteilen, dass man etwas händisch ausführen soll. Das finde ich nicht nett, denn Debian GNU/Linux mault bei so etwas typischerweise mittels eines Dialogs und erledigt auch verdammt viel selbst.

Entscheidend für mich war allerdings ein anderer Punkt: Ein Vergleich der vorhandenen Pakete hat ergeben, dass Debian GNU/Linux zumindest im LaTeX-Umfeld *wesentlich* mehr bietet als Gentoo. Das muss ich zwar nicht unbedingt auf meinem Server betreiben, aber auf meinem Desktop-System. Und ich bin ein fauler Hund – ich möchte nur eine LINUX-Distribution verwenden, nicht derer zwei. Und bei Gentoo müsste ich ja gleich doppelt compilieren … Nee, lass mal.

Dementsprechend habe ich pünktlich zum Jahreswechsel von 2004 auf 2005 die Umstellung meiner privaten Systeme von SuSE Linux auf Debian GNU/Linux vollzogen. Mein Server, der Gegenstand dieser Anleitung ist, läuft unter *Stable* (derzeit ist das „*Sarge*“), mein Desktop-System unter *Unstable*, Codename „*Sid*“. Den Wechsel habe ich bis jetzt nicht bereut, ganz im Gegenteil.

Teil II

Installation von Debian GNU/Linux

KAPITEL 4

Download von Debian GNU/Linux

Bevor Du daran gehen kannst, Debian GNU/Linux zu installieren, musst Du Dir erst einmal einen halbwegs aktuellen Stand davon besorgen. Dafür gibt es drei Möglichkeiten:

1. Du kaufst Dir einen Satz CDs bzw. DVDs von einem Händler*, besorgst sie Dir bei einer LINUX-Usergroup oder leihst sie Dir von einem Bekannten aus.

2. Du lädst Dir Debian GNU/Linux aus dem Internet herunter.†

3. Du ziehst Dir lediglich eine Grundinstallation aus dem Internet und lädst während der Installation die zusätzlich benötigten Pakete aus dem Netz nach.‡

Optimal, was die herunterzuladende Datenmenge angeht, ist sicher das zuletzt genannte Verfahren. Allerdings stehe ich da nicht so drauf – ich neige dazu, die Installation auch mal abzubrechen und neu aufzusetzen§, so dass der Download nochmals erfolgen müsste. Und da das Kosten verursacht (dank Flatrate zwar nicht für mich, aber für die Institutionen, auf deren Servern die Debian-Pakete liegen, sowie meinen Internet-Service-Provider, T-Online), ziehe ich eine Installation von ein paar optischen Datenträgern aus doch vor.

Ich habe mich demzufolge für die Download-Variante entschieden. Zum Download verwende ich, wie vom Debian-Team vorgeschlagen, das Programm *jigdo-lite*. Dieses lädt die erforderlichen Pakete von einem Debian-Archiv herunter und verschmilzt sie geeignet in einer Datei, die man dann als ISO-Image auf CD bzw. DVD brennen kann.

Gerade dann, wenn man *Testing* oder gar *Unstable* verwenden möchte (wovon für einen Server, der aus dem Internet zugreifbar sein oder in einer Produktionsumgebung verwendet werden soll, explizit abgeraten sei), bietet es sich an, CD-RW- bzw. DVD±RW-Medien zu verwenden. Denn die Images, die man auf CD bzw. DVD brennt, lassen sich mittels *jigdo-lite* hervorragend aktualisieren, indem automatisch nur die aktualisierten Pakete aus dem Internet heruntergeladen und mit den noch aktuellen zu einem neuen Image

* Eine Liste selbiger findet sich unter http://www.de.debian.org/CD/vendors/#de.
† Das Debian-Team beschreibt das Verfahren unter http://www.debian.org/CD/jigdo-cd/.
‡ Dieses Verfahren ist unter http://www.debian.org/distrib/netinst beschrieben.
§ Kinder finden immer den Ausschaltknopf oder den Resettaster an einem PC, immer!

verschmolzen werden. Dieses kann man dann wieder auf die RW-Medien brennen, was sowohl die Umwelt als auch den Geldbeutel freut.

Ach ja: Leider erst nachdem ich mir beide DVD-Images geholt hatte, habe ich erfahren, dass es prinzipiell ausreicht, lediglich die *erste* Installations-CD zur Hand zu haben. Auf der ist erstmal alles drauf, was Du für die Basisinstallation brauchst. Alles, was darüber hinausgeht, muss dann per Internet-Verbindung geholt werden. Da sich zumindest bei *Testing* und *Unstable* ohnehin ständig neue Pakete einfinden, ist das eine gute Sache. Und alle Pakete braucht man sowieso nicht.

jigdo-lite gibt unter dem URL `http://atterer.net/jigdo/#download` in Versionen für LINUX, Suns Solaris und Microsofts Windows zum Download. Portierungen existieren auch für Mac OS X (erhältlich über Fink*) und IRIX.

Unter LINUX entpackt man die heruntergeladene Archivdatei mittels

```
obiwan@coruscant:/tmp $ tar xvjf jigdo-bin-*
```

Andere UNIX-Systeme, die ohne die GNU-Version von *tar* auskommen müssen, erfordern zusätzlich den Einsatz von *bzip2*:

```
obiwan@coruscant:/tmp $ bzip2 -dc  jigdo-bin* | tar xvf -
```

Unter Windows XP reicht – wie üblich – ein Doppelklick zum Entpacken; bei älteren Windows-Versionen muss man ggf. auf die Dienste eines Entpackers wie *winzip* zurückgreifen.

In jedem Fall erhält man als Ergebnis der Entpackerei ein Verzeichnis, in dem sich u. a. eine Datei namens `jigdo-lite` findet. Diese Datei ist ausführbar.

Bevor Du nun *jigdo-lite* startest, musst Du Dir darüber klar werden, welchen Zweig von Debian GNU/Linux Du zum Aufbau Deines Servers verwenden möchtest. Da das derzeitige *Stable*-Release namens *Sarge* noch sehr aktuell ist, verwende ich dieses. Entscheide selbst, ob Du die Stabilität von *Stable* vorziehst oder die Features von *Testing* (oder gar *Unstable*).

Nachdem das geklärt wäre, rufst Du nun endlich *jigdo-lite* auf. Nachdem Du dessen Eingangsfragen beantwortet hast,† machst Du am besten was anderes Interessantes (der Fantasie sind keine Grenzen gesetzt). Es dauert nämlich eine ganze Weile, bis die Images zusammengestellt sind.

Im Folgenden findest Du eine Beispielsitzung zum Download der DVD-Version für die *x86*-Architektur.‡

Debian GNU/Linux-Image mittels *jigdo-lite* erstellen

```
1  obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $ ./jigdo-lite
2
3  Jigsaw Download "lite"
4  Copyright (C) 2001-2004  |  jigdo@
5  Richard Atterer          |  atterer.net
6
7  ---------------------------------------------------------------
8  To resume a half-finished download, enter name of .jigdo file.
9  To start a new download, enter URL of .jigdo file.
10 You can also enter several URLs/filenames, separated with spaces,
```

* `http://fink.sourceforge.net`

† Liste der offiziellen Images: `http://www.de.debian.org/CD/jigdo-cd/#which`

‡ Wundere Dich nicht: Der Download fand statt, als *Sarge* noch *Testing* war.

```
or enumerate in {}, e.g. 'http://server/cd-{1_NONUS,2,3}.jigdo'
jigdo: http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-{1,2}.jigdo

---------------------------------------------------------------------
You have asked me to process several files/URLs:
  http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-1.jigdo
  http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-2.jigdo

Entering batch mode

Downloading .jigdo file
--12:14:20--  http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-1.jigdo
           => 'sarge-i386-1.jigdo'
Auflösen des Hostnamen »cdimage.debian.org«.... fertig.
Verbindungsaufbau zu cdimage.debian.org[130.239.18.151]:80... verbunden.
HTTP Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 202,442 [text/plain]

100%[=====================================================>] 202,442       85.40K/s    ETA 00:00

12:14:23 (85.40 KB/s) - »sarge-i386-1.jigdo« gespeichert [202442/202442]


---------------------------------------------------------------------
Images offered by 'http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-1.jigdo':
  1: 'Debian GNU/Linux testing "Sarge" - Official Snapshot i386 Binary-1' (sarge-i386-1.iso)

---------------------------------------------------------------------
Batch mode: Will download 'sarge-i386-1.iso'

Further information about 'sarge-i386-1.iso':
Generated on Fri, 5 Nov 2004 20:11:52 -0700

---------------------------------------------------------------------
If you already have a previous version of the CD you are
downloading, jigdo can re-use files on the old CD that are also
present in the new image, and you do not need to download them
again. Mount the old CD ROM and enter the path it is mounted under
(e.g. '/mnt/cdrom').
Alternatively, just press enter if you want to start downloading
the remaining files.
Files to scan:

---------------------------------------------------------------------
The jigdo file refers to files stored on Debian mirrors. Please
choose a Debian mirror as follows: Either enter a complete URL
pointing to a mirror (in the form
'ftp://ftp.debian.org/debian/'), or enter any regular expression
for searching through the list of mirrors: Try a two-letter
country code such as 'de', or a country name like 'United
States', or a server name like 'sunsite'.
Debian mirror: ftp://ftp.de.debian.org/debian/

---------------------------------------------------------------------
The jigdo file also refers to the Non-US section of the Debian
archive. Please repeat the mirror selection for Non-US. Do not
simply copy the URL you entered above; this does not work because
the path on the servers differs!
Debian non-US mirror: ftp://ftp.de.debian.org/debian-non-US/

Downloading .template file
http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/jigdotemplates/sarge-i386-1.template
           => 'sarge-i386-1.template'
Auflösen des Hostnamen »cdimage.debian.org«.... fertig.
Verbindungsaufbau zu cdimage.debian.org[130.239.18.151]:80... verbunden.
HTTP Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 23,897,325 [text/plain]

100%[=====================================================>] 23,897,325    89.29K/s    ETA 00:00

12:19:03 (89.29 KB/s) - »sarge-i386-1.template« gespeichert [23897325/23897325]

general:        Image file:
general:        Jigdo:
general:        Template:
general:        [exit(0)]

---------------------------------------------------------------------
Merging parts from 'file:' URIs, if any...
general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template
general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template
general:        [exit(0)]
Found 0 of the 6920 files required by the template

ftp://ftp.de.debian.org/debian/pool/main/l/lg-meta/lg-all_103_all.deb
    => 'sarge-i386-1.iso.tmpdir/ftp.de.debian.org/debian/pool/main/l/lg-meta/lg-all_103_all.deb'
```

```
101  Auflösen des Hostnamen »ftp.de.debian.org«.... fertig.
102  Verbindungsaufbau zu ftp.de.debian.org[141.76.2.4]:21... verbunden.
103  Anmelden als anonymous ... Angemeldet!
104  ==> SYST ... fertig.    ==> PWD ... fertig.
105  ==> TYPE I ... fertig.  ==> CWD /debian/pool/main/l/lg-meta ... fertig.
106  ==> EPSV ... failed.  ==> PASV ... fertig.    ==> RETR lg-all_103_all.deb ... fertig.
107  Länge: 3,020 (unmaßgeblich)
108
109  100%[=====================================================>] 3,020         2.88M/s    ETA 00:00
110
111  »sarge-i386-1.iso.tmpdir/ftp.de.debian.org/debian/pool/main/l/lg-meta/lg-all_103_all.deb« gespeichert [3020]
112
113  ...
```

Abschließend solltest Du zwei ISO-Images im aktuellen Verzeichnis vorfinden:

```
obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $ ls -o
...
-rw-r--r--  1 obiwan  4688390144 2004-11-08 14:52 sarge-i386-1.iso
-rw-r--r--  1 obiwan  4428951552 2004-11-08 17:58 sarge-i386-2.iso
...
obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $
```

Diese kannst Du mit einem Brennprogramm Deiner Wahl auf optische Medium schreiben. Du kannst sie aber beispielsweise auch per Loop-Device mounten (siehe nächsten Abschnitt) und dann innerhalb Deines Netzwerks freigeben.

4.1 Aktualisierung vorhandener Images

Eine sehr nette Eigenschaft von *jigdo-lite* ist es, bereits vorhandene Images auf den aktuellen Stand bringen zu können. Dann werden nur die Pakete geholt, die sich im Vergleich zur vorhandenen Version geändert haben. Praktisch, das. Es gibt also keine Ausrede, einem Kumpel eine aktuelle Version von Debian GNU/Linux zu versagen.

Dazu macht man anfänglich dasselbe wie weiter oben beschrieben. Allerdings gibt man – in der obigen Beispielsitzung wäre das Zeile 52 – ein Verzeichnis an, in dem die Inhalte der bereits vorhandenen Medien liegen.*

Für das Verzeichnis hast Du mehrere Möglichkeiten: Du kannst der Reihe nach Deine optischen Datenträger mounten, deren Inhalte auf Deine Festplatte kopieren oder aber die ohnehin bereits vorhandenen ISO-Images per *Loop-Device* einhängen. Um letzteres zu realisieren, würdest Du als Benutzer `root` einen beliebigen *Mount Point* anlegen (so Du keinen geeigneten mehr in Reserve hast)

```
coruscant:/ # mkdir /tmp/mnt
```

und anschließend mittels

```
coruscant:/ # mount /pfad/zum/iso /tmp/mnt -o loop=/dev/loop1
```

* Hat man das einmal getan, so merkt sich *jigdo-lite* dies. Unter UNIX/LINUX ist dafür die Datei `~/.jigdo-lite` zuständig. Man kann den Pfad dann menügesteuert auswählen.

das Image einhängen. Jetzt kannst Du die Inhalte des ISO-Image unter `/tmp/mnt` genauso einsehen, als hättest Du es auf einen optischen Datenträger gebrannt und diesen eingebunden.

Auch für die Aktualisierung habe ich eine Beispielsitzung mitgeschnitten. Man beachte: Da man zwischendurch das Medium gegen ein Weiteres austauschen muss, wenn man die Inhalte mehrerer Medien aktualisieren möchte, kann man nicht alle auf einen Rutsch erledigen, sondern muss sequentiell vorgehen.

Debian GNU/Linux-Image mittels *jigdo-lite* aktualisieren

```
obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $ su -
Password:
coruscant:~ # mount /export2/data/Debian/sarge-i386-1.iso /tmp/mnt -o loop=/dev/loop1
coruscant:~ # exit
obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $ ./jigdo-lite

Jigsaw Download "lite"
Copyright (C) 2001-2004  |  jigdo@
Richard Atterer          |  atterer.net
Loading settings from ‘/home/obiwan/.jigdo-lite’

---------------------------------------------------------------
To resume a half-finished download, enter name of .jigdo file.
To start a new download, enter URL of .jigdo file.
You can also enter several URLs/filenames, separated with spaces,
or enumerate in {}, e.g. ‘http://server/cd-{1_NONUS,2,3}.jigdo’
jigdo [http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-1.jigdo]:

Downloading .jigdo file
http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-1.jigdo
           => ‘sarge-i386-1.jigdo’
Auflösen des Hostnamen »theshire.arda.lan«.... fertig.
Verbindungsaufbau zu theshire.arda.lan[192.168.1.254]:3128... verbunden.
Proxy Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 202,442 [text/plain]

100%[=============================================================>] 202,442       85.99K/s    ETA 00:00

»sarge-i386-1.jigdo« gespeichert [202442/202442]


---------------------------------------------------------------
Images offered by ‘http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/sarge-i386-1.jigdo’:
  1: ’Debian GNU/Linux testing "Sarge" - Official Snapshot i386 Binary-1’ (sarge-i386-1.iso)

Further information about ‘sarge-i386-1.iso’:
Generated on Fri, 5 Nov 2004 20:11:52 -0700

---------------------------------------------------------------
If you already have a previous version of the CD you are
downloading, jigdo can re-use files on the old CD that are also
present in the new image, and you do not need to download them
again. Mount the old CD ROM and enter the path it is mounted under
(e.g. ‘/mnt/cdrom’).
Alternatively, just press enter if you want to start downloading
the remaining files.

You can also enter a single digit from the list below to
select the respective entry for scanning:
  1: /tmp/mnt
Files to scan: 1

Downloading .template file
http://cdimage.debian.org/pub/cdimage-testing/dvd/jigdo-area/i386/jigdotemplates/sarge-i386-1.template
           => ‘sarge-i386-1.template’
Auflösen des Hostnamen »theshire.arda.lan«.... fertig.
Verbindungsaufbau zu theshire.arda.lan[192.168.1.254]:3128... verbunden.
Proxy Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 23,897,325 [text/plain]

100%[=============================================================>] 23,897,325    89.49K/s    ETA 00:00

»sarge-i386-1.template« gespeichert [23897325/23897325]

general:        Image file:
general:        Jigdo:
general:        Template:
general:        [exit(0)]
general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template
   scanning ‘tmp/mnt/dists/sarge/contrib/binary-i386/Packages’
   scanning ‘tmp/mnt/dists/sarge/contrib/binary-i386/Packages.gz’
```

```
    ...
Found 6147 of the 6920 files required by the template

  writing image  100%
  Copied input files to temporary file ‘sarge-i386-1.iso.tmp’ - repeat command and supply more files to continue
general:        [exit(1)]

-----------------------------------------------------------------
If you already have a previous version of the CD you are
downloading, jigdo can re-use files on the old CD that are also
present in the new image, and you do not need to download them
again. Mount the old CD ROM and enter the path it is mounted under
(e.g. ‘/mnt/cdrom’).
Alternatively, just press enter if you want to start downloading
the remaining files.

You can also enter a single digit from the list below to
select the respective entry for scanning:
  1: /tmp/mnt
Files to scan:

-----------------------------------------------------------------
The jigdo file refers to files stored on Debian mirrors. Please
choose a Debian mirror as follows: Either enter a complete URL
pointing to a mirror (in the form
‘ftp://ftp.debian.org/debian/’), or enter any regular expression
for searching through the list of mirrors: Try a two-letter
country code such as ‘de’, or a country name like ‘United
States’, or a server name like ‘sunsite’.
Debian mirror [ftp://ftp.de.debian.org/debian/]:

-----------------------------------------------------------------
Merging parts from ‘file:’ URIs, if any...
general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template
general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template
general:        [exit(0)]
Found 0 of the 773 files required by the template

ftp://ftp.de.debian.org/debian/pool/main/l/lg-meta/lg-all_103_all.deb
           => ‘sarge-i386-1.iso.tmpdir/ftp.de.debian.org/debian/pool/main/l/lg-meta/lg-all_103_all.deb’
Auflösen des Hostnamen »ftp.de.debian.org«.... fertig.
Verbindungsaufbau zu ftp.de.debian.org[141.76.2.4]:21... verbunden.
Anmelden als anonymous ... Angemeldet!
==> SYST ... fertig.    ==> PWD ... fertig.
==> TYPE I ... fertig.  ==> CWD /debian/pool/main/l/lg-meta ... fertig.
==> EPSV ... failed.  ==> PASV ... fertig.    ==> RETR lg-all_103_all.deb ... fertig.
Länge: 3,020 (unmaßgeblich)

100%[=====================================================>] 3,020          2.88M/s    ETA 00:00

»sarge-i386-1.iso.tmpdir/ftp.de.debian.org/debian/pool/main/l/lg-meta/lg-all_103_all.deb« gespeichert [3020]

...

BEENDET
Geholt: 1,785,182 Bytes in 3 Dateien
general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template

writing image 100%
Successfully created ‘sarge-i386-1.iso’
general:        [exit(0)]

-----------------------------------------------------------------
Finished!
The fact that you got this far is a strong indication that ‘sarge-i386-1.iso’
was generated correctly. I will perform an additional, final check,
which you can interrupt safely with Ctrl-C if you do not want to wait.

general:        Image file: sarge-i386-1.iso
general:        Jigdo:      sarge-i386-1.jigdo.unpacked
general:        Template:   sarge-i386-1.template
verifying image
OK: Checksums match, image is good!
general:        [exit(0)]
obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $ su -
Password:
coruscant:~ # umount /tmp/mnt
coruscant:~ # exit
obiwan@coruscant:/tmp/jigdo-bin-0.7.1 $
```

KAPITEL 5

Debian-Grundlagen

Bevor wir loslegen, möchte ich Dir im Folgenden einige wichtige Konzepte von Debian GNU/Linux vorstellen – Du wirst sie noch brauchen. Eine detaillierte Einführung kann ich Dir hier nicht geben – das ist auch nicht der Sinn dieser Anleitung. Das verstehst Du sicher.

5.1 Informationsquellen

Eine Unmenge nützlicher Informationen zu Debian GNU/Linux findest Du in der *Debian-Referenz*[*]. Das *APT-HowTo*[†] führt Dich prima in die Verwendung des Paketmanagers *APT* ein. Das Archiv aller Debian-Mailinglisten[‡] und eine FAQ-Sammlung[§] seien an dieser Stelle auch genannt. Außerdem empfehle ich, das *Security-HowTo*[¶] durchzuarbeiten.

Suchst Du Informationen zu einem bestimmten Paket, so findest Du die zugehörige Dokumentation auf Debian GNU/Linux immer unter `/usr/share/doc/<Paketname>`. Zusätzlich gibt es üblicherweise Manuals (sog. *Man-Pages*) zu den einzelnen Programmen innerhalb eines Pakets, die Du typischerweise mittels

```
obiwan@coruscant:~ $ man <programmname>
```

abrufen kannst.

Wenn Du nicht die *Stable*-Release von Debian GNU/Linux einsetzt, sondern *Testing* oder gar *Unstable*, so solltest Du zumindest die `debian-devel-announce`-Mailingliste verfolgen, um über Updates informiert zu sein.

[*] http://qref.sourceforge.net/Debian/reference/reference.de.html
[†] http://www.debian.org/doc/manuals/apt-howto/
[‡] http://lists.debian.org/
[§] http://www.de.debian.org/debian-user-german-FAQ/dug-faq.htm
[¶] http://www.debian.org/doc/manuals/securing-debian-howto/

5.2 Paketverwaltung

Eines vorweg, das Dir völlig fremd sein dürfte, falls Du aus der Windows-Ecke kommst: Unter LINUX kannst Du problemlos alle möglichen Pakete* installieren und wieder deinstallieren, ohne dass – wie unter Windows – „Reste“ zurückbleiben. Keine überflüssigen Registry-Einträge, keine DLLs, nix. Herrlich.

Pakete werden unter Debian GNU/Linux grundsätzlich von *dpkg* gehandhabt. Obwohl Du damit nur selten in Kontakt kommen wirst, habe ich einige wichtige Anwendungen davon für Dich in Abb. 5.1 zusammengefasst.

`dpkg -i <Paketname>` installiert das als Parameter angegebene Paket.

`dpkg -S <Dateiname>` sucht Dir die Pakete heraus, in denen der angegebene Dateiname (der auch ein kompletter Pfad sein darf) enthalten ist. Dabei wird aber nur in den *installierten* Paketen gesucht. Willst Du in *allen* Paketen suchen, benutze stattdessen *apt-file* mit folgendem Aufruf:

```
obiwan@coruscant:~ $ apt-file search <Dateiname>
```

`dpkg -l` listet Dir alle installierten Pakete auf. Zusätzlich kannst Du auch ein Muster als Parameter angeben. Da die Paketnamen evtl. gekürzt werden, damit die Information komplett in Deine Terminalbreite passt, solltest Du stets händisch eine hohe Spaltenanzahl angeben, z. B. (in *bash*-Syntax) mittels

```
obiwan@coruscant:~ $ COLUMNS=200 dpkg -l "*emacs*"
```

Abbildung 5.1: *wichtige `dpkg`-Anwendungen*

Als Front-End für *dpkg* wird normalerweise *APT*, das *Advanced Packaging Tool*, verwendet. *APT* hilft Dir u. a. dabei, Abhängigkeiten aufzulösen. Doch besser noch: *APT* vermag es auch, Abhängigkeiten selbständig aufzudröseln. Wenn Paket *A* Paket *B* benötigt, welches wiederum die Pakete *C* und *D* voraussetzt, installiert *APT* einfach alle vier auf einen Streich. Während man mit *APT* schon arbeitet, werden mit RPM noch händisch Abhängigkeiten aufgelöst ...

Auch *RPM*, der Paketmanager von RED HAT, SuSE und einigen anderen Distributionen, vermerkt je Paket, welche anderen es voraussetzt. *APT* arbeitet aber feingranularer: Neben der *depends*-Liste gibt es auch noch *recommends*- und *suggests*-Listen. Diese werden von *APT* normalerweise nicht berücksichtigt, sondern Dir nur mitgeteilt.

* Debian GNU/Linux unterscheidet zwischen Binär- und Quellpaketen. Ich meine stets Binärpakete, wenn ich von Paketen schreibe.

Wie man *APT* bedient, siehst Du in der Manual-Page zu *apt-get*, die Du per

```
obiwan@coruscant:~ $ man apt-get
```

einsehen kannst. Abbildung 5.2 auf der nächsten Seite zeigt Dir ein paar Anwendungsbeispiele. Ergänzt Du in diesen Beispielen noch die Option `-s` wie *simulate*, so wird nur angezeigt, was durchgeführt würde – es werden dabei keinerlei Veränderungen im Dateisystem vorgenommen, so dass Dir beim Ausprobieren kein „Unfall“ passieren kann.

Eine Alternative zu purem *apt-get* ist z. B. *aptitude*, das eine nette semigraphische Oberfläche besitzt und auch die als *„recommends“* und *„suggests“* gelisteten Abhängigkeiten berücksichtigt, falls Du das im Menüpunkt `Handling Dependencies`, den Du im per F10 erreichbaren Menü `Options` findest, so eingestellt hast.

Würde die Installation eines Pakets einen Konflikt mit einem anderen verursachen, wirst Du ausdrücklich gewarnt. Sollte eine von Dir geänderte Konfigurationsdatei in einer neueren Version vorliegen, wirst Du gefragt, wie Du das handhaben möchtest (ersetzen, vergleichen, Shell aufrufen etc.). Es wird *niemals* etwas überschrieben. Eine neue Welt tut sich für Dich auf, falls Du von der Windows-Seite mit ihrer DLL- und Registry-Hölle her kommst.

Solltest Du feststellen, dass nach der Installation eines neueren Pakets Probleme auftreten, die vorher nicht da waren (wie z. B. mit *sed* aus dem *Unstable*-Zweig in Version 4.1.2-2 geschehen), so solltest Du diesen Fehler mittels *reportbug* melden, so das nicht schon geschehen ist (dazu suchst Du kurz unter http://bugs.debian.org/). Außerdem hast Du dann gelernt, dass Du das Paket *apt-listbugs* installieren solltest, weil es Dich vor genau so etwas warnt (vorausgesetzt, jemand anderer vor Dir hatte auch schon Probleme mit dem Paket und hat dies auch in der Fehlerdatenbank vermerkt, wie es jeder gute Debianer machen sollte). Einstweilen kannst Du ältere Versionen des Pakets unter http://archive.debian.org/ (*Stable*) bzw. http://snapshot.debian.net/ (*Testing* und *Unstable*) finden und mittels des Kommandos

```
coruscant:/tmp # dpkg -i --force-downgrade sed-4.1.2-1-i386.deb
```

einspielen. Solange das Problem nicht behoben ist, setzt Du den Paketstatus auf *hold*, damit nicht wieder versucht wird, dieses „schlechte“ Paket einzuspielen, sobald Du wieder aktualisierst. Wie das geht? Zu Fuß, also mit *dpkg*, ganz einfach:

```
coruscant:~ # dpkg --get-selections | grep ^sed
sed             install
coruscant:~ # echo sed hold | dpkg --set-selections
coruscant:~ # dpkg --get-selections | grep ^sed
sed             hold
coruscant:~ # apt-get update; apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages have been kept back:
  sed
The following packages will be upgraded:
...
coruscant:~ #
```

`apt-cache search <Stichwort>` sucht in der Liste der verfügbaren Pakete.

`apt-cache showpkg <package>` liefert detaillierte Informationen zum angegebenen Paket. Sieh Dir auch `apt-cache show <package>` an.

`apt-get install <package>` installiert ein Paket.

`apt-get install -t unstable <package>` installiert ein Paket aus dem *Unstable*-Zweig und erfüllt dabei auch die Abhängigkeiten aus *Unstable*.

`apt-get remove <package>` entfernt ein Paket, lässt aber die Konfigurationsdateien übrig.

`apt-get --purge remove <package>` entfernt ein Paket samt den Konfigurationsdateien.

`apt-get update` bringt den Paketindex auf den aktuellen Stand. Die Quellen werden `/etc/apt/sources.list` entnommen.

`apt-get upgrade` bringt die installierten Pakete auf den aktuellen Stand, wobei weder zusätzliche Pakete hinzugefügt oder vorhandene entfernt werden. Option `-u` zeigt vor dem Download an, welche Pakete betroffen sind.

`apt-get dist-upgrade` macht dasselbe, löst dabei aber Konflikte auf, bringt also ggf. zusätzliche Pakete auf das System oder entfernt vorhandene. Auch hier ist die Option `-u` sehr hilfreich.

`apt-get clean` ist nützlich, um unter `/var/cache/apt/...` Platz zu schaffen; denn *apt-get* legt Pakete immer erst einmal in dem Verzeichnis ab, bevor sie installiert werden, lässt sie dann aber auch dort liegen, denn es könnte ja sein, dass Du das Paket mit

```
coruscant:~ # apt-get --reinstall <package>
```

erneut installieren möchtest – und dann geht das ohne weiteren Download.

Abbildung 5.2: *wichtige `apt-get`-Anwendungen*

Sobald das Problem im Paket behoben ist,* brauchst Du nur ein

```
coruscant:/tmp # echo sed install | dpkg --set-selections
```

einzugeben, um den *hold*-Status wieder zurückzusetzen. Du kannst das natürlich auch menügesteuert mittels *aptitude* machen.

Eine nette Eigenschaft von Debian GNU/Linux ist es, sog. *virtuelle Pakete* zu unterstützen. Wofür die gut sind? Ein Beispiel: Du brauchst auf Deinem System zwingend einen *Mail Transfer Agent*, also ein Programm, das E-Mail entgegennimmt und weiter transportiert. Es gibt aber eine ganze Menge davon: *exim4*, *postfix*, *sendmail* und weitere. Jedes einzelne davon stellt das virtuelle Paket *mail-transport-agent* bereit, so dass die Installation von *einem* dieser Pakete ausreicht, *alle* Abhängigkeiten anderer Pakete nach *mail-transport-agent* zu befriedigen. Gäbe es keine virtuellen Pakete, müsste man in jedem Paket, das einen MTA benötigt, alle existierenden Mail Transfer Agents aufzählen und mit einem logischen Oder verknüpfen.† Kommt jetzt ein *neuer* MTA heraus bzw. passt Du Dir einen bestehenden an Deine speziellen Bedürfnisse an, müsste dieser händisch in die Abhängigkeitsbeschreibung aller Pakete eingepflegt werden, die einen MTA benötigen. Unschön, nicht wahr? Eben. Da sind virtuelle Pakete schon *deutlich* netter, denn durch diese braucht dem Paket mit dem neuen bzw. von Dir angepassten MTA lediglich die Information `Provides: mail-transport-agent` hinzugefügt werden, und schon ist die Sache erledigt. Krass, oder?

Hast Du übrigens mehrere Pakete installiert, die dasselbe virtuelle Paket bereitstellen (z. B. bieten die Pakete *vim* und *emacs21* das virtuelle Paket *editor*), so kannst Du mittels *update-alternatives* wählen, welches davon Dein Standardprogramm sein soll. Damit wirkt sich das Ganze so aus (Zeilen geeignet umgebrochen):

```
obiwan@coruscant:~ $ which editor
/usr/bin/editor
obiwan@coruscant:~ $ ls -o /usr/bin/editor
lrwxrwxrwx  1 root 24 2004-07-01 10:05 /usr/bin/editor -> \
                                       /etc/alternatives/editor
obiwan@coruscant:~ $ ls -o /etc/alternatives/editor
lrwxrwxrwx  1 root 12 2004-11-17 12:45 /etc/alternatives/editor -> \
                                       /usr/bin/vim
obiwan@coruscant:~ $
```

Die vermeintliche Datei `/usr/bin/editor` ist also vielmehr ein Soft-Link auf `/etc/alternatives/editor`. Das ist seinerseits ein Soft-Link, der auf die gewünschte Alternative der Editoren, in dem Fall *Vim*, zeigt. Wie bei vielen großen Ideen gilt: Auf den ersten Blick mag es etwas verwirrend wirken, auf den zweiten dagegen genial.

Eines noch: Der Paketverwaltungsmechanismus funktioniert wirklich prima. Ich habe ein komplettes *Testing*-Desktop-System erfolgreich auf *Unstable* hochgezogen und mit

---

* Beobachte einfach den von Dir oder jemand anderem gesetzten Fehlereintrag zu dem Problem auf `http://bugs.debian.org/`, ob der Fehler als *closed* gekennzeichnet wurde.

† Der Sinn dieser Vorgehensweise ist, dem Paketverwaltungsprogramm einen Hinweis der Art "Du brauchst einen von den drei MTAs *exim4*, *postfix* oder *sendmail*" zu geben.

dem System gearbeitet. Weil ich neugierig war, ob das klappt, habe ich ein Downgrade zurück von *Unstable* auf *Testing* gewagt, und zwar folgendermaßen:

```
coruscant:~ # cat /etc/apt/preferences
Package: *
Pin: release a=testing
Pin-Priority: 1001
coruscant:~ # cat /etc/apt/apt.conf
APT::Default-Release "unstable";
coruscant:~ # apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be REMOVED:
   ...
The following packages will be DOWNGRADED:
   ...
coruscant:~ #
```

Ging fast problemlos – ein paar Abhängigkeiten gingen schief, doch das ließ sich flott beheben: Ein

```
coruscant:~ # dpkg -i --force-downgrade ...
```

mit den Namen der problematischen Pakete* als Parameter, gefolgt von einem erneuten

```
coruscant:~ # apt-get dist-upgrade
```

für den Downgrade der restlichen Pakete, reichte aus.

5.3 Methodik bei Problemen

Sollte ein Daemon, also beispielsweise der NTP-Server, nicht richtig tun, so findest Du oft hilfreiche Hinweise dazu im *Syslog* (oder *Systemprotokoll*), das Dateien unter `/var/log/...` führt. Oft reicht es aber aus, als Benutzer `root` die Zauberformel

```
coruscant:~ # apt-get update; apt-get -u dist-upgrade
```

zu tippen, weil es zwischenzeitlich möglicherweise ein Update gibt, das den Fehler behebt. Ansonsten empfiehlt sich ein Blick in die Debian-Fehlerdatenbank†, wo Du von Dir gefundene Fehler auch eintragen kannst.

Hat das noch keine Problemlösung ergeben, verbleibt neben den Debian-typischen Informationsquellen samt Mailinglisten und natürlich deren Archiv (vgl. Abschnitt 5.1 auf Seite 25) noch Dein bester Freund – Google‡.

* namentlich *Samba* und einiges von *KDE*

† http://www.debian.org/Bugs/

‡ http://www.google.de

5.4 Konfigurationsdateien und Versionsverwaltung derselben

Konfigurationsdateien, die je Benutzer unterschiedlich sind, liegen im jeweiligen `$HOME`-Verzeichnis. Allgemeine Konfigurationsdateien dagegen findest Du unter `/etc`, `/etc/<paketname>` oder `/etc/default`.

Apropos Konfigurationsdateien: Mit geringem Zusatzaufwand kannst Du mit der Konfiguration herumspielen, bis Du schwarz wirst, und trotzdem die letzte Version der Dateien oder beliebige definierte andere Versionen zuverlässig wiederherstellen. Am einfachsten machst Du das, indem Du das Paket *rcs* installierst. Das ist ein *Versionsverwaltungssystem*.

Bevor Du an einer Datei herumpfuschst, schaust Du nach, ob auf derselben Verzeichnisebene, in der die gewünschte Datei liegt, auch ein Verzeichnis `RCS` existiert. Wenn nicht, legst Du es mittels

```
coruscant:/etc # mkdir RCS
```

an.

Dann führst Du auf der Kommandozeile

```
coruscant:/etc # ci -l <Dateiname>
```

aus. Damit hast Du die aktuelle Version der Datei „eingecheckt“. Jetzt fummelst Du an der Datei herum. Bist Du mit Deinen Änderungen zufrieden, checkst Du sie sicherheitshalber auch gleich wieder ein. Wenn nicht, so holst Du Dir einfach die alte Version wieder:

```
coruscant:/etc # co -l <Dateiname>
```

Du kannst auch herausfinden, was sich zwischen einer beliebigen der eingecheckten Versionen und Deiner aktuellen Version unterscheidet:

```
coruscant:/etc # rcsdiff <Dateiname>
```

vergleicht die zuletzt eingecheckte,

```
coruscant:/etc # rcsdiff -r<Version> <Dateiname>
```

die angegebene `Version` mit Deiner Datei. Ein beherztes

```
coruscant:/etc # co -l<Version> <Dateiname>
```

holt Version `Version` wieder aus dem Versionsverwaltungssystem hervor und ersetzt Deine aktuelle Datei damit.

```
coruscant:/etc # man rcsintro
```

verrät Dir noch einiges mehr darüber. Sehr praktisch, das Zeug. Mach Gebrauch davon!

5.5 Start und Stopp von Daemonen

Eine der Standardvorgehensweisen ist es, einen Daemon zu stoppen bzw. zu starten. Man macht das gerne, um den Daemon dazu zu bringen, Konfigurationsdateien erneut einzulesen, weil man darin Änderungen durchgeführt hat.

Um einen Daemon zu stoppen bzw. zu starten, verwendest Du das zugehörige *init*-Skript. Für den *cupsd*, den CUPS-Daemon, ruft man dazu

```
coruscant:~ # /etc/init.d/cupsys stop
```

bzw.

```
coruscant:~ # /etc/init.d/cupsys start
```

auf. Willst Du zwischen den beiden Kommandos keine weiteren Schritte durchführen, bietet sich stattdessen das Kommando

```
coruscant:~ # /etc/init.d/cupsys restart
```

an.

Viele Daemonen unterstützen auch ein *Reload*, bei dem kein Stopp des Daemons, gefolgt von einem Start, erforderlich ist, sondern die Konfigurationsdatei(en) im laufenden Betrieb erneut eingelesen werden. In dem Fall verwendest Du logischerweise das Kommando

```
coruscant:~ # /etc/init.d/cupsys reload
```

Das korrekte Skript findet man normalerweise dadurch heraus, dass man einen Blick in `/etc/init.d` wirft. Und selbst in obskuren Fällen sollte ein

```
coruscant:~ # grep <daemonname> /etc/init.d/*
```

ausreichen, um das passende Init-Skript zu finden.

Möchtest Du erreichen, dass ein Daemon automatisch beim Systemstart gestartet und beim Herunterfahren wieder beendet wird, so verwendest Du das Programm *update-rc.d* dafür.

5.6 Setzen und Ändern der Locale

Die sog. *Locale* sind eines der Problemfelder schlechthin im nicht-US-amerikanischen Sprachraum. Du willst einen Umlaut eingeben und siehst nur ein seltsames Symbol? Oder Du siehst stattdessen zwei Zeichen? Kein Problem, lässt sich alles mit den richtigen Locale-Einstellungen beheben.

Locale dienen im Wesentlichen dazu, das System an die Gepflogenheiten in Deinem Sprachraum anzupassen. Wie sollen Datum und Zeit angezeigt werden, wie sollen Dateinamen etc. alphabetisch korrekt sortiert werden – und wie sollen die Zeichen auf Dein Terminal ausgegeben werden.

Unter Debian GNU/Linux kannst Du die vorhandenen Locale jederzeit mittels des Kommandos

```
coruscant:~ # dpkg-reconfigure locales
```

abändern. Ich lasse typischerweise die folgenden auf meinem System erzeugen:

- `de_DE`
- `de_DE@euro`
- `de_DE.UTF-8`
- `en_GB`
- `en_GB.ISO-8859-1`
- `en_GB.UTF-8`
- `en_US`
- `en_US.ISO-8859-1`
- `en_US.UTF-8`

Anschließend darf man sich ein Standard-Locale aussuchen. Wer tapfer ist, kann ruhig `de_DE.UTF-8` wählen. Allerdings sind dann auch alle mittels *vim* erzeugten Dateien in diesem Multibyte-Zeichensatz verfasst. Ist schlecht zum Datenaustausch mit älteren Systemen. Auch LaTeX steht da nicht so drauf (auch wenn man das Paket *inputenc* mittlerweile damit verwenden kann). Ich gehe in diesem Fall den Weg des geringsten Widerstands und ziehe es daher vor, das klassische `de_DE@euro` zu verwenden.

Sofern Du die gewünschten Locale generiert hast, kannst Du Deine persönlichen Locale jederzeit zur Laufzeit wechseln, indem Du die `LANG`-Shell-Variable umsetzt. In der *bash* geht das mittels

```
obiwan@coruscant:~ $ export LANG=de_DE@euro
```

Anzeigen kannst Du sie mit dem folgenden Kommando (wobei ich die Ausgabe für meinen Benutzer abgedruckt habe)

```
obiwan@coruscant:~ $ locale
LANG=de_DE@euro
LC_CTYPE="de_DE@euro"
LC_NUMERIC="de_DE@euro"
LC_TIME="de_DE@euro"
LC_COLLATE="de_DE@euro"
LC_MONETARY="de_DE@euro"
LC_MESSAGES="de_DE@euro"
LC_PAPER="de_DE@euro"
LC_NAME="de_DE@euro"
LC_ADDRESS="de_DE@euro"
LC_TELEPHONE="de_DE@euro"
LC_MEASUREMENT="de_DE@euro"
LC_IDENTIFICATION="de_DE@euro"
LC_ALL=
obiwan@coruscant:~ $
```

Das Kommando (hier wieder mit der Ausgabe für meinen Benutzer)

```
obiwan@coruscant:~ $ locale charmap
ISO-8859-15
obiwan@coruscant:~ $
```

schließlich zeigt Dir die gerade in Deinem Terminal verwendete Zeichencodierung an.

KAPITEL 6

Vorbereitung der Installation

In diesem Kapitel helfe ich Dir dabei, ein paar Überlegungen anzustellen. Ab dem Folgekapitel geht es rund.

6.1 Planung der Partitionierung

Das mit der Partitionierung ist so eine Sache. Für manche ist es sogar eine Glaubensfrage. Am besten wird es sein, ich zeige Dir erstmal, wie ich meine Platte partitioniert habe:

```
khazad-dum:~ # fdisk -l
Disk /dev/hda: 203.9 GB, 203928109056 bytes
255 heads, 63 sectors/track, 24792 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot  Start    End     Blocks   Id  System
/dev/hda1           1     62     497983+  83  Linux
/dev/hda2          63    184     979965   83  Linux
/dev/hda3         185  24792  197663760    f  W95 Ext'd (LBA)
/dev/hda5         185   2129   15623181   83  Linux
/dev/hda6        2130   4561   19535008+  83  Linux
/dev/hda7        4562   5777    9767488+  83  Linux
/dev/hda8        5778   6020    1951866   82  Linux swap
/dev/hda9        6021  24792  150786058+  83  Linux

khazad-dum:~ # mount | grep hda | sort
/dev/hda1 on /boot type ext3 (rw)
/dev/hda2 on / type xfs (rw)
/dev/hda5 on /usr type xfs (rw)
/dev/hda6 on /var type xfs (rw)
/dev/hda7 on /home type xfs (rw)
/dev/hda9 on /export type xfs (rw)

khazad-dum:~ #
```

Und jetzt sage ich Dir, warum ich das so und nicht anders gemacht habe.

Schon beim *Swap Space* scheiden sich die Geister. Wie Du siehst, bin ich einer der Anhänger der „Man nehme die doppelte Hauptspeichermenge als Swap Space her“-Theorie. Für mein Anwendungsprofil ist das vielleicht überdimensioniert, aber ich möchte mal mit *User Mode Linux* herumspielen, so dass der Swap Space möglicherweise doch gebraucht wird. Und bei 200 GB Kapazität stören mich die 2 GB nun wirklich nicht.

Leider unterstützt das BIOS meines Mainboards (es handelt sich um ein ASUS P3B-F) nicht die vollen 200 GB meiner Platte. Also lege ich mir eine kleine `/boot`-Partition am Anfang der Platte an, die im erkannten Bereich von unterhalb 8 GB liegt. Es würden wenige MB reichen, aber ich mache mal 512 MB. Rechnet man etwa 5 MB pro Kernel (ggf. inklusive Initial RAM Disk), kann ich damit 100 Kernelversionen dort ablegen. Das sollte ausreichen. `:-)`

`/var` muss deshalb relativ groß sein, weil dort zum einen unsere komplette elektronische Korrespondenz lagert, zum anderen, weil *apt* dort die Pakete ablegen möchte (genauer unter `/var/cache/apt/archives`). `/` sollte eine Mindestgröße von 512 MB nicht unterschreiten, weil dort `/tmp` liegt. Und einige Programme *lieben* es, dort erstmal Müll abzulegen. Für `/usr` würden auch 5 GB dicke reichen, allerdings stehe ich auf Reserven. Weiß ich, was ich alles noch einspielen möchte? `/home` sollte nicht zu klein gewählt werden, da dort die Benutzerdaten abgelegt werden. Der Rest wandert bei mir traditionell in das `/export`-Verzeichnis.

Wenn Du magst, kannst Du auch mit dem typischerweise in allen LINUX-Distributionen vorhandenen *Logical Volume Manager* (LVM) arbeiten. Die Idee, die dahinter steckt, ist die folgende: Man zieht eine Abstraktionsebene zwischen Festplatten und Dateisysteme ein. Festplatten bzw. Partitionen auf beliebigen (aber natürlich an den Rechner angeschlossenen) Festplatten (*Physical Volumes*) werden zu einem Pool zusammengefasst, den man als *Volume Group* bezeichnet. Aus diesem Pool kann man nun *Logical Volumes* bilden und auf diesen dann die Dateisysteme anlegen. Kaufst Du eine neue Platte, kannst Du diese dem Pool hinzufügen und dann Deine vorhandenen Logical Volumes vergrößern. Zusammen mit Dateisystemen, die Du bei Bedarf erweitern kannst, bist Du hier platzmäßig auf der sicheren Seite. Allerdings vergrößerst Du dadurch auch Deine Ausfallwahrscheinlichkeit, denn wenn eine Deiner Platten ausfällt, sind nicht nur die darauf liegenden Partitionen fort, sondern die kompletten Logical Volumes, die auch nur Teile ihrer Daten auf dieser Platte liegen hatten. Und genau das ist der Grund, weshalb ich nicht mit dem LVM arbeite. Hätte ich ein RAID-Array zu Hause stehen (Spenden anyone?), sähe das schon anders aus. LVM ist cool. Vergiss nur nicht die Datensicherung ...

6.2 Planung der Dateisysteme

Ein interessanter Punkt ist auch, welche Dateisysteme Du einsetzen solltest. Glaubt man den üblichen Performancemessungen (siehe [1]), machst Du weder mit *ReiserFS 4* noch

mit *XFS* viel falsch. Allerdings ist die CPU-Last, die ReiserFS verursacht, deutlich höher als die von XFS, so dass sich XFS zu meinem Standarddateisystem gemausert hat.*

Für die Partition, von der ich boote (auf meinem Server ist das `/dev/hda1`), verwende ich allerdings immer *ext3*. Man kann mit *GRUB*, dem unter LINUX heutzutage gebräuchlichsten Bootmanager, zwar auch von XFS-Partitionen booten, aber eben nur dann, wenn GRUB im Master Boot Record (MBR) installiert ist. Aber vielleicht will ich mal eine andere Platte einsetzen oder etwas anderes treiben? Daher die Wahl von ext3 für meine Boot-Partition.

6.3 Planung des Einsatzzwecks

Wenn wir schon am Planen sind, solltest Du Dir vorab ein paar Gedanken über den Einsatzzweck Deiner Kiste machen. Mein Aufgabenprofil für die Maschine ist das Folgende:

- Mailserver samt Spam-Filter und Virenschleuse
- Fileserver
- Proxyserver
- Printserver
- DNS-Server
- DHCP-Server

Als Firewall sollte das System der reinen Lehre nach nicht betrieben werden – einfach deshalb, weil die Gefahr, dass etwas nach außen freigegeben wird, das eigentlich nicht von außen sichtbar sein sollte, relativ groß ist. Ich hatte allerdings schon eine Maschine im Einsatz, die auch als Router und Firewall diente – und wenn man weiß, was man tut, passiert auch nichts. Nachdem dedizierte Router heutzutage jedoch auch nicht mehr viel kosten, sollte man es sich gut überlegen, ob man sich nicht doch lieber eines der kleinen Kästchen zulegt, die diese Aufgaben prima zu übernehmen vermögen.

Ich für meinen Teil habe mir inzwischen einen solchen Router angeschafft, der den Internet-Zugang für mich handhabt und über eine integrierte Firewall (mit *Stateful Inspection*) verfügt. Daher brauche ich auf meinem Server keine Firewall aufzusetzen und auch keinen *pppd* laufen zu haben, der für mich eine Internetverbindung auf- und abbaut.

Soll Dein Rechner auch die Aufgaben eines Routers und einer Firewall übernehmen oder vom Internet aus zugänglich sein, weil das Teil auch noch Shell-, Web- oder Gameserver (oder was weiß ich) spielen soll, wirst Du einerseits noch ein paar andere Pakete als ich installieren und Dich andererseits mit Firewalls vertraut machen und eine solche auf der Kiste etablieren müssen. Aber nur Mut, Du schaffst das. Da gibt es auch so nette Tools wie *firehol*, *fwbuilder*, *mason* oder *shorewall*, die Dich dabei unterstützen. Ich

* Außer bei SuSE Linux 9.1, weil da ja der Kernel beim Mounten absemmelte ...

habe allerdings nur mal einen Blick auf *fwbuilder* geworfen, und das sah sehr interessant aus. Außerdem sei noch mal dringend auf das *Security HowTo* verwiesen (siehe 5.1 auf Seite 25).

KAPITEL 7

Durchführung der Grundinstallation

Nach dem Vorgeplänkel in den vorangegangenen Kapiteln legen wir nun endlich los. Davor möchte ich Dir allerdings noch etwas mit auf den Weg geben, das Du Dir bitte gut merkst:

> *Ich kann (und will!) Dein Händchen nicht bei jeder Kleinigkeit halten. Ich gehe davon aus, dass Du während der Installation auf dem Bildschirm erscheinende Dialogboxen als solche erkennst und den Text darin zu entziffern und zu verstehen vermagst. Ich werde daher nicht zu jedem Pipifax-Schritt etwas sagen, wenn ich der festen Überzeugung bin, dass Du selbst in der Lage bist (oder zumindest sein solltest), die richtige Handlung zu vollziehen.*

Wenn Du bei einer Box mit dem Text „Drücken Sie „OK“, um die Installation abzuschließen“ und einem einzigen Button mit Beschriftung „OK“ fragen musst, was Du jetzt machen sollst, solltest Du diese Anleitung auch gleich wieder zur Seite legen und stattdessen die Teletubbies im Fernsehen anschauen …

7.1 Vorbereitung

Bevor Du auf Deinem Rechner mit der Installation beginnst, rate ich Dir *dringendst*, doppelt und dreifach zu überprüfen, dass Du eine Sicherung der evtl. bereits auf dem Rechner befindlichen Daten durchgeführt hast *und diese Sicherung auch vollständig und intakt ist*. Nicht, dass Dir da ein Missgeschick à la „Huch, jetzt habe ich versehentlich meine Windows-Datenpartition formatiert, die ich eigentlich behalten und ins interne Netz freigeben wollte!“ passiert …

Außerdem solltest Du wenigstens halbwegs wissen, welche Hardware in Deinem Rechner steckt. Die Hardwareerkennung ist gut, aber nicht allmächtig. Und es wäre ja ganz nett, wenn Du den zu Deinem Prozessor passenden Kernel auswählen könntest und wenn Du wüsstest, über welchen Chipsatz Deine Festplatten angesprochen werden müssen.

Aber halt – bevor Du jetzt Deinen Rechner aufschraubst, solltest Du einfach mal cool an die Sache rangehen. Denn sobald der *Debian-Installer* läuft, hast Du die Möglichkeit, die

wichtigsten Hardware-Bestandteile einzusehen. Das geht folgendermaßen: Nach Erscheinen der ersten Dialogbox des Debian-Installers (bei mir ist das die „Choose Language"-Box) drückst Du die Tastenkombination Alt–F2. Du landest dann auf der zweiten Konsole.* Konsequenterweise kommst Du mittels Alt–F1 wieder zum Installer zurück. Nach Betätigung einer beliebigen Taste kannst Du dort das Kommando

```
debian:~ # dmesg | more
```

eingeben. Du siehst seitenweise die Kernel-Ausgaben und damit die vorhandene Hardware. Das sollte normalerweise ausreichen, um alle die Hardware betreffenden Fragen des Debian-Installers erschöpfend beantworten zu können.

Wenn Du im Moment keine Ahnung hast, welche Netzwerk-, Grafik- oder Soundkarte Du in Deinem Rechner stecken hast, ist das unproblematisch. Diese Komponenten kannst Du auch nach der Grundinstallation noch problemlos in Betrieb nehmen. Normalerweise reicht ein

```
khazad-dum:~ # lspci
```

bzw.

```
khazad-dum:~ # lsusb
```

aus, um die Komponenten zuverlässig zu erkennen. Diese Programme findest Du in den Paketen *pciutils* bzw. *usbutils*.

7.2 Booten vom Installationsmedium

Jetzt wird es spannend, denn Du darfst von Deinem Installationsmedium booten. Bei mir ist das, wie bereits erwähnt, eine DVD±RW. Dazu muss ich im BIOS meines Rechners (meist durch Drücken der Tasten F2 oder Entf während der Einschaltmeldung erreichbar) die entsprechende Bootreihenfolge korrekt einstellen: Erst vom optischen Laufwerk, dann von der Festplatte. Notfalls kannst Du Dir mit einer Bootdiskette behelfen, die dann das Weitere anstößt.

Nach dem Start vom Installationsmedium musst Du am `boot:`-Prompt auswählen, welchen Kernel Du verwenden willst. Die per Druck auf die Taste F1 verfügbare Hilfe ist recht aussagekräftig. Ich verwende `linux26`, weil ich einen Kernel aus der 2.6-Reihe haben möchte und weiß, dass die `expert26`-Funktionalität nichts bringt, das ich unbedingt bräuchte, ich dann aber hier mehr Schritte beschreiben müsste. Hast Du eine USB-Tastatur, kann das aber möglicherweise ein Grund für die Experteninstallation sein.

7.3 Lauf des Debian-Installers

Nach dem Booten des Installationssystems beginnt der *Debian-Installer* seine Arbeit. Er stellt Dir dabei eine ganze Menge Fragen, u. a. zur Partitionierung, den zu verwendenden Dateisystemen etc. Ich denke, das packst Du nach all unseren Vorüberlegungen auch

* Hast Du eine USB-Tastatur, passiert an dieser Stelle möglicherweise nichts. Sobald Du später Deine Tastatur konfiguriert hast, tut das aber. Versprochen.

ohne meine Unterstützung. Übrigens ist es kein Zeichen von Schwäche, als Allererstes „deutsch“ als Sprache des Systems festzulegen.

In meinem *Sarge*-Snapshot vom 23. November 2004 läuft der Debian-Installer für die Installationsoption `linux26` wie folgt durch:

1. **Choose Language.** Hier geht es um die Sprache, in der der Installer laufen und später das System betrieben werden soll. Entscheide Dich für „deutsch“.

2. **Land oder Gebiet wählen.** Klar: „Deutschland“.

3. **Wählen Sie Ihre Tastaturbelegung aus.** Nimm „deutsch“.

4. **Hardwareerkennung.** Jetzt sucht der Debian-Installer nach optischen Laufwerken und dem darin befindlichen Debian-Medium.

5. **Netzwerk einrichten.** Jetzt wird es deutlich interessanter. Hier musst Du einen Rechner- und Domainnamen sowie eine IP-Adresse festlegen.

   a) **Auswahl des Rechnernamens**

   Das ist traditionell eines der großen Probleme bei der Installation – es ist schwierig, sich einen geeigneten Namen auszudenken.

   Zulässige Zeichen sind:

   - die Buchstaben `a-z`
   - die Ziffern `0-9`
   - der Bindestrich `-`

   Groß- und Kleinbuchstaben werden dabei nicht unterschieden.

   Ich verwende für meine Rechner Ortsnamen aus dem „Herrn der Ringe“. Für den Server bietet sich der Name `khazad-dum` an.

   b) **Auswahl des Domainnamens**

   Grundsätzliche Überlegungen zu diesem Thema habe ich bereits in 1.3.2 auf Seite 6 angestellt. Das solltest Du Dir zunächst zu Gemüte führen, falls Du das noch nicht getan hast.

   Welche *Top Level Domain* verwenden wir also? Man liest oft, man solle für das lokale Netz die TLD `.local` verwenden. Davon sei allerdings abgeraten, wenn Rechner eingesetzt werden sollen, die auf Mac OS X basieren. Denn die eingangs genannte Konvention beißt sich mit Apples *Rendezvous*-Technologie, die dazu dient, dass Geräte in einem lokalen Netz einander auch ohne einen zentralen DNS erkennen. Dann sollte man entweder auf `.local` verzichten und stattdessen eine andere Bezeichnung wählen* oder aber den Weg gehen, den Apple in der Knowledge Base im Artikel *„Mac OS X 10.3: How*

* Apple empfiehlt die Verwendung von `.home`, `.office` oder `.lan`; vgl. Artikel *„Mac OS X 10.2: About Multicast DNS“* `http://docs.info.apple.com/article.html?artnum=107174`

to Look Up ".local" Hostnames via Both Rendezvous and Standard DNS"[*] beschreibt. Um es kurz zu machen: Ich habe mich für die TLD `.lan` entschieden.

Fehlt noch die Subdomain. Da eines meiner Lieblingsbücher J. R. R. Tolkiens *„Der Herr der Ringe"*[†] ist, verwende ich für mein lokales Netz den Namen der Welt, auf der Mittelerde liegt. Zusammen mit der oben besprochenen Wahl der TLD gibt das zusammen die Domain `arda.lan`.

c) **Auswahl der IP-Adresse**

Dieser Schritt fällt weg, wenn Du schon einen DHCP-Server in Deinem Netz hast – ein Feature, das normalerweise schon jeder Baumarkt-Router bietet.

Ich habe DHCP auf meinem alten Server laufen, so dass die Angabe einer IP-Adresse für mich zunächst wegfällt. Später werde ich den DHCP-Server auf den Debian-Rechner umziehen[‡] und dementsprechend die Datei `/etc/network/interfaces` nachbearbeiten müssen. Wie das geht, beschreibe ich ausführlich in Kapitel 10.2 auf Seite 72.

Musst Du eine IP-Adresse angeben, so beachte, dass sie aus demselben Bereich sein muss wie die der Rechner, die Du bereits mit einer Netzwerkkonfiguration versehen hast. Hast Du keine, nimmst Du eine beliebige Adresse 192.168.*x*.*y* mit

$$0 \leq x \leq 255$$
$$1 \leq y \leq 254$$

In Worten ausgedrückt: *x* darf die Werte 0, 1, 2, ..., 255 annehmen, *y* dagegen lediglich 1, 2, 3, ..., 254. Die reservierte IP-Adresse 192.168.*x*.0 bezeichnet übrigens die Netzwerk-, 192.168.*x*.255 die Broadcast-Adresse.

6. **Festplatte partitionieren.** Die grundsätzlichen Überlegungen hierzu haben wir bereits angestellt. Da kannst Du Dich jetzt selbstständig durchhangeln. Aber wundere Dich nicht: Der Installer blendet Dir immer *alle* Standard-Mount-Points vor – auch die, die Du bereits verwendet hast. Aber vielleicht ist die von Dir verwendete Version des Debian-Installers an der Stelle bereits klüger. Falls nicht: Pass auf, dass Du keine Mount-Points doppelt vergibst.

7. **Einspielen des Grundsystems.** Das ist angenehm, denn Du hast jetzt Pause. Geh Tee oder Kaffee kochen, hol Dir Obst oder einen ungesünderen Snack. Döner macht schöner.

8. **GRUB-Bootloader auf der Festplatte installieren.** Hast Du noch andere Betriebssysteme auf Deinem Rechner, so musst Du jetzt aufpassen wie ein Schießhund,

[*] http://docs.info.apple.com/article.html?artnum=107800

[†] Allerdings nur im Original oder in der Übersetzung von MARGARET CARROUX, nicht jedoch der meiner Ansicht nach unsäglichen Übersetzung von WOLFGANG KREGE ...

[‡] Ich nutze nicht den DHCP-Server in meinem Router, weil mir der zu wenig kann ...

damit Du nichts kaputt machst. Auf meinem Server dagegen habe ich nur mein Debian GNU/Linux, so dass ich GRUB einfach in den *Master Boot Record* installieren kann.

9. **Das System startet jetzt neu von Festplatte.** Nimm die CD bzw. DVD aus dem optischen Laufwerk und bestätige die Anfrage, ob das System neu starten darf, mit „Ja“. Diese Gelegenheit nutzt Du, die ursprüngliche Bootreihenfolge wiederherzustellen und die Rechneruhr zu überprüfen. Da ich nur Debian GNU/Linux auf meinem Rechner habe, stelle ich die Uhr im BIOS auf *UTC* ein.

   UTC steht dabei für *Universal Time Coordinated*. Sie hat in ihrer Bedeutung die Greenwich Mean Time (GMT) abgelöst. In Deutschland sind wir UTC um eine Stunde voraus, in der Sommerzeit um zwei. Mathematisch ausgedrückt:

$$\begin{aligned} t_{Deutschland} &= t_{UTC} + 1 \qquad (Winterzeit) \\ t_{Deutschland} &= t_{UTC} + 2 \qquad (Sommerzeit) \end{aligned}$$

10. **Konfiguration des Debian-Grundsystems.** Wenn der Reboot geklappt hat, darfst Du nun Dein System einrichten.

11. **Zeitzone einrichten.** Am einfachsten ist es, die Hardware-Uhr mit UTC laufen zu lassen. Ansonsten wählst Du eben die lokale Zeit aus, wenn Du danach gefragt wirst. Wenn Du aus Deutschland kommst, dürfte Deine Zeitzone „Europe/Berlin“ sein.

12. **Root-Passwort eingeben.** Jetzt denkst Du Dir ein Passwort aus, das acht Zeichen lang sein und auch Sonderzeichen enthalten sollte, damit es schwer zu erraten ist. Auf Umlaute und andere Zeichen, die Du mit z. B. einer US-Tastatur schwer oder gar nicht erreichen könntest, würde ich verzichten – wäre sonst u. U. fatal, wenn mal Deine Locale verfummelt sein sollten.

13. **Einen normalen Benutzer anlegen.** Dieser Account besitzt keine `root`-Rechte. In unserem Beispiel verwenden wir als Benutzernamen „Obi-Wan Kenobi“ mit dem Login-Namen „`obiwan`“. Auch für diesen musst Du ein möglichst sicheres Passwort eingeben.

14. **Installationsmedien indexieren.** Jetzt werden Deine Installationsmedien indexiert. Diese bilden den Grundstock, um Dein System mit weiteren Paketen zu versehen. Lege – nach Aufforderung – sämtliche Medien, die Du Dir zuvor heruntergeladen hast, in Dein optisches Laufwerk ein. In meinem Fall sind das die beiden DVDs.

15. **Fehlende Pakete installieren.** Das System installiert jetzt selbsttätig noch ein paar fehlende Pakete. Du brauchst nichts weiter zu tun als nach Aufforderung die gewünschten Datenträger einzulegen.

16. **Debian-Software-Auswahl per *tasksel*.** Jetzt kannst Du Profile aussuchen, denen Dein System entsprechen soll. Wähle am Besten gar nichts aus, da Du sicher keinen unnötigen Ballast auf Deinem System wünschst. Du machst das nachher nach meiner Anleitung von Hand. Wenn Du dagegen ein paar Profile als Rundumschlag bevorzugst, dann bitte, zögere nicht …

17. **Weiter Pakete einspielen.** Jetzt spielt das System noch mal massig Pakete ein. Du lässt das geschehen (zumal Dir nichts anderes übrig bleibt).

18. ***exim4* konfigurieren.** Das holst Du später nach. Wähle einstweilen das Profil „nur lokale Mailzustellung“ aus. Gib Deinen vorhin angegebenen Benutzer als lokalen Mailempfänger an.

19. **Grundinstallation ist fertig.** So, das wäre geschafft. Solltest Du irgendwo Bockmist gebaut haben, so kannst Du den gesamten Vorgang jederzeit durch den Aufruf des Programms *base-config* wiederholen.

Teil III

Erweiterte Einrichtung des Systems

KAPITEL 8

Vorbereitungen

Bevor Du all die Pakete installieren und konfigurieren darfst, die Dein Server braucht, um die geforderten Leistungen zu erbringen, solltest Du die Schritte nachvollziehen, die ich in diesem Kapitel durchführe. Gerade der erste Folgeabschnitt ist natürlich essentiell für das weitere Vorgehen. :-)

8.1 Anmeldung am Server

Wenn Du es vor Deinem Server bequem findest: Prima, bleib genau da, wo Du bis gerade an der Grundinstallation gearbeitet hast, melde Dich als `root` an und lies im Abschnitt 8.2 auf der nächsten Seite weiter.

Alle anderen können einen angenehmeren Ort aufsuchen, vorausgesetzt, sie haben dort einen Rechner stehen, der netzwerktechnisch an den neuen Server angebunden ist. Es reicht nämlich völlig aus, wenn Du Dich per *ssh*[*] auf der Maschine einloggst. Gewöhn Dich schon mal daran, denn an der Console wirst Du normalerweise wenig Zeit verbringen. Zumindest ich mache das nicht – im Keller ist es immer so schattig ...

Interessanterweise ist der `root`-Zugang per *ssh* in Debian GNU/Linux standardmäßig freigeschaltet. Wenn Du das ändern möchtest, setzt Du in `/etc/ssh/sshd_config`

```
PermitRootLogin no
```

und startest den *SSH*-Daemon neu mittels

```
khazad-dum:~ # /etc/init.d/ssh restart
```

Für den Zugriff per *ssh* auf Deinen Server kannst Du Dich entweder als `obiwan` oder direkt als `root` anmelden (letzteres logischerweise nur dann, wenn Du die o. g. Veränderung an der Konfigurationsdatei des Secure Shell-Daemons *nicht* vorgenommen hast):

```
obiwan@coruscant:~ $ ssh obiwan@khazad-dum
```

[*] *OpenSSH* findest Du für so ziemlich alle UNIX-/LINUX-Varianten und damit auch für Mac OS X. Unter Windows rate ich zur Verwendung von *PuTTY*.

bzw.

```
obiwan@coruscant:~ $ ssh root@khazad-dum
```

Beides setzt voraus, dass Du bereits einen lauffähigen Nameserver in Deinem Netz hast, der den Namen Deines neuen Rechners kennt. Bei mir ist das so, da ich meinen alten Server durch den neuen ersetze. Ist das bei Dir nicht der Fall, verwendest Du statt des Servernamens einfach die IP-Adresse, die Du vorhin vergeben hast.*

Falls Du Dich als `obiwan` angemeldet hast, führst Du abschließend das Kommando

```
obiwan@khazad-dum:~ $ su -
```

aus, um Dich zum Superuser `root` aufzuschwingen.

Wie Du Dich auch immer auf der Kiste angemeldet hast: Jetzt bist Du als `root` auf Deinem neuen Server. Als solcher kannst Du nun mit der Installation fortfahren bzw. für Chaos und Verderben sorgen.

8.2 Anlegen weiterer Benutzer

Jetzt ist der optimale Zeitpunkt, weitere Benutzer anzulegen – etwa für die Beste aller Ehefrauen. Das geht einfach so (den Namen bitte ich geeignet auszutauschen, um Ärger mit der besseren Hälfte zu vermeiden):

```
khazad-dum:~ # adduser bunny
```

Fehlende Informationen werden interaktiv abgefragt, das `$HOME`-Verzeichnis angelegt und die Dateien aus `/etc/skel` dorthin kopiert. Soll das Anlegen des `$HOME`-Verzeichnisses nicht stattfinden, weil Du z. B. das Verzeichnis von einer alten Kiste rüberkopiert hast, so verwendest Du stattdessen

```
khazad-dum:~ # adduser --no-create-home bunny
```

Danach kannst Du mittels *vigr* noch die Gruppenrechte anpassen. Dazu zwei Tipps von mir:

1. `id obiwan` zeigt Dir an, welchen Gruppen der während der Grundinstallation eingerichtete User angehört. Das kannst Du als Beispiel für die weiteren Benutzer nutzen.

2. Wenn Du den eigenen Benutzer zur Gruppe `adm` hinzufügst, stößt Du auf eine sehr angenehme Eigenart von Debian GNU/Linux: Mitglieder der `adm`-Gruppe können u. a. viele Logdateien unterhalb von `/var/log` lesen.

* Das setzt voraus, dass der Rechner, an dem Du jetzt sitzt, eine IP-Adresse im selben Netzbereich hat wie Dein neuer Server.

8.3 APT-Ergänzungen

Jetzt installierst Du geschickterweise gleich ein paar hilfreiche Tools:

apt-file ermöglicht es Dir, Inhalte von Paketen einzusehen und zu durchsuchen.

Willst Du beispielsweise wissen, was alles in *vim* enthalten ist? Kein Problem, das Kommando

```
obiwan@coruscant:~ $ apt-file -F list vim
```

bringt sofortige Erleuchtung.*

Fragst Du Dich, aus welchem Paket die Datei `/usr/lib/libxvim.la` stammt, so beantwortet Dir

```
obiwan@coruscant:~ $ apt-file search /usr/lib/libxvim.la
```

diese Frage *stante pede*.

apt-listbugs ist ein sehr freundliches Tool. Wenn Du Pakete installierst bzw. updatest, befragt es vor der Installation die Debian-Fehlerdatenbank nach bereits gemeldeten Problemen, die die gerade einzuspielenden Pakete betreffen, und zeigt Dir diese Information an. Du kannst somit sehr bequem bereits vor der Installation beurteilen, ob Du durch ein Update Probleme befürchten musst.

Achtung: Es wird dazu jedes Mal eine Internet-Verbindung aufgebaut. Wenn Du noch mit dem Modem arbeitest, solltest Du Dir das vielleicht überlegen. Aber für einen *Unstable*-Nutzer ist das Paket lebensnotwendig.

apt-listchanges zeigt vor der Installation eines Pakets an, was sich im Vergleich zu der derzeit auf Deinem System installierten Version geändert hat.

apt-show-versions verrät Dir an, welche Version eines Pakets auf Deinem System liegt bzw. verfügbar ist. Außerdem siehst Du, aus welchem Entwicklungszweig ein Paket stammt.

apt-howto-de verschafft Dir einen schnellen Überblick über die Verwendung von APT – wesentlich umfassender als das, was ich Dir bisher verraten habe. Aber wie gesagt: Das entspräche sonst auch nicht dem Anspruch dieser Anleitung.

* Die Option `-F` sorgt dafür, dass kein automatisches Wildcard am Anfang und Ende von „`vim`“ eingefügt wird. Ohne diese Option würdest Du daher auch die Inhalte von *kvim*, *vimpart* etc. angezeigt bekommen.

8.4 Aktivierung des *bootlogd*

Wenn Du die Datei `/etc/default/bootlogd` in einem Editor öffnest und darin

```
BOOTLOGD_ENABLE=Yes
```

setzt, wird der komplette Startvorgang Deines Systems (genauer: all das, was auf `/dev/console` ausgegeben wird) in `/var/log/boot` protokolliert. Das ist sehr nützlich, wenn Du Deinen Server ohne angeschlossenen Monitor betreibst oder aber beim Bootvorgang auf Deinem Bildschirm interessant erscheinende Meldungen auftauchen, die Dir jedoch zu schnell vorüberflitzen, als dass Du sie entziffern könntest.

8.5 Aktivierung der *bash*-Command Completion

Um uns etwas Bequemlichkeit zu gönnen, wollen wir für die *bash*, die LINUX-Standard-Shell, noch die *Command Completion* aktivieren. Unter diesem Begriff versteht man, dass wir durch Betätigung der Tabulator-Taste eine Vervollständigung des gerade zu Tippen begonnenen Dateinamens erreichen. Allerdings ist da noch mehr möglich – deutlich mehr.* Beispielsweise kann man dafür sorgen, dass auf eine Command Completion, die in einem Begriff erfolgt, der nach einem `man` auftaucht, nur passende Handbuchseiten angezeigt werden. Nett, nicht wahr? Und so etwas ist bereits vorkonfiguriert – man braucht es nur zu aktivieren.

Bearbeite dazu die Datei `/etc/bash.bashrc` mit einem Editor und entferne die Kommentarsymbole in dem Bereich, der `/etc/bash_completion` reinzieht. Sobald Du eine neue *bash* ausführst, ist diese überaus nützliche Erweiterung aktiv.

8.6 Virtuelle Screens

Auf neuen Systemen ziehe ich es vor, mir als erstes das Paket *screen* einzuspielen:

```
khazad-dum:~ # apt-get -u install screen
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  screen
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/581kB of archives.
After unpacking 1004kB of additional disk space will be used.
Preconfiguring packages ...
Selecting previously deselected package screen.
(Reading database ... 22191 files and directories installed.)
Unpacking screen (from .../screen/screen_4.0.2-3_i386.deb) ...
```

* Wenn Du Dich einlesen möchtest: Suche in der Man-Page zu *bash* nach der Überschrift “Programmable Completion” und leg los.

```
Setting up screen (4.0.2-3) ...

khazad-dum:~ #
```

Screen arbeitet ähnlich einem Window-Manager unter dem X-Window-System: Wie X11 zur selben Zeit mehrere Fenster verwaltet, so handhabt *Screen* gleichzeitig mehrere Terminals, zwischen denen man bequem umschalten, kopieren etc. kann. Du brauchst auf Deinem Client also nicht zig Fenster mit Server-Shells gleichzeitig aufzureißen, nur weil Du mehrere Dinge zur selben Zeit im Auge behalten möchtest.

8.7 Installation eines vernünftigen Editors

Jeder hat seinen Lieblingseditor – auch Du, auch wenn Du Dich möglicherweise noch nicht für einen entschieden hast. Bei mir ist das für systemadministrative Aufgaben typischerweise der *Vim*.*

Dummerweise ist nach der Grundinstallation kein *Vim* installiert, sondern nur ein *Nvi*. Das ist für mich (und falls Du noch unschlüssig bist, auch Dich) natürlich unerträglich, so dass ich das schleunigst ändern will. Daher muss als nächstes *vim* eingespielt werden:

```
khazad-dum:~ # apt-get -u install vim vim-common vim-doc ctags
```

Das sorgt auch gleich dafür, dass *vim* als Default-*vi* gesetzt wird. Um das noch rasch zu überprüfen:

```
khazad-dum:~ # update-alternatives --display vi
vi - status is auto.
link currently points to /usr/bin/vim
/usr/bin/nvi - priority 30
slave vi.1.gz: /usr/share/man/man1/nvi.1.gz
/usr/bin/vim - priority 120
slave vi.1.gz: /usr/share/man/man1/vim.1.gz
Current 'best' version is /usr/bin/vim.
khazad-dum:~ #
```

Besser ist das.

8.8 Installation eines besseren Kernels

Wenn Du – so wie ich in dieser Anleitung – auf einer x86-Architektur installiert hast und nicht die Experten-Konfiguration angewählt hast, dürfte sich auf Deinem System jetzt der Default-Kernel befinden – einer, der für alle x86-CPUs ab i386 aufwärts gedacht ist. Das bedeutet, dass Verbesserungen im CPU-Befehlssatz, die in neuere Prozessoren wie

* Dieses Dokument habe ich übrigens mit *Emacs* geschrieben. Java-Code erstelle ich dagegen mit dem in *Eclipse* integrierten Editor. *Den* Editor gibt es für mich also nicht, nur immer einen Geeignetsten für eine Aufgabenklasse.

einem Pentium 4 eingeflossen sind, nicht genutzt werden und Du dadurch Performance verschenkst.

Welche CPU-Typ Du einsetzt, findest Du mittels

```
obiwan@khazad-dum:~ $ uname -m
i686
obiwan@khazad-dum:~ $
```

heraus, falls Du es nicht sowieso weißt. Wenn Du Genaueres wissen willst, etwa die genaue Modellbezeichnung, so verwende das Kommando

```
obiwan@khazad-dum:~ $ cat /proc/cpuinfo
```

Für meinen Pentium III installiere ich mittels

```
khazad-dum:~ # apt-get install kernel-image-2.6-686
```

den jeweils gerade aktuellen und für meine CPU optimalen Kernel der Serie 2.6.

Wer möchte, kann auf analoge Weise auch noch die Pakete *kernel-source-2.6.8* bzw. *kernel-doc-2.6.8* einspielen, um in den Genuss der zugehörigen Quelldateien bzw. Dokumentation zu kommen.

Danach wird ein Reboot fällig, um den neuen Kernel zu starten – eine der wenigen Gelegenheiten, die tatsächlich einen Neustart Deines LINUX-Servers erfordern.

Erlaube mir noch eine Anmerkung: Auf meinem Server haben die Debian-Kernels prima funktioniert. Auf meinem Desktop-System dagegen, das unter *Unstable* läuft, konnte ich den DMA-Modus für meine ATA-Laufwerke nicht aktivieren, weshalb ich dort gezwungen war, meinen eigenen Kernel zu bauen. Wie das geht, beschreibe ich in Kapitel 18 auf Seite 137.

KAPITEL 9

Inbetriebnahme, Optimierung und Überwachung der Hardware

In diesem Kapitel kümmern wir uns um die Hardware Deines Servers.

9.1 Temperaturüberwachung

Du willst garantiert, dass es nicht nur Dir und Deinen Lieben, sondern auch Deinen Rechnern gut geht. Dazu gehört die Temperaturüberwachung der CPU, des Mainbords sowie der Festplatte(n).

CPU- und Mainboardüberwachung

Daher installierst Du als nächstes das Paket *lm-sensors*. Das erledigst Du entweder mit

```
khazad-dum:~ # apt-get -u install lm-sensors sensord
```

oder mittels eines menügesteuerten Programms wie *aptitude*. Ich schlage letzteres vor, damit auch die vorgeschlagenen Pakete installiert werden.

Ein kurzer Blick in die Datei `/usr/share/doc/lm-sensors/README.Debian` klärt Dich darüber auf, dass Du nun als Benutzer `root`

```
khazad-dum:~ # sensors-detect
```

aufrufen sollst. Das machst Du einfach mal. Nachdem Du ein paar Fragen mit einem tapferen „nur zu!“ beantwortet hast (bei mir tat das prima), spuckt Dir das Programm aus, dass Du ein paar Eintragungen in `/etc/modules` vornehmen sollst. Diese Tätigkeit kannst Du auch vertrauensvoll dem Programm anvertrauen. Und wenn Du diese Module einzeln von Hand mittels *modprobe* nachlädst, kannst Du auch gleich *sensors* aufrufen und Dich (hoffentlich) an der Ausgabe ergötzen. Ach so: Die Module werden ab dem nächsten Systemstart automatisch nachgeladen.

Der *sensord* protokolliert per Default in der Datei `/var/log/daemon.log` Änderungen in den Sensordaten. Wir werden die Auswertung dieser Datei später noch angehen. Vorerst startest Du den Daemon mittels

```
khazad-dum:~ # /etc/init.d/sensord start
```

von Hand. Beim nächsten Systemstart geschieht auch das automatisch.

Festplattenüberwachung

Um sicher zu gehen, dass es auch Deiner Festplatte nicht zu warm wird, installierst Du als nächstes *hddtemp*:

```
khazad-dum:~ # apt-get -u install hddtemp
```

Der Konfigurationsdialog klärt Dich über die Möglichkeiten, aber auch über die Gefahren dessen auf, das Programm auch für normale Benutzer mit den Rechten von `root` ausführbar zu machen. Du entscheidest Dich sicherheitshalber dagegen. Wie der Dialog auch so schön sagt: Wenn Du es Dir anders überlegen solltest, kannst Du jederzeit

```
khazad-dum:~ # dpkg-reconfigure hddtemp
```

eingeben, um dies zu ändern. Auch auf den Einsatz des zugehörigen Daemons verzichtest Du. Du hast nämlich standardmäßig nichts laufen, das diesen periodisch abfragt. Und auch diese Entscheidung kannst Du per *dpkg-reconfigure* jederzeit ändern.

Jetzt kannst Du per

```
khazad-dum:~ # hddtemp /dev/hda
```

jederzeit nachsehen, wie warm es Deiner Platte gerade ist. Und zu meiner Freude sehe ich, dass meine gerade 30 °C hat, was deutlich im grünen Bereich liegt.

9.2 Überwachung des SMART-Status

Ich weiß ja nicht, wie Du das siehst – aber für mich ist es von enormer Wichtigkeit, die Festplatte meines Rechners im Auge zu behalten. Denn wenn die CPU abraucht, ist das bestenfalls lästig. Wenn jedoch die Platte stirbt, sind auch die Daten auf ihr fort. Das ist eine äußerst üble Angelegenheit. Dagegen helfen zwei Maßnahmen, die Du stets *beide* konsequent durchführen solltest:

1. regelmäßige Durchführung von Datensicherungen
2. permanente Überwachung des Festplattenstatus

Die Geschichte mit den Datensicherungen heben wir uns für Kapitel 17 auf Seite 131 auf. Im späteren Betrieb ist das die wichtigere Maßnahme,* aber da Du noch keine Daten

* Denn gegen ein versehentliches `rm -rf /` hilft nur ein aktuelles Backup, jedoch keine Überwachung des Festplattenstatus ...

auf der Kiste hast, brauchst Du Dir Deinen Kopf zumindest im Moment noch nicht darüber zu zerbrechen. Die Überwachung des Festplattenstatus dagegen wird unser Thema für diesen Abschnitt abgeben.

Moderne ATA- und SCSI-3-Festplatten unterstützen die „Self-Monitoring, Analysis and Reporting Technology“, kurz *SMART*. Diese ermöglicht es uns nicht nur, die Temperatur der Festplatte(n) zu überwachen, sondern auch eine Vielzahl weiterer Parameter abzufragen. Der Witz dabei ist: Die Festplatte notiert Fehler zwar in ihrem internen Speicher, aber sie meldet sie *nie* von sich aus. Sie leidet im schlimmsten Fall also still vor sich hin.

Du brauchst daher immer ein Programm, das diese Daten (in der SMART-Terminologie *Attribute* genannt) ausliest. Zu diesem Zweck installierst Du das Paket *smartmontools*. Darin befinden sich u. a. *smartctl* zur Abfrage der Attribute mittels der Kommandozeile sowie *smartd*, ein Daemon, der selbiges leistet, aber eben periodisch die Attribute abfragt und bei Verschlechterung der Werte warnt. Grundsätzliches Verhalten des im Paket enthaltenen Software stellst Du über die Konfigurationsdatei `/etc/default/smartmontools` ein.

Wenn Du wissen willst, ob Deine Festplatte von den *smartmontools* unterstützt wird, rufst Du *smartctl* einfach mit der Option `-i` und dem Device-Namen Deiner Festplatte(n) auf. Bei mir sieht das so aus:

```
khazad-dum:~ # smartctl -i /dev/hda
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen
Home page is http://smartmontools.sourceforge.net/

=== START OF INFORMATION SECTION ===
Device Model:     Maxtor 6Y200P0
Serial Number:    [geheim]
Firmware Version: YAR41BW0
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   7
ATA Standard is:  ATA/ATAPI-7 T13 1532D revision 0
Local Time is:    Sun Nov 28 13:57:44 2004 CET
SMART support is: Available - device has SMART capability.
SMART support is: Enabled

khazad-dum:~ #
```

Wie Du an der Ausgabe erkennen kannst, wird meine Platte bestens unterstützt. Ist das bei Dir auch so, so änderst Du Deine Konfigurationsdatei wie folgt ab (die Device-Namen Deiner Platten musste halt geeignet eintragen):

/etc/default/smartmontools

```
1  # Defaults for smartmontools initscript (/etc/init.d/smartmontools)
2  # This is a POSIX shell fragment
3
4  # list of devices you want to explicitly enable S.M.A.R.T. for
```

```
5  # not needed if the device is monitored by smartd
6  enable_smart="/dev/hda"
7
8  # uncomment to start smartd on system startup
9  start_smartd=yes
10
11 # uncomment to pass additional options to smartd on startup
12 #smartd_opts="--interval=1800"
```

Damit läuft ab dem nächsten Systemstart der *smartd*. Die zugehörige Konfigurationsdatei findest Du unter `/etc/smartd.conf`, deren Dokumentation Du mittels

```
obiwan@khazad-dum:~ $ man 5 smartd.conf
```

einsehen kannst (und solltest). Wenn Du die Konfigurationsdatei in Deinem Lieblingseditor ansiehst, findest Du ziemlich am Anfang der Datei auch den Vorschlag, `DEVICESCAN` zu deaktivieren und dafür die Zeile

```
/dev/hda -a -o on -S on -s (S/../../2/02|L/../../6/03) \
-m obiwan@arda.lan
```

zu ergänzen. Dadurch wird regelmäßig ein Selbsttest ausgeführt und Dir eine E-Mail geschickt, falls Deine Platte muckt.

Willst Du den *smartd* zunächst händisch starten, so verwende dazu – analog zu den anderen Daemonen – den Aufruf

```
khazad-dum:~ # /etc/init.d/smartmontools start
```

Änderungen in den SMART-Stati werden auch in `/var/log/daemon.log` protokolliert. Ich zeige Dir in Kapitel 16.1 auf Seite 127, wie Du dafür sorgst, dass in den Logdateien periodisch nach Problemen gesehen wird und Du im Fall des Falles per E-Mail verständigt wirst.

Spiel ruhig noch etwas mit *smartctl* herum. Viele weitere Informationen findest Du natürlich in der Man-Page dazu. Die interessantesten Anwendungen dürften die folgenden sein:

- `smartctl -s on /dev/hda` schaltet das SMART-Monitoring ein, falls es ausgeschaltet ist.
- `smartctl -H /dev/hda` liefert den Gesundheitsstatus der Platte.
- `smartctl -a /dev/hda` gibt Dir alles aus, was das Programm erfragen kann. Interessant sind dabei besonders die mit `Pre-fail` gekennzeichneten Attribute.

  Die mit `Old_age` gekennzeichneten Attribute zeichnen im Wesentlichen statische Daten auf: Attribut 194 gibt bei mir die aktuelle Temperatur an, 9 dagegen die Laufzeit der Platte in Minuten.

  Die nicht-korrigierbaren Fehler sind bei mir 0, die `Soft_Read_Error_Rate` ebenfalls. Ein gutes Zeichen.

- `smartctl -t offline /dev/hda` führt die Offline-Tests sofort durch. Normalerweise laufen diese nur alle paar Stunden.

Eine wichtige Bemerkung noch zum Schluss:

Erwarte keine Wunder.

Mir ist eine Platte abgeraucht, obwohl der SMART-Status hervorragend war. Ich vermute, ein Chip hat den Geist aufgegeben. Dagegen ist man nicht gefeit, ebenso wenig wie gegen Wasser im Keller, das auch den Server überschwemmt. Denk immer daran, Backups anzufertigen und sie so aufzubewahren, dass eine Katastrophe immer nur entweder die Backups oder den Server zerstört. Server im Keller, Backup in einem oberen Stockwerk ist also dafür eine Lösung. Wenn es dagegen brennt, hast Du wieder verloren. Solltest Du also wichtige Daten haben, kannst Du die periodisch bei einem Freund deponieren, um etwas sicherer zu sein. Allerdings weißt Du nicht, was Dein Kumpel dann mit den Daten macht, weshalb Du die Backups wieder verschlüsseln müsstest. Liegen die Backups allerdings bei Dir und Deinem Kumpel auf Festplatte (ihr synchronisiert täglich per *rsync*) und in euerer Gegend gibt es ein Erdbeben, sind wieder alle Daten verloren. Du siehst also, nichts ist so kompliziert, dass man es nicht noch komplizierter machen könnte …

9.3 Optimierung des Festplattenzugriffs

Wo wir schon gerade bei den Festplatten sind, machen wir da auch gleich weiter, wenn wir lediglich ATA-Festplatten (auch IDE genannt) besitzen. SCSI-Platten-Eigner sind fein raus, denn die brauchen keinerlei Voodoo mehr zu treiben, um ihre Platten dazu zu bewegen, flott zu arbeiten.

Installiere als nächstes *hdparm*. Damit kannst Du Festplattenparameter modifizieren, beispielsweise *DMA* aktivieren oder *Acoustic Management* betreiben, um die Arbeitsgeräusche Deiner Platten zu verringern.

Rumspielen mit *hdparm* ist potentiell *gefährlich*! Deshalb gebe ich Dir hier nur einen Tipp – für alles andere musst Du entweder die Man-Page durchschmökern (empfohlenes Verfahren) oder eine Suchmaschine Deiner Wahl bemühen.

Sieh nach, ob bei

```
khazad-dum:~ # hdparm -d /dev/hda
```

die Meldung „`using_dma = 1 (on)`“ erscheint. Bei mir ist das der Fall. Wenn nicht, kannst Du in den allermeisten Fällen ein

```
khazad-dum:~ # hdparm -d 1 /dev/hda
```

machen, um DMA zu aktivieren. Ansonsten wird nämlich der langsame *PIO*-Modus für den Plattenzugriff verwendet (schnarch).

9.4 Inbetriebnahme der ISDN-Karte

Leider werde ich Dir in diesem Abschnitt nur dann eine präzise Anleitung geben können, wenn Du – wie ich – eine ISDN-Karte des Herstellers AVM aus Berlin nutzt. Für andere ISDN-Karten mag es eine vergleichbare Methodik geben, aber da musst Du Dich dann schon selbst durchwursteln.

Ich beschreibe im Folgenden, was Du tun musst, um mit einer FRITZ!CARD PCI und darüber hinaus eine Menge anderer Ereignisse auf dem S_0-Bus protokollieren, Anrufe entgegennehmen sowie Faxe empfangen und senden zu können. Solltest Du diese oder eine andere aktive oder passive* AVM-ISDN-Karte besitzen, kannst Du meine Schritte vermutlich analog nachvollziehen.

Wenn Du eine andere passive Karte besitzt, die einen Siemens-Chipsatz enthält, ist die Chance groß, dass wenigstens *ISDN4Linux*† damit tut. Ob *CAPI4Linux*‡ auch damit kooperiert, kann ich Dir nicht sagen. Für meine Fritz!Card PCI gilt jedenfalls glücklicherweise beides – wobei das mit Glück zugegebenermaßen wenig zu tun hat, denn ich habe bei der Komponentenauswahl darauf geachtet, eine ISDN-Karte mit guter LINUX-Unterstützung zu erwerben. Man könnte also auch sagen, dass das Glück auf der Seite derer steht, die sich gut vorbereiten …

Damit die beiden Projekte *ISDN4Linux* und *CAPI4Linux* nicht einfach so im Raum stehen bleiben, will ich Dir eine eine kurze Erläuterung dieser beiden Projekte liefern und Dir insbesondere verraten, welchen Nutzen Du aus beiden ziehen kannst, bevor Du Deine ISDN-Karte in Betrieb nimmst.

Das ältere der beiden Projekte ist *ISDN4Linux*. Dieses gibt es bereits seit Mitte der 90er Jahre, um LINUX (übrigens sehr erfolgreich) die ISDN-Unterstützung beizubringen. Wenn bei Dir eines der Kernel-Module `isdn` oder `hisax` geladen ist (per `lsmod` leicht herauszufinden), dann nutzt Du bereits *ISDN4Linux*. Auf diese Module und die von ihnen bereitgestellte Programmierschnittstelle setzen Userland-Tools auf. Um nur zwei davon zu nennen:

- *ippp* stellt eine PPP-Verbindung über ISDN bereit. Damit ist also eine Internet-Anbindung über ISDN statt über langsame analoge Modems möglich. Vermutlich war das der Hauptgrund für die Entstehung des *ISDN4Linux*-Projekts – endlich mal richtig schnell (zumindest doppelt so schnell als per Modem) Daten übertragen zu können.

- *isdnlog* vermag es, alle möglichen Ereignisse auf dem S_0-Bus mitzuprotokollieren. Dazu gehören entgangene Anrufe, die Aktivierung und Deaktivierung von Rufumleitungen und vieles mehr (also etwas, das ich *unbedingt* nutzen möchte, zumal aus mir unverständlichen Gründen eine Menge Leute keine Lust haben, ihr Anliegen unserem Anrufbeantworter anzuvertrauen).

---

* Man unterscheidet generell zwischen aktiven und passiven IDSN-Karten. Der Knackpunkt dabei ist die auf den Karten vorhandene bzw. nicht vorhandene Eigenintelligenz. Mit aktiven Karten ist man i. allg. fein heraus, bei den Passiven dagegen kommt es auf die Anwendung an, ob man Glück oder Pech hat.

† siehe `http://www.isdn4linux.org`

‡ siehe `http://www.capi4linux.org`

Das andere Projekt, *CAPI4Linux*, hat das Ziel, CAPI* auf LINUX zu portieren. Das ist *der* plattform übergreifend verfügbare Standard, wenn es um die Nutzung von ISDN-basierten Geräten in Rechnern geht. Insbesondere dann, wenn man Anrufe entgegennehmen oder Faxe senden oder empfangen möchte.

Wie Du siehst, bieten beide Projekte sehr interessante Möglichkeiten für Dich. Jetzt gibt es bloß ein Problem: Die Treiber von *ISDN4Linux* verwenden *CAPI4Linux* nicht und umgekehrt.

Die große Aufgabe besteht also darin, *ISDN4Linux* und *CAPI4Linux* zur Kooperation zu überreden, so dass Du die Fähigkeiten von *isdnlog* nutzen, aber trotzdem Anrufe und Faxe per CAPI entgegennehmen kannst. Das Kernelmodul `capidrv` leistet genau diese Kopplung eines CAPI-basierten Treibers an *ISDN4Linux*. Mit anderen Worten: *ISDN4Linux* benötigt nun nicht mehr `hisax` als Treiber, sondern bedient sich – unter Vermittlung von `capidrv` – CAPI-basierter Treiber.

Jetzt brauchst Du nur noch einen für Deine ISDN-Karte geeigneten Treiber, der „nach oben“ CAPI-Dienste anbietet. Sollte es so etwas für Deine ISDN-Karte nicht geben, wirst Du Dich mit *ISDN4Linux* begnügen müssen.

Freundlicherweise bietet AVM für die Fritz-Serie genau so einen Treiber an.† Obwohl dieser Treiber ursprünglich für SuSE Linux gedacht war, tut er mindestens genauso gut unter Debian GNU/Linux.

Ein Hinweis noch, bevor wir loslegen: Ich habe das Ganze mit Kernel 2.6.12-1 (aus *Unstable*) ausprobiert. Dieser enthält freundlicherweise bereits alle Kernel-Treiber für die folgende Methode, so dass ich mir *keinen* eigenen Kernel bauen musste. Welche Module das sind bzw. welche Kernelkonfiguration dafür erforderlich ist, steht in der Datei `compile-help-german.txt` aus dem vom AVM heruntergeladenen Treiber-Archiv. Ich gehe aber davon aus, dass auch der mit *Stable* mitgelieferte Standard-Kernel bereits diese Module enthält. Ansonsten sei auf meine Kurzanleitung zum Bauen eines eigenen Kernels in Kapitel 18 auf Seite 137 verweisen.

Zunächst musst Du ein paar Pakete installieren: *isdnlog*, *isdnlog-data* und *isdnutils-base* sowie *isdnactivecards*. Letzteres sorgt übrigens für die automatische Installation der beiden Pakete *capiutils* und *libcapi20-2*.

Jetzt passt Du die Datei `/etc/isdn/capi.conf` so an, dass *als einziger* der gewünschte Treiber, in meinem Fall `fcpci`, verwendet wird:

/etc/isdn/capi.conf

```
1  # card      file    proto   io      irq mem cardnr  options
2  #b1isa     b1.t4   DSS1    0x150   7   -   -       P2P
3  #b1pci     b1.t4   DSS1    -       -   -   -
4  #c4        c4.bin  DSS1    -       -   -   -
5  #c4        -       DSS1    -       -   -   -
```

* **C**ommon ISDN-**A**pplication **P**rogramming **I**nterface

† Für meine Fritz!Card PCI ist das der unter dem URL `ftp://ftp.avm.de/cardware/fritzcrd.pci/linux/suse.93/fcpci-suse93-3.11-07.tar.gz` Verfügbare. Solltest Du eine andere Fritz!Card besitzen, musst Du Dir den passenden Treiber über die Web-Seite von AVM (`http://www.avm.de`) heraussuchen.

```
#c4          -        DSS1     -        -    -    -        P2P
#c4          -        DSS1     -        -    -    -        P2P
#c2          c2.bin   DSS1     -        -    -    -
#c2          -        DSS1     -        -    -    -
#t1isa       t1.t4    DSS1     0x340    9    -    0
#t1pci       t1.t4    DSS1     -        -    -    -
fcpci        -        -        -        -    -    -
#fcclassic   -        -        0x150    10   -    -
```

Jetzt gilt es, das heruntergeladene Archiv mit dem Treiber auszupacken und unter dem Kernel zu compilieren, auf dem er laufen soll. Wenn Du einen neuen Kernel einspielst, musst Du diesen Vorgang natürlich erneut durchführen.

```
khazad-dum:~ # cd /usr/src
khazad-dum:/usr/src # tar xvzf <pfad>/fcpci-suse93-3.11-07.tar.gz
...
khazad-dum:/usr/src # cd fritz
khazad-dum:/usr/src/fritz # make install
...
khazad-dum:/usr/src/fritz # depmod -a
khazad-dum:/usr/src/fritz #
```

Sollten bei Dir die HiSax-Kernelmodule bereits geladen sein, müssen sie entladen und anschließend die neuen CAPI-Module geladen werden. Dazu sind die Dienste zu stoppen, die darauf aufsetzen, anschließend wird entladen und dann das CAPI gestartet.

```
khazad-dum:~ # /etc/init.d/capiutils stop
Stopping ISDN CAPI Cards: done
khazad-dum:~ # /etc/init.d/isdnutils stop
Stopping ISDN services... iprofd isdnlog.
khazad-dum:~ # lsmod|cut -d' ' -f 1|grep hisax
hisax_fcpcipnp
hisax_isac
hisax
khazad-dum:~ # rmmod hisax_fcpcipnp
khazad-dum:~ # rmmod hisax_isac
khazad-dum:~ # rmmod hisax
khazad-dum:~ # /etc/init.d/capiutils start
Starting ISDN CAPI Cards....
khazad-dum:~ #
```

Die CAPI-Treiber sollten damit tun. Ruf jetzt *capiinfo* auf, um diese Annahme zu verifizieren:

```
khazad-dum:~ #
Number of Controllers : 1
```

```
Controller 1:
Manufacturer: AVM GmbH
CAPI Version: 2.0
Manufacturer Version: 3.101-07  (49.23)
Serial Number: 123456789
BChannels: 2
...
khazad-dum:~ #
```

Ausgezeichnet, das hat geklappt. Jetzt musst Du die Brücke zwischen *CAPI4Linux* und *ISDN4Linux* errichten. Füge dazu zunächst die folgenden Zeilen in `/etc/modutils/aliases` hinzu:

```
alias char-major-43 capidrv
alias char-major-44 capidrv
alias char-major-45 capidrv
```

Anschließend musst Du das Kommando `update-modules` ausführen. Dieses aktualisiert die Datei `/etc/modules.conf`.

Solltest Du *Logcheck* einsetzen (vgl. Abschnitt 16.1 auf Seite 127), erhältst Du von diesem für jeden eingehenden Anruf eine Warnmeldung. Dies müssen wir umgehen, indem wir im Verzeichnis `/etc/logcheck/ignore.d.server` eine Datei anlegen, die die `capidrv`-Meldungen ignoriert. Du brauchst dazu folgenden Inhalt:

```
^\w{3} [ :0-9]{11} [._[:alnum:]-]+ kernel: capidrv-[0-9]+: incoming \
    call [0-9]{0,},[0-9]{0,},[0-9]{0,},[0-9]{0,}
^\w{3} [ :0-9]{11} [._[:alnum:]-]+ kernel: capidrv-[0-9]+: patching \
    si2=[0-9] to 0 for VBOX
```

Ich selbst habe noch weitere reguläre Ausdrücke darin stehen, die allerdings aus der Zeit stammen, als *ISDN4Linux* auf meinem Server über `hisax` lief. Insofern will ich Dich damit nicht langweilen (und mich auch nicht damit, herauszufinden, ob die noch nötig sind). Falls Dir weitere Meldungen auffallen, kannst Du die sicher auch nach obigem Muster selbst in dieser Datei einpflegen.

Nun solltest Du den Start von *ISDN4Linux* ausprobieren. Wenn Du – wie ich – nicht per ISDN ins Internet willst, solltest Du zuvor jedoch noch dafür sorgen, dass *ISDN4Linux* beim Start nicht über eine fehlende Konfiguration jammert, indem Du

```
khazad-dum:~ # touch /etc/isdn/noconfig
khazad-dum:~ #
```

eingibst. Wenn Dich das Gejammere dagegen nicht stört, kannst Du diesen Schritt getrost auslassen.

Beim Start von *ISDN4Linux* über CAPI hatte ich das Problem, dass das Modul `capidrv` nicht automatisch geladen wurde. Ich habe das Problem pragmatisch dadurch gelöst, dass ich `/etc/init.d/isdnutils` so erweitert habe, dass das Modul händisch nachgeladen wird. Nicht schön, tut aber.*

```
...
```

* Kurzes Googlen hat ergeben, dass andere da auch nicht einfallsreicher als ich gewesen zu sein scheinen.

```
case "$1" in
  start)
    modprobe capidrv
    if [ ! -z "$2" ]; then
...
```

Spätestens jetzt solltest Du die *isdnutils* starten können:

```
khazad-dum:~ # /etc/init.d/isdnutils start
Starting ISDN services... iprofd isdnlog.
khazad-dum:~ #
```

Das sieht nun zwar zunächst unheimlich gut aus, doch beim nächsten Reboot hatte ich das Problem, dass das Modul `hisax` wieder geladen wurde und damit die CAPI-Treiber nicht aktiviert werden konnten. Hartnäckiges kleines Biest. Laut der Einträge in `/etc/hotplug/blacklist/capiutils` sollte das eigentlich nicht passieren. Dummerweise funkt uns *discover* dazwischen. Abhilfe verschafft die folgende Ergänzung am Ende von `/etc/discover.conf`:

```
# use CAPI only
skip hisax
skip hisax_fcpcipnp
skip hisax_isac
```

Jetzt bietet sich ein Reboot an, um zu sehen, ob ISDN auch dann noch einwandfrei läuft.

Einen Tipp habe ich noch für Dich in Bezug auf *isdnlog*: In der Datei `/etc/isdn/callerid.conf` können Zuordnungen von Namen und Telefonnummern für *isdnlog* vorgenommen werden:

```
khazad-dum:~ # cat /etc/isdn/callerid.conf
[MSN]
NUMBER = +49 911/12345678
ALIAS  = Wir

[MSN]
NUMBER = +49 911/87654321
ALIAS  = Jemand anders
khazad-dum:~ #
```

Im Protokoll, das Du unter `/var/log/isdn/isdnlog` findest, tauchen dann statt der Nummern, mit denen die meisten Menschen wenig anfangen können, die als `ALIAS` eingetragenen Namen auf.

Sollte es Dich übrigens stören, wenn *isdnlog* beim Start “Error: could not load holidays from `/usr/share/isdn/holiday-de.dat`: No such file or directory” mault, kannst Du Dir einen Tar-Ball der `isdn4k-utils` holen* und damit folgendes tun:

```
khazad-dum:~ # cd /tmp
```

* ftp://ftp.isdn4linux.de/pub/isdn4linux/utils/isdn4k-utils.v3.2p1.tar.bz2

```
khazad-dum:/tmp # tar xvjf <pfad>/isdn4k-utils.v3.2p1.tar.bz2
...
khazad-dum:/tmp # cp isdn4k-utils/isdnlog/holiday-de.dat \
                    /usr/share/isdn
khazad-dum:/tmp # /etc/init.d/isdnutils reload isdnlog
Restarting isdnlog....
khazad-dum:/tmp #
```

Soviel zu *ISDN4Linux*. Lass uns nun mit *CAPI4Linux* herumspielen.

Um Anrufe entgegennehmen zu lassen und Faxe senden und empfangen zu können, war für mich die optimale Lösung die Verwendung von *CapiSuite*. Willst Du dagegen nur faxen, kann für Dich möglicherweise *HylaFAX* die bessere Lösung sein.

Die Verwendung von *CapiSuite* gedenke ich an dieser Stelle nicht weiter zu erläutern, da Du prima Dokumentation auf der zugehörigen Web-Seite* findest. Alles zum Einrichten Wichtige erläutere ich Dir natürlich im Folgenden.

Die Installation von *CapiSuite* erfolgt durch einfaches Einspielen des Pakets *capisuite*. Anschließend gilt es, im Verzeichnis `/etc/capisuite` die Konfigurationsdateien für die Anrufbeantworter- und Fax-Funktionalität an Deine Anforderungen anzupassen. Ich zeige Dir das anhand eines `rcsdiff` auf:

```
khazad-dum:~ # cd /etc/capisuite
khazad-dum:/etc/capisuite # rcsdiff answering_machine.conf
===================================================================
RCS file: RCS/answering_machine.conf,v
retrieving revision 1.1
diff -r1.1 answering_machine.conf
57c57
< user_audio_files="0"
---
> user_audio_files="1"
86c86
< voice_email_from="capisuite daemon <root>"
---
> voice_email_from="KenobiVoice <obiwan@arda.lan>"
147a148,154
> [obiwan]
> voice_numbers="12345678"
> voice_action="MailAndSave"
> voice_email="obiwan@arda.lan,bunny@arda.lan"
> record_length="180"
> pin="12345678"
khazad-dum:/etc/capisuite # rcsdiff fax.conf
===================================================================
RCS file: RCS/fax.conf,v
retrieving revision 1.1
```

* http://www.capisuite.de

```
diff -r1.1 fax.conf
101c101
< fax_stationID="+49 000 0000"
---
> fax_stationID="+49 89 123456789"
115c115
< fax_email_from="capisuite daemon <root>"
---
> fax_email_from="KenobiFax <obiwan@arda.lan>"
163a164,169
> [obiwan]
> fax_numbers="123456789"
> fax_headline="Obi-Wan Kenobi - sent by CapiSuite"
> fax_email="obiwan@arda.lan,bunny@arda.lan"
> fax_action="MailAndSave"
>
khazad-dum:/etc/capisuite #
```

Wichtig ist – neben der Angabe der korrekten Rufnummern – die korrekte Angabe der jeweiligen *Absenderadresse*. `voice_email_from` und `fax_email_from` *muss* angepasst werden, weil *Exim* keine Adressen ohne Domainangabe mag.

Vergiss abschließend keinesfalls, in der Datei `/etc/default/capisuite` die Variable `run_capisuite_daemon` auf `y` zu setzen. Sonst wird *CapiSuite* nicht automatisch gestartet.

In den jeweiligen benutzerspezifischen Konfigurationsabschnitten ist festgelegt, dass unter den jeweiligen Rufnummern (MSNs) eingehende Anrufe und Faxe per E-Mail an `obiwan` und `bunny` geschickt werden. Übrigens werden bei der derzeitigen Konfiguration eingegangene Anrufe sowie alle Faxe bis zum St.-Nimmerleinstag aufgehoben. Wie man dafür sorgt, dass da periodisch aufgeräumt wird, ist in `/usr/share/doc/capisuite/examples` beschrieben.

KAPITEL 10

Namensdienste

Dieses Kapitel befasst sich mit der Bereitstellung von Namensdiensten für Dein lokales Netz.

10.1 Nameserver

Ein Nameserver wandelt Rechnernamen – die menschliche Art – in IP-Adressen – so mögen es die Computer – und umgekehrt um.

Als klassischer Nameserver gilt der *BIND*[*]. Er stellt nicht nur den eigentlichen Nameserver bereit, sondern auch eine Systembibliothek, die solche Namen bzw. IP-Adressen ineinander auflösen kann.

Der *BIND* ist eigentlich für große Installationen gedacht und für meinen Zweck völlig überdimensioniert. Da er aber gleichzeitig auch interessant ist, stört mich das nicht.[†]

Zur Installation eignet sich

```
khazad-dum:~ # apt-get -u install bind9 bind9-doc
```

Dabei werden noch einige andere Bibliotheken eingespielt. Und es lohnt sich für Dich, erst mal die Dokumentation von *bind9* anzusehen. Darin ist nämlich auch die Migration der Konfigurationsdateien von *bind* in Version 8 beschrieben (falls man denn welche hat); außerdem findet sich ein Hinweis, wie man den *bind* in einem Chroot-Jail laufen lassen kann.[‡]

Die Information in *bind9-doc* fand ich eher enttäuschend. Insgesamt steht man etwas verloren da, wenn man *bind* nicht kennt. Aber nicht verzagen, denn es gibt das *DNS-HowTo*.[§] Außerdem zeige ich Dir, wie man ein lokales Netz richtig versorgt. Wenn Du nicht mehr willst als das und mir vertraust, dass ich das richtig mache (was Du getrost

[*] **B**erkeley **I**nternet **N**ame **D**omain

[†] Wer will, kann sich aber auch mal *dnsmasq* und Konsorten ansehen. Für die meisten Anwender dürfte das völlig ausreichen.

[‡] Genaueres ist in http://en.tldp.org/HOWTO/Chroot-BIND-HOWTO.html beschrieben.

[§] Zu finden unter http://www.tldp.org/HOWTO/DNS-HOWTO.html.

kannst), dann kannst Du Dir die Lektüre des DNS-HowTo auch schenken. Solltest Du aber nur, wenn Du extrem wenig Zeit hast, denn das Teil ist echt interessant.

Die statischen Konfigurationsdateien von *bind* findest Du unter `/etc/bind`. Damit Du da nichts kaputt machst, checkst Du erstmal alle Dateien unter `/etc/bind` in eine Versionsverwaltung ein (vgl. Kapitel 5.4 auf Seite 31). Und nein, das ist nicht nur etwas für Weicheier. Gerade die *BIND*-Konfiguration kann extrem nervenaufreibend sein. Also sorge dafür, dass Du einen definierten Stand hast, zu dem Du wieder zurück kannst.

Übrigens ist in `/etc/default/bind9` voreingestellt, dass *bind* als User `bind` laufen soll. Die Doku sagt dazu, dass das zwar sicherer sei, aber den Nachteil habe, dass dann keine neuen Interfaces gefunden werden könnten, die dynamisch auftauchen – das betrifft hauptsächlich PCMCIA sowie verschiedene Tunnel. Damit habe ich kein Problem, deshalb belasse ich es dabei. Sei aber darauf hingewiesen, falls Du so etwas basteln möchtest.

Jetzt beginnen wir mit der Fummelei. Als erstes modifizierst Du die Datei `/etc/bind/named.conf.options`. Wie ich schon sagte, setze ich einen Router ein. Dieser verfügt über einen internen Nameserver, der Anfragen an die Nameserver meines ISPs* weiterleitet, so dass meine Einträge folgendermaßen aussehen:

`/etc/bind/named.conf.options`

```
1  options {
2          directory "/var/cache/bind";
3
4          forwarders {
5                  192.168.1.254;
6          };
7
8          allow-query { 192.168.1.0/24; localhost; };
9
10         auth-nxdomain no;    # conform to RFC1035
11 };
```

Die im `forwarders`-Abschnitt stehende IP-Adresse ist die meines Routers. Du solltest hier die Adresse der Nameserver Deines Providers eintragen oder aber die Deines Routers, so Du einen mit integriertem Nameserver hast.

Jetzt passt Du die Datei `/etc/bind/named.conf.local` folgendermaßen an (vorausgesetzt, Dein Netz soll auch `arda.lan` heißen und im Adressbereich 192.168.1 liegen – ansonsten sollten die erforderlichen Änderungen trivial sein):

`/etc/bind/named.conf.local`

```
1  zone "arda.lan" {
2          type master;
3          notify no;
4          file "/etc/bind/db.arda.lan";
```

* Die Information, welche Nameserver meines ISPs aktuell gültig sind, wird beim Verbindungsaufbau an meinen Router übermittelt, so dass sie ständig aktuell ist.

```
};

zone "1.168.192.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/db.192.168.1";
};
```

Nun musst Du noch die beiden darin angegebenen Zone-Files erstellen:

/etc/bind/db.arda.lan

```
$TTL    3D
@ IN SOA khazad-dum.arda.lan.    root.khazad-dum.arda.lan. (
          2004122102    ; Serial
          8H            ; Refresh
          2H            ; Retry
          4W            ; Expire
          1D )          ; Negative Cache TTL
;
; Name Servers
;
@ IN NS khazad-dum.arda.lan.
;
; Mail Exchangers
;
@ IN MX 10 khazad-dum.arda.lan.
;
; Computer Addresses
;
imladris    IN A     192.168.1.1
lorien      IN A     192.168.1.2
khazad-dum  IN A     192.168.1.3
morannon    IN A     192.168.1.254

cvs         IN CNAME khazad-dum
gateway     IN CNAME morannon
imap        IN CNAME khazad-dum
ntp         IN CNAME morannon
proxy       IN CNAME khazad-dum
router      IN CNAME morannon
smtp        IN CNAME khazad-dum
svn         IN CNAME khazad-dum
```

/etc/bind/db.192.168.1

```
$TTL 3D
@ IN SOA khazad-dum.arda.lan.    root.khazad-dum.arda.lan. (
```

```
3           2004122002    ; Serial
4           8H            ; Refresh
5           2H            ; Retry
6           4W            ; Expire
7           1D )          ; Negative Cache TTL
8
9       NS    khazad-dum.arda.lan.
10
11  1   PTR   imladris.arda.lan.
12  2   PTR   lorien.arda.lan.
13  3   PTR   khazad-dum.arda.lan.
14  254 PTR   morannon.arda.lan.
```

Wie Du siehst, habe ich in meiner `db.arda.lan`-Zone nicht nur die *Address Records* (`IN A`), sondern auch `CNAME`s festgelegt, die Aliases auf „richtige“ Namen bilden. Wozu, fragst Du? Nun, es könnte ja sein, dass mir der Name `khazad-dum` irgendwann nicht mehr gefällt und ich den Rechner in `server` umbenenne. Dann müsste ich auf all meinen Rechnern und teilweise in den Mail-Clients den Namen des Mailservers umsetzen. So dagegen heißt mein Mailserver einfach zusätzlich zu seinem eigentlichen Namen `imap` bzw. `smtp`, so dass ich davon verschont bleibe. Clever, nicht wahr? Wusste ich. Danke.

Wenn Du die Dateien zukünftig editierst, um neue IP-Adressen bzw. Namen einzutragen, vergiss nicht, den als „`serial`“ gekennzeichneten Eintrag anzupassen. Bei mir ist das immer das Änderungsdatum mit einer angehängten Ziffer. Ändere ich das erste Mal, ist das die 01, ändere ich ein zweites Mal, verwende ich 02 – Du blickst das System, hoffe ich. Sinn des Ganzen ist der: Der Nameserver muss von Dir davon in Kenntnis gesetzt werden, dass sich an den Dateien etwas geändert hat. Das entscheidet er an genau dieser „`serial`“. Daraus folgt: Hat die sich nicht geändert, denkt der Nameserver, dass sich die Datei wohl auch nicht geändert hat. Die restlichen Inhalte der Datei wertet er dazu gar nicht aus. Natürlich ist das nur relevant, wenn Du den Nameserver nicht beendest und neu startest (kann fatal sein, wenn Programme währenddessen Namen auflösen wollen), sondern die korrekte Methode dazu verwendest, nämlich

```
khazad-dum:~ # rndc reload
```

Spätestens jetzt solltest Du wirklich paranoid werden: Falls Dein Netz nicht durch eine Firewall geschützt wird und Du einen automatischen Verbindungsaufbau konfiguriert haben solltest, ziehst Du bitte das Modem- oder ISDN-Kabel aus der Buchse bzw. steckst das Netzwerkkabel aus der Netzwerkkarte aus, die die Verbindung nach draußen bereitstellt. Es könnte nämlich sein, dass der *BIND* eine Verbindung zu seinen `forwarders` aufbauen möchte und damit eine Internet-Verbindung hochgeht. Ohne schützende Firewall kann das fatale Folgen für die Rechner in Deinem lokalen Netz haben.

Der Nameserver läuft nach der Installation bereits. Daher brauchst Du ihn nicht mittels der üblichen Zauberformel

```
khazad-dum:~ # /etc/init.d/bind9 start
```

zu starten, sondern ihn lediglich freundlich mittels

```
khazad-dum:~ # rndc reload
```

zum erneuten Einlesen seiner Konfiguration zu veranlassen, und beobachtest die Meldungen in `/var/log/daemon.log`, um Dich davon zu überzeugen, dass der Start reibungslos verläuft. Allerdings ist das unwahrscheinlich, wenn Du das erste Mal eine Zonen-Datei geschrieben hast. Tipp: Überprüfe genau, ob Deine Dateien den meinen entsprechen – insbesondere Punkte am Ende eines Namens spielen eine entscheidende Rolle; ansonsten erinnere ich noch mal an die Lektüre des DNS-HowTos, falls Du Schwierigkeiten haben solltest.

Ein einwandfrei ablaufender Reload sollte im Systemprotokoll etwa folgende Meldungen produzieren:

```
named: loading configuration from '/etc/bind/named.conf'
named: zone 1.168.192.in-addr.arpa/IN: loaded serial 2004122001
named: zone arda.lan/IN: loaded serial 2004122001
```

Der ist aber nur die halbe Miete, denn falls Du bei der Installation bereits einen Nameserver angegeben hast, so wird weiterhin der verwendet, nicht jedoch der, den Du gerade mehr oder weniger mühevoll eingerichtet hast. Sorge daher dafür, dass die Datei `/etc/resolv.conf` folgendermaßen aussieht:

```
search arda.lan
nameserver 127.0.0.1
```

Nun testest Du Deinen Nameserver, indem Du ausprobierst, ob Du einen Namen aus Deinem Netz korrekt auflösen kannst. Dabei kannst Du, wie im Folgenden vorgeführt, getrost einen `CNAME`- statt eines `A`-Records verwenden:

```
khazad-dum:~ # dig +norec +noques +nostats +nocmd imap.arda.lan
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6566
;; flags: qr aa ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; ANSWER SECTION:
imap.arda.lan.        259200  IN  CNAME   khazad-dum.arda.lan.
khazad-dum.arda.lan.  259200  IN  A       192.168.1.3

;; AUTHORITY SECTION:
arda.lan.             259200  IN  NS      khazad-dum.arda.lan.

khazad-dum:~ #
```

Die Optionen `+norec +noques +nostats +nocmd` kannst Du ruhig weglassen. Sie sorgen lediglich dafür, dass die Ausgabe nicht ganz so lang wird. Beachte übrigens, dass eine `AUTHORITY SECTION` vorhanden ist, in der unser Nameserver steht.

Jetzt versuchst Du, Deine Domain aufzulösen:

```
khazad-dum:~ # dig +norec +noques +nostats +nocmd any arda.lan
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22820
;; flags: qr aa ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; ANSWER SECTION:
arda.lan.   259200  IN  SOA    khazad-dum.arda.lan. \
                               root.khazad-dum.arda.lan. \
                               2004122102 28800 7200 2419200 86400
arda.lan.   259200  IN  NS     khazad-dum.arda.lan.
arda.lan.   259200  IN  MX  10 khazad-dum.arda.lan.

;; ADDITIONAL SECTION:
khazad-dum.arda.lan.   259200  IN      A       192.168.1.3

khazad-dum:~ #
```

Du siehst: Auch das klappt. Die Ausgabe sollte eigentlich auch eine `AUTHORITY SECTION` enthalten, was aber interessanterweise nicht der Fall ist. Der Grund ist mir derzeit unklar, obwohl ich das DNS-HowTo gelesen und ziemlich lange per Google gesucht habe. Macht aber nix, Du kannst mittels eines versuchten *Zone Transfers* schauen, ob sich Dein Nameserver verhält wie gewünscht:

```
khazad-dum:~ # dig arda.lan axfr
; <<>> DiG 9.2.4 <<>> arda.lan axfr
;; global options:  printcmd
arda.lan.             259200  IN      SOA     khazad-dum.arda.lan. \
     root.khazad-dum.arda.lan. 2004122102 28800 7200 2419200 86400
arda.lan.             259200  IN      NS      khazad-dum.arda.lan.
arda.lan.             259200  IN      MX      10 khazad-dum.arda.lan.
cvs.arda.lan.         259200  IN      CNAME   khazad-dum.arda.lan.
gateway.arda.lan.     259200  IN      CNAME   morannon.arda.lan.
imap.arda.lan.        259200  IN      CNAME   khazad-dum.arda.lan.
imladris.arda.lan.    259200  IN      A       192.168.1.1
khazad-dum.arda.lan.  259200  IN      A       192.168.1.3
lorien.arda.lan.      259200  IN      A       192.168.1.2
morannon.arda.lan.    259200  IN      A       192.168.1.254
ntp.arda.lan.         259200  IN      CNAME   morannon.arda.lan.
proxy.arda.lan.       259200  IN      CNAME   khazad-dum.arda.lan.
router.arda.lan.      259200  IN      CNAME   morannon.arda.lan.
smtp.arda.lan.        259200  IN      CNAME   khazad-dum.arda.lan.
svn.arda.lan.         259200  IN      CNAME   khazad-dum.arda.lan.
arda.lan.             259200  IN      SOA     khazad-dum.arda.lan. \
     root.khazad-dum.arda.lan. 2004122102 28800 7200 2419200 86400
;; Query time: 12 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Dec 22 22:37:29 2004
;; XFR size: 30 records
```

```
khazad-dum:~ #
```

Und das tut er. Also ignorierst Du das Problem (so es denn überhaupt eines ist).

So weit, so gut. Fehlt die *Reverse Zone*, also die Ermittlung von Namen zu IP-Adressen. Auch dies testest Du jetzt, indem Du ein Reverse Lookup versuchst:

```
khazad-dum:~ # dig +norec +noques +nostats +nocmd -x 192.168.1.3
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30279
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; ANSWER SECTION:
3.1.168.192.in-addr.arpa. 259200 IN  PTR   khazad-dum.arda.lan.

;; AUTHORITY SECTION:
1.168.192.in-addr.arpa.   259200 IN  NS    khazad-dum.arda.lan.

;; ADDITIONAL SECTION:
khazad-dum.arda.lan.      259200 IN  A     192.168.1.3

khazad-dum:~ #
```

Sieht gut aus. Jetzt noch einen simulierten Zone Transfer für die Reverse Zone getestet:

```
khazad-dum:~ # dig -x 192.168.1 axfr
; <<>> DiG 9.2.4 <<>> @khazad-dum -x 192.168.1 axfr
;; global options:  printcmd
1.168.192.in-addr.arpa.     259200 IN  SOA   khazad-dum.arda.lan. \
       root.khazad-dum.arda.lan. 2004122002 28800 7200 2419200 86400
1.168.192.in-addr.arpa.     259200 IN  NS    khazad-dum.arda.lan.
1.1.168.192.in-addr.arpa.   259200 IN  PTR   imladris.arda.lan.
2.1.168.192.in-addr.arpa.   259200 IN  PTR   lorien.arda.lan.
254.1.168.192.in-addr.arpa. 259200 IN  PTR   morannon.arda.lan.
3.1.168.192.in-addr.arpa.   259200 IN  PTR   khazad-dum.arda.lan.
1.168.192.in-addr.arpa.     259200 IN  SOA   khazad-dum.arda.lan. \
       root.khazad-dum.arda.lan. 2004122002 28800 7200 2419200 86400
;; Query time: 5 msec
;; SERVER: 192.168.1.3#53(khazad-dum)
;; WHEN: Wed Dec 22 22:41:12 2004
;; XFR size: 9 records

khazad-dum:~ #
```

Klasse. Falls Du mit den oben abgedruckten vergleichbare Ergebnisse erzielst, ist alles im grünen Bereich.

Jetzt probierst Du noch aus, ob Du auch Namen wie `www.franken.de` auflösen kannst, damit Du siehst, ob das Forwarding funktioniert. Wenn auch das tut (dazu überprüfst Du die `ANSWER SECTION`, lässt aber die ganzen `+no...` weg, weil sonst u. a. dieser Abschnitt

unterdrückt wird), bist Du fertig mit der Konfiguration Deines eigenen Nameservers. Meinen Glückwunsch!

Pflege braucht der *bind9* nicht – abgesehen davon, dass er ab und an ein neues `root.hints`-File möchte, nämlich dann, wenn sich da etwas geändert hat. Aber im DNS-HowTo ist ein nettes Skript angegeben, das das für Dich erledigt. Den genauen URL sage ich Dir aber nicht, denn es lohnt sich wirklich, wenn Du Dich wenigstens mal durchs Inhaltsverzeichnis des HowTos arbeitest und vielleicht doch das eine oder andere interessante Kapitel findest ...

10.2 DHCP

Bevor Du das Paket *dhcp3-server* installierst, möchte ich Dich noch darauf hinweisen, dass Du *niemals* zwei DHCP-Server im selben Netz laufen lassen solltest. Das kann sehr subtile Fehler zur Folge haben.

Nach der Installation des Pakets fragt Dich das Konfigurationsprogramm, welche Netzwerkschnittstellen mit IP-Adressen versorgt werden sollen. Welche Du überhaupt hast, siehst Du per

```
khazad-dum:~ # ifconfig
eth0  Link encap:Ethernet  HWaddr 00:11:22:33:44:55
      inet addr:192.168.1.3  Bcast:192.168.1.255  Mask:255.255.255.0
      inet6 addr: fe80::250:daff:fe34:feb2/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:34994910 errors:0 dropped:0 overruns:0 frame:0
      TX packets:31049881 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:843906370 (804.8 MiB)  TX bytes:1573427601 (1.4 GiB)
      Interrupt:5 Base address:0xd000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:5628911 errors:0 dropped:0 overruns:0 frame:0
      TX packets:5628911 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:3035018235 (2.8 GiB)  TX bytes:3035018235 (2.8 GiB)
khazad-dum:~ #
```

In meinem Fall habe ich die Interfaces `eth0` (meine Netzwerkkarte) und `lo`, das Loopback-Interface. Bei Dir dürfte das ähnlich aussehen, weshalb Du `eth0` angibst.

Anders als die meisten Daemonen läuft der DHCP-Server nach der Installation *nicht* automatisch – eben deshalb, weil Du noch rumkonfigurieren musst. Daher nimmst Du

Dir jetzt die Konfigurationsdatei `/etc/dhcp3/dhcpd.conf` vor. Meine habe ich Dir im Folgenden abgedruckt.

/etc/dhcp3/dhcpd.conf

```
1  ddns-update-style none;
2
3  option domain-name "arda.lan";
4  option domain-name-servers 192.168.1.3;
5
6  default-lease-time 600; # 10 Minuten
7  max-lease-time 7200;    #  2 Stunden
8
9  authoritative;
10
11 log-facility local7;
12
13 subnet 192.168.1.0 netmask 255.255.255.0 {
14     range 192.168.1.5 192.168.1.252;
15     option subnet-mask 255.255.255.0;
16     option broadcast-address 192.168.1.255;
17     option routers 192.168.1.254;
18 }
19
20 host imladris {
21     hardware ethernet 00:e0:f0:a0:b0:c0;
22     fixed-address imladris.arda.lan;
23     option host-name "imladris";
24 }
25
26 host lorien {
27     hardware ethernet 00:0d:0e:0f:0a:0b;
28     hardware ethernet 00:0d:0c:0b:af:fe;
29     fixed-address lorien.arda.lan;
30     option host-name "lorien";
31 }
32
33 host morannon {
34     hardware ethernet 00:a0:b0:c0:d0:e0;
35     fixed-address morannon.arda.lan;
36     option host-name "morannon";
37 }
```

Ich habe meine Domain und die IP-Adresse meines Nameservers gleich am Anfang festgelegt. Damit werden die angeschlossenen Rechner versorgt. Die Zeilen 13–18 legen den Bereich für die zu vergebenden Adressen fest, aber auch verschiedene weitere Angaben, deren Bedeutung Du Dir sicher leicht erschließen kannst.

Die Zeilen 20–24 sind etwas Besonderes. Denn mit ihnen bestimme ich, dass ein Rechner mit einer bestimmten *MAC-Adresse* eine feste IP-Adresse erhalten soll. So etwas eignet sich nicht nur für Server, sondern für alle Rechner, die immer unter einem bestimmten Namen erreichbar sein sollen.

Die Zeilen 26–31 geben zwei MAC-Adressen an. Das ist also ein Rechner, der mit unterschiedlichen Netzwerkkarten angeschlossen werden kann. In meinem Fall ist das ein Notebook, das neben der normalen, aber eher selten genutzten Gigabit-Ethernet-Netzwerkkarte auch eine Funk-LAN-Karte besitzt.

Ach so: Du erinnerst Dich vielleicht, dass ich meinen Server während der Grundinstallationsphase per DHCP mit einer IP-Adresse habe versehen lassen. Da der Server nun selbst DHCP-Server spielen soll und man sich bekanntlich nicht selbst an den Haaren aus dem Sumpf ziehen kann (außer, man ist der BARON VON MÜNCHHAUSEN), muss ich die Datei `/etc/network/interfaces` nun von Hand von einer per DHCP bezogenen auf eine feste IP-Adresse umstellen:

```
khazad-dum:~ # cat /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
    address 192.168.1.3
    netmask 255.255.255.0
    network 192.168.1.0
    gateway 192.168.1.254
    broadcast 192.168.1.255

khazad-dum:~ #
```

Beende nun den anderen in Deinem Netz laufenden DHCP-Server (so vorhanden), starte Dein Netzwerk mittels

```
khazad-dum:~ # /etc/init.d/network restart
```

neu und lasse dann den gerade frisch konfigurierten DHCP-Daemon auf Deinem Server von der Leine. Um Dich von dessen einwandfreier Funktion zu überzeugen, solltest Du von einem Deiner anderen Rechner aus eine IP-Adresse anfordern.

KAPITEL 11

E-Mail-Dienste

Dieses Kapitel lehrt Dich, wie Du deinen Server optimal für den Umgang mit E-Mail einrichtest.

11.1 Cyrus IMAPd

> *Cyrus is easy, all the trouble is in SASL, and even that becomes easy after you understand how SASL works.*
>
> – aus `/usr/share/doc/cyrus-common/README.Debian.simpleinstall`

Wenn Du schon weißt, was IMAP (oder genauer: IMAP4) ist, dann dürfte Dich die nun folgende Einführung langweilen, so dass Du sie getrost überspringen kannst. Alle andere dagegen sollten sie lesen – sonst entginge ihnen eine überaus angenehme Möglichkeit, E-Mail zu verwalten.

Ich zitiere mich mal selbst. Das folgende stammt aus einem meiner Artikel in einer *SnailMail*-Ausgabe des KNF.

> *Die zwei Verfahren POP3 (**P**ost **O**ffice **P**rotocol Version 3) und IMAP4 (**I**nternet **M**essage **A**ccess **P**rotocol Version 4) erfüllen grundsätzlich denselben Zweck: Beide ermöglichen es einem Rechner, über das Internet auf ein Postfach zuzugreifen, das auf einem anderen Rechner liegt. Dabei bezeichnet man das System, das das Postfach vorhält, als* Server *das andere als* Client.
>
> *Auch die Vorgehensweise beider Verfahren ist sehr ähnlich: Wenn ein E-Mail-Client auf ein Postfach zugreifen möchte, baut er eine TCP-Verbindung zum Server auf. Dort startet ein sog.* Daemon*, der serverseitig für das Handling des Postfachs verantwortlich ist. Hat das geklappt, tauschen Server (eigentlich der Daemon darauf) und Client eine Reihe von Kommandos und Antworten aus, bis die Verbindung abgebaut (oder unterbrochen) wird.*
>
> *Der wesentlich Unterschied jedoch liegt in der Art, auf die die E-Mail gespeichert wird. Bei POP3 lädt der Client die E-Mails aus dem Postfach auf die*

lokale Festplatte herunter und löscht sie anschließend auf dem Server. Die E-Mails liegen dann also auf dem Client-Rechner. Das ist solange gut und schön, wie eine Person nur von einem einzigen Rechner aus auf ihre E-Mail zugreifen möchte. Sobald ein zweiter Rechner ins Spiel kommt, wird dieses Verfahren unpraktikabel: Die eine E-Mail liegt dann nämlich auf Rechner A, die andere auf Rechner B.

IMAP4 wurde entwickelt, um diesem Problem zu begegnen. IMAP4 belässt die E-Mail auf dem Server. Ein Download findet nur zum Lesen der E-Mail statt. Dadurch findet ein Anwender unabhängig von seinem Aufenthaltsort immer die gleichen E-Mails auf dem Mailserver vor. Der Benutzer kann sich auf dem Server neben der eigentlichen inbox, in der neue E-Mail landet, mehrere andere Verzeichnisse (folder) definieren, zwischen denen einzelne E-Mails hin- und hergeschoben werden können. Natürlich kann E-Mail auch gelöscht oder in ein Verzeichnis auf dem Client verschoben werden.

Mancher POP3-Client, beispielsweise Netscapes Communicator, bietet inzwischen auch das Feature an, die E-Mails auf dem Server zu belassen. Im Vergleich zu IMAP4 ist diese Lösung jedoch bestenfalls eine Krücke, da die E-Mails zur Organisation mittels mehrerer Verzeichnisse doch wieder auf den lokalen Rechner heruntergeladen werden müssen – POP3 kennt eben, anders als IMAP4, kein Folder-Konzept.

IMAP4 bietet gegenüber POP3 noch einen weiteren, mehr technischen Vorteil: Wird der Datenaustausch zwischen dem POP3-Daemon und dem Mail-Client des Anwenders unterbrochen, etwa weil der Mail-Client abstürzt, so hält der Daemon das Postfach solange gesperrt, bis er sich sicher ist, dass der Client sich wirklich nicht mehr meldet. Startet der Benutzer seinen Mail-Client neu und versucht einen erneuten Verbindungsaufbau, schlägt dieser solange fehl, wie die Sperre durch den POP3-Daemon besteht – und das kann dauern. IMAP4 vermeidet dieses Problem, da der IMAP4-Daemon mehrere gleichzeitige Zugriffe auf das Postfach zulässt.

Ein Problem bleibt jedoch bei beiden Verfahren bestehen: Die E-Mail wird zwischen Server und Client unverschlüsselt über das Internet übertragen. Liest man daher im Büro seine private E-Mail, wäre es dem Systemadministrator der Firma, in der man angestellt ist, möglich, diese private E-Mail mitzulesen. Als Abhilfe gibt es die Möglichkeit, eine Verbindung aufzubauen, die per SSL verschlüsselt ist – man redet dann von POP3/S bzw. IMAP4/S. Dadurch wird das Mitlesen wirkungsvoll verhindert.

Für eine detaillierte Beschreibung beider Protokolle vgl. die folgenden RFCs (zu finden z. B. unter `http://rfc.sunsite.dk/`):

- *RFC 1725: POP3*
- *RFC 2060: IMAP4*

Du siehst, das ist etwas, das Du unbedingt haben willst, aber nie danach zu fragen wagtest. Ist aber überhaupt nicht weiter kompliziert – zumindest nicht mit dieser krassen Anleitung ...

Ich persönlich ziehe es vor, den *Cyrus IMAPd* zu verwenden. Andere mögen lieber den *Courier*. Jedem das seine. Du folgst am besten meinem Beispiel uns spielst die Cyrus-Pakete *cyrus21-imapd*, *cyrus21-common*, *cyrus21-admin*, *cyrus21-doc* und *cyrus21-clients*, die *Simple Authentication and Security Layer*-Pakete *sasl2-bin*, *libsasl2-modules* und *libsasl2* sowie *OpenSSL* (Paket *openssl*) ein.

Wenn was nicht tut, rate ich *dringend* dazu, in der Dokumentation von *cyrus-common* wenigstens `README.Debian` und `README.Debian.simpleinstall` zu lesen.

Als nächstes bearbeitest Du `/etc/cyrus.conf`. Darin entfernst Du das Kommentarzeichen (#) vor dem Dienst `imaps` und praktizierst eines vor den Dienst `pop3`. Sinn des Ganzen ist es, nur IMAP4 mit und ohne SSL-Unterstützung zuzulassen, aber kein POP3.

Dann ist die Datei `/etc/imapd.conf` an der Reihe. Darin entfernst Du die Kommentarzeichen vor folgenden Zeilen bzw. änderst die Zeilen wie folgt ab:

```
lmtp_downcase_rcpt: yes
admins:             cyrus
sasl_minimum_layer: 0
tls_cert_file:      /etc/ssl/cyrus.pem
tls_key_file:       /etc/ssl/cyrus.pem
```

Anschließend musst Du noch die Zertifikate generieren. Der saubere Weg wäre, eine eigene *Certification Authority* aufzusetzen und dann korrekte Zertifikate zu basteln. Das ist in Kapitel 19 auf Seite 141 ausführlich beschrieben. Wenn Du dazu jetzt keine Lust hast (verständlich), kannst Du das auch *quick and dirty* mit einem selbstsignierten Zertifikat machen, das Du über den Aufruf

```
khazad-dum:~ # openssl req -new -x509 -nodes \
                    -out /etc/ssl/cyrus.pem \
                    -keyout /etc/ssl/cyrus.pem -days 365
```

erzeugst. Dieses Zertifikat ist ein Jahr gültig. Beachte bitte, als *Common Name* den zu verwendenden Rechnernamen anzugeben, in diesem Fall `imap.arda.lan` – ansonsten mault Dein Mail-Client, dass der Name im Zertifikat nicht mit dem Namen des Rechners übereinstimmt. Ach so: Vergiss nicht, ein

```
khazad-dum:~ # chown cyrus.mail /etc/ssl/cyrus.pem
```

auszuführen – sonst darf der Cyrus IMAPd das Zertifikat nicht lesen, was leicht kontraproduktiv wäre.

Jetzt startest Du den Cyrus IMAPd nach dem üblichen Schema zum Neustart eines Daemons neu:

```
khazad-dum:~ # /etc/init.d/cyrus21 restart
```

Dann legst Du mit dem folgenden Kommando einen Account für den Admin-User `cyrus` an:

```
khazad-dum:~ # saslpasswd2 -c cyrus
```

Das wiederholst Du für Dich selbst und alle weiteren Benutzer:

```
khazad-dum:~ # saslpasswd2 -c obiwan
khazad-dum:~ # saslpasswd2 -c bunny
```

Stell sicher, dass das geklappt hat:

```
khazad-dum:~ # sasldblistusers2
cyrus@khazad-dum: userPassword
obiwan@khazad-dum: userPassword
bunny@khazad-dum: userPassword
khazad-dum:~ #
```

Abschließend musst Du noch die Mailboxen für Deine Benutzer anlegen. Das Kommando

```
khazad-dum:~ # cyradm --user cyrus localhost
Password:
localhost>
```

startet die Cyrus-eigene Verwaltungsoberfläche (eine Art Shell). Darin kannst Du dann Kommandos angeben (`?` liefert Dir eine Auflistung der möglichen Kommandos und deren Abkürzungen). Du legst nun die Mailbox für Dich und analog die weiteren Benutzer an. Anschließend beendest Du *cyradm*.

```
localhost> cm user.obiwan
localhost> cm user.bunny
localhost> exit
```

Solltest Du – so wie ich – einen alten Server durch den neuen ablösen wollen und auf dem alten bereits Deine E-Mail mittels eines beliebigen IMAP-Servers verwaltet haben, so gibt es einen sehr einfachen Weg, Deine Mails vom alten auf den neuen Server zu migrieren: Du brauchst ein Tool, das – ähnlich einem Mail-Client – lesend auf den alten und schreibend auf den neuen Server zugreift. Ich benutze dazu *imapsync*, das Du als Source von http://www.linux-france.org/prj/imapsync/ beziehen kannst.* Das Einspielen ist trivial. Es besteht darin, als `root` mittels des Kommandos

```
khazad-dum:~ # apt-get -u install libmail-imapclient-perl \
                    libmd5-perl libdigest-hmac-perl
```

ein paar benötigte Perl-Pakete zu Deinem System hinzuzufügen und im Verzeichnis mit der ausgepackten Version von *imapsync* als Benutzer `root` ein

```
khazad-dum:/tmp/imapsync-1.103 # make install
```

aufzurufen. Anschließend synchronisiert das im Folgenden abgedruckte Skript die Mailbox von User `obiwan` auf Deinem neuen Server mit der auf dem alten Server.† Denk daran, dass das Skript mittels

```
khazad-dum:~/bin # chmod u+x imapSync.sh
```

* In *Unstable* ist das im Mai 2005 auch in Form des Pakets *imapsync* eingetrudelt.

† Weitere interessante Optionen sind `--dry` zum Testen und `--delete` zum Löschen der Mails auf dem alten Server.

ausführbar gemacht und Dein Passwort in der Datei `~/pwdfile` hinterlegt sein muss.

`~/bin/imapSync.sh`

```
1  #!/bin/bash
2
3  SOURCE_HOST=theshire.arda.lan
4  SOURCE_USER=obiwan
5  SOURCE_PWDFILE=~/pwdfile
6  TARGET_HOST=imap.arda.lan
7  TARGET_USER=$SOURCE_USER
8  TARGET_PWDFILE=$SOURCE_PWDFILE
9
10 IMAPSYNC=‘which imapsync 2>/dev/null‘
11
12 if test "$IMAPSYNC" -a -x "$IMAPSYNC" ; then
13   $IMAPSYNC --noauthmd5 \
14             --host1 $SOURCE_HOST --user1 $SOURCE_USER \
15             --passfile1 $SOURCE_PWDFILE \
16             --host2 $TARGET_HOST --user2 $TARGET_USER \
17             --passfile2 $TARGET_PWDFILE
18 else
19         echo imapsync nicht gefunden.
20         exit 1
21 fi
```

Willst Du Deine Mails filtern, sobald sie an den Cyrus IMAPd übergeben werden, musst Du *Sieve* nutzen. Filtern ist prima, um Mail in mehrere Folder zu verteilen. So kannst Du beispielsweise als Spam gekennzeichnete Mail in Deinen `INBOX.Spam`-Folder packen lassen.* Auch das ist einfach. Erst erstellst Du ein Sieve-Skript. Eines, das Spam verschiebt, findest Du im Folgenden abgedruckt. Willst Du eines, das Spam einfach löscht (*nicht empfohlen*), entfernst Du das Kommentarzeichen in Zeile 10 und machst dafür eines an den Anfang von Zeile 9.

`~/filter.sieve`

```
1  require "fileinto";
2  require "reject";
3  require "regex";
4
5  # Spam ausfiltern
6  if anyof (
7      header :contains ["X-Spam-Flag"] ["YES"]
8  ) {
9      fileinto "INBOX.spam" ;
10 #    discard;
```

* Achtung: Dieser Folder *muss* bereits existieren; er wird von *Sieve nicht* angelegt.

```
    stop;
}

# den Rest in die INBOX packen
keep;
stop;
```

Dann schiebst Du *Sieve* diese Datei unter, indem Du die *sieveshell* nutzt. Fertig!

```
obiwan@khazad-dum:~ $ sieveshell localhost
connecting to localhost
Please enter your password:
> put filter.sieve
> activate filter.sieve
> list
filter.sieve  <- active script
> quit
obiwan@khazad-dum:~ $
```

11.2 Taylor-UUCP

Du wirst Deine E-Mail wahrscheinlich per POP3 bzw. IMAP4 von Deinem Provider abholen. Ich dagegen nutze beim KNF e. V.* **U**NIX to **U**NIX **Co**p**y**. Das ist eine Möglichkeit, Mail und News-Artikel *gepackt* zu übertragen. Ein Verfahren also, das rockt.

Zuerst steht die Installation per

```
khazad-dum:~ # apt-get -u install uucp
```

an. `/usr/share/doc/uucp/README.Debian` setzt Dich davon in Kenntnis, dass Du ein *Taylor-UUCP* vor Dir hast. Das ist erfreulich, denn die Konfiguration ist in dem Fall nicht weiter schwierig.

Zuerst checkst Du alle Konfigurationsdateien ein, damit Du den Ausgangsstand für den Notfall in Reserve hast:

```
khazad-dum:~ # cd /etc/uucp
khazad-dum:/etc/uucp # mkdir RCS
khazad-dum:/etc/uucp # ci -l *
```

Die Dateien `dial` und `port` lässt Du unverändert. In die Datei `call` trägst Du den Namen des anzurufenden Systems, Deinen Login-Namen und Dein Passwort ein. In `config` ergänzt Du als `nodename` Deinen Knotennamen. `passwd` enthält Deinen Login-Namen und das zugehörige Passwort. `sys` drucke ich im Folgenden ab:

/etc/uucp/sys

```
protocol gvG
protocol-parameter G packet-size 1024
```

* Kommunikationsnetzwerk Franken e. V., siehe http://www.franken.de

```
protocol-parameter G short-packets

# System 'chico'
system          chico
address         chico.franken.de
time            any
call-login      *
call-password   *
commands        rmail rsmtp bsmtp rmail
command-path    /usr/bin /usr/sbin /usr/lib/news/bin
port            TCP
protocol        t
myname          galileo
chat            ogin: \L word: \P
```

Abschließend prüfst Du die Berechtigungen: Alle Dateien gehören `root` und haben als Gruppe `uucp`. Ausnahme: `call` hat `uucp`. Guckst Du hier:

```
khazad-dum:/etc/uucp # ls -l
-rw-r--r--  1 root uucp  587 Sep 28 21:33 Poll
drwxr-xr-x  2 root root  115 Dec  8 21:07 RCS
-rw-r-----  1 uucp uucp  538 Dec  8 21:20 call
-rw-r--r--  1 root uucp  478 Dec  8 21:11 config
-rw-r--r--  1 root uucp  532 Sep 28 21:33 dial
-rw-r--r--  1 root uucp 2025 Sep 28 21:33 expire
-rw-r-----  1 root uucp  509 Dec  8 21:22 passwd
-rw-r--r--  1 root uucp  500 Sep 28 21:33 port
-rw-r--r--  1 root uucp 1686 Dec  8 21:24 sys

khazad-dum:/etc/uucp #
```

Wunderbar soweit. Nur: Jetzt schicke ich ein Passwort über ein unsicheres Netz, nämlich das Internet. Außerdem möchte nicht nur ich mich gegenüber meiner Gegenstelle authentifizieren müssen, sondern ich möchte auch, dass diese sich mir gegenüber als der korrekte Ansprechpartner ausweist. Du merkst: Ich traue keinem. Überall Schlapphüte. Früher, als UUCP über eine Direktverbindung per Telefonleitung lief, war das unproblematischer. Ich werde also einen zusätzlichen Schutz brauchen: Eine SSL-Verschlüsselung mit Überprüfung der Server-Zertifikate. Das Ergebnis des Ganzen ist *UUCP over SSL*.

Dazu installierst Du nach Deiner persönlichen Vorliebe das Paket *stunnel* (das ist *Stunnel* in Version 3) bzw. *stunnel4* (*Stunnel* in Version 4). Dann platzierst Du das serverseitige Zertifikat in `/var/spool/uucp/certificates`. Dort führst Du noch ein

```
khazad-dum:/var/spool/uucp/certificates # c_rehash .
```

durch (wozu das gut ist, erläutert Kapitel 19 auf Seite 141). Achte wiederum auf die korrekten Rechte:

```
khazad-dum:/var/spool/uucp/certificates # ls -l
```

```
lrwxrwxrwx  1 uucp root   14 2004-12-13 17:42 7f07aa8a.0 \
                                               -> knfca-2003.pem
-rw-------  1 uucp root 3428 2004-09-28 12:11 chico-cert.pem
lrwxrwxrwx  1 uucp root   14 2004-12-13 17:42 eab70464.0 \
                                               -> chico-cert.pem
-rw-------  1 uucp root 1720 2003-10-01 13:01 knfca-2003.pem

khazad-dum:/var/spool/uucp/certificates #
```

Ach ja: Vergiss nicht, den `$PATH` des Benutzers `uucp` um `/usr/sbin` zu ergänzen – andernfalls werden weder *uucico* noch *stunnel* bzw. *stunnel4* gefunden, was natürlich fatal wäre. Ich mache das folgendermaßen:

/var/spool/uucp/.bashrc

```
# Aliases laden
test -e ~/.alias && . ~/.alias

# Pfad setzen
export PATH=/usr/sbin:"$PATH"
```

/var/spool/uucp/.profile

```
# Mal wird .bashrc, mal .profile ausgeführt. Ich füge alles in .bashrc
# ein und lade dieses auch in .profile.
if test -f ~/.bashrc; then
    . ~/.bashrc
fi
```

Jetzt bearbeitest Du noch zwei Dateien in `/etc/uucp`:

1. An `port` hängst Du flugs folgende Zeilen an:

   ```
   #
   # UUCP over SSL
   #
   port SSL
   type pipe
   # Stunnel v3.x
   # command /usr/sbin/stunnel -c -r chico.franken.de:996 -v 3 \
   # -a /var/spool/uucp/certificates
   # Stunnel v4.x
   command /usr/sbin/stunnel4 /etc/uucp/stunnel.conf
   ```

   Wie Du siehst, verwende ich *Stunnel* in Version 4. Willst Du Version 3 verwenden, musst Du die beiden Kommentarzeichen vor der `stunnel`- und der Folgezeile entfernen und eines vor die `stunnel4`-Zeile praktizieren. Warum erkläre ich das eigentlich?

2. In `sys` ersetzt Du
   `port TCP`
   durch
   `port SSL`.

Beachte, dass immer noch die o. g. Rechte für die Dateien gelten müssen. Auch für die `stunnel.conf`, falls Du *Stunnel* in Version 4 verwenden möchtest. Dann brauchst Du allerdings noch den Inhalt der Datei. Bitte schön:

`/etc/uucp/stunnel.conf`

```
client = yes
connect = chico.franken.de:996
CApath = /var/spool/uucp/certificates
CAfile = /var/spool/uucp/certificates/knfca-2003.pem
verify = 3
```

Jetzt kannst Du als Benutzer `uucp` mit

```
uucp@khazad-dum:~ $ uucico -S chico
```

Deine E-Mail sicher und einfach per UUCP abholen. Ich habe ein entsprechendes Alias in `~uucp/.alias` angelegt. Außerdem kannst Du diese Zeile in *cron* packen, damit das Ganze regelmäßig ausgeführt wird.

11.3 Exim 4

Exim (aktuell ist Version 4, die auch in *Sarge* enthalten ist) ist ein enorm mächtiger *Mail Transfer Agent*,* der an der Universität von Cambridge von PHILIP HAZEL entwickelt wurde. Vom selben Autor gibt es geschickterweise auch ein Buch namens “The Exim SMTP Mail Server”, das mir zum Preis von € 60,90 angesichts der Qualität der mitgelieferten Dokumentation etwas zu teuer ist, als dass ich auf den Gedanken käme, es mir zulegen zu wollen.

Bevor ich Dir etwas über die Konfiguration verrate, erkläre ich Dir noch kurz zwei wesentliche Begriffe aus der Welt von *Exim*: Den *Router* und den *Transport*.

Router arbeiten auf E-Mail-Adressen und bestimmen dabei, wie ausgeliefert werden soll – entweder indem ein bestimmter Transport ausgewählt wird oder indem aus der Adresse neue Adressen generiert werden (z. B. aus `/etc/aliases`). Dabei wird eine Adresse der Reihe nach solange einem Router nach dem anderen vorgelegt, bis sie einer akzeptiert oder zurückweist.

* Wie es die Bezeichnung schon ausdrückt, kümmert sich ein MTA darum, E-Mail entgegenzunehmen und irgendwohin weiter zu leiten – sei es ein MTA auf einem anderen Rechner oder ein MDA, ein *Mail Delivery Agent*, der die Mail typischerweise lokal ablegt. Um das begriffliche Sammelsurium zu vervollständigen, sei auch der MUA, der *Mail User Agent*, genannt. Dabei handelt es sich um die Mailclients wie *mutt*, *pine*, *Outlook* etc.

Transports übertragen eine Nachricht vom Spool-Verzeichnis an irgendein Ziel, entweder lokal oder an einen anderen Rechner. Ein Transport implementiert also eine von einem Router angesteuerte Zustellungsmethode.

Du willst doch keine halben Sachen machen, oder? Logo. Also installierst Du den *exim4-daemon-heavy*. In der Installationsphase werden Dir einige Fragen zur Konfiguration gestellt, davon zwei, die ich als Schlüsselfragen empfinde:

1. Du hast die Wahl, ob Du mit einer monolithischen Konfigurationsdatei oder mit lauter kleinen Schnipseln arbeiten möchtest. Implizit hast Du noch eine dritte Wahl – wenn Du diese Möglichkeit nutzen möchtest, ist es egal, womit Du die Frage beantwortest: Du legst Dir selbst eine Konfigurationsdatei namens `/etc/exim4/exim4.conf` an. Dann nimmt *exim4* die statt der automatisch generierten Variante.

   Mir konnte die Standardkonfiguration nicht das bieten, was ich wollte. Und da der Versand von E-Mail eine etwas kitzlige Angelegenheit ist, bei der ich sowieso lieber mir als jemand anderem vertraue, habe ich mich für die dritte Methode entschieden. Dir jedoch empfehle ich zunächst einmal die Schnipselmethode – Updates werden dann prima eingepflegt.

   Übrigens kannst Du Deine Entscheidung jederzeit über den folgenden Aufruf abändern:

   ```
   khazad-dum:~ # dpkg-reconfigure exim4-config
   ```

   Wie funktioniert nun die Erstellung der Konfigurationsdatei, wenn Du Dich für eine der beiden offensichtlichen Möglichkeiten entschieden hast? Im Prinzip ist das ganz einfach. Je nach Deiner Wahl nimmt *update-exim4.conf* entweder die Schnipsel aus dem Verzeichnis `/etc/exim4/conf.d` je Verzeichnis alphabetisch sortiert (erinnert schwer an die Anordnung der `init`-Skripten in `/etc/rc?.d`, nicht wahr?) oder aber `/etc/exim4/exim4.conf.template` und bettet die in der Datei `/etc/exim4/update-exim4.conf.conf` gesetzten Variablen darin ein, indem Zeichenketten wie `DEBCONFfooDEBCONF` durch den Wert von `dc_foo` aus `/etc/exim4/update-exim4.conf.conf` ersetzt werden. Ergebnis des Ganzen ist die Datei `/var/lib/exim4/config.autogenerated`, die – wie es auch der Kommentar am Anfang der Datei besagt – jedes Mal *überschrieben* wird, wenn *update-exim4.conf* aufgerufen wird. Du solltest daher tunlichst darauf verzichten, direkt in dieser Datei zu ändern, und stattdessen auf die Templates zurückgreifen.

   Wenn Du die implizite dritte Möglichkeit wählst und daher fleißig in der Datei `/etc/exim4/exim4.conf` herumeditieren willst, gebe ich Dir noch zwei Tipps mit auf den Weg:

   - Du solltest diese Datei nie bearbeiten, während *exim4* läuft – sobald eine neue *exim4*-Instanz gestartet wird, weil z. B. E-Mail eintrudelt, Du aber die geänderte Konfigurationsdatei schon gespeichert hast, wird bereits die neue Konfigurationsdatei verwendet, während der bereits laufende Prozess noch die alte benutzt. Sei gewarnt – Eisberge voraus!

- Ein ziemlich schmerzfreier Weg, um *exim4* kennen zu lernen und an Deine Bedürfnisse anzupassen, ist es, die Beispiel-Konfiguration aus `/usr/share/doc/exim4-base/examples/example.conf.gz` als Grundgerüst heranzuziehen und um Deine Features zu erweitern, die Du der *exim4*-Spezifikation entnehmen kannst, welche Du in der Datei `/usr/share/doc/exim4-base/spec.txt.gz` findest.

Es bleibt dem Leser als Übung überlassen, meine Konfigurationsdatei geeignet umzusetzen, so dass aus der Schnipselmethode dasselbe generiert wird ...

2. Du wirst nach dem Konfigurationstyp gefragt, der auf Dich am meisten zutrifft. Ich hatte an dieser Stelle ein ernsthaftes Problem, denn davon kam für mich nichts in Frage.

   Was ich erreichen will, ist das Folgende: Intern arbeite ich mit (E)SMTP. E-Mail von Außen kommt bei mir mittels UUCP an. Diese soll auf Spam und Virenbefall untersucht und dann zur Filterung mittels *Sieve* an den *Cyrus IMAPd* zur lokalen Ablage übergeben werden.

   Die meisten Leser dürften ihre E-Mail per IMAP4 oder POP3 bei ihrem Provider abholen und per SMTP einliefern, so dass sie mit der Variante "`mail sent by smarthost; received via SMTP or fetchmail`" glücklich werden müssten. Mails für die lokale Domain werden dann ins Mail-Spool-Verzeichnis gepackt, Mails nach draußen dagegen per SMTP an den Smarthost verschickt. Woher ich das weiß? Ich habe mir die generierte Konfigurationsdatei angesehen. Aber wenn Du nicht überzeugt bist, probieren wir das einfach aus (Achtung, es gibt was zu lernen):

```
khazad-dum:~ # exim4 -bt obiwan@arda.lan
R: system_aliases for obiwan@arda.lan
R: userforward for obiwan@arda.lan
R: procmail for obiwan@arda.lan
R: maildrop for obiwan@arda.lan
R: local_user for obiwan@arda.lan
obiwan@arda.lan
  router = local_user, transport = mail_spool

khazad-dum:~ # exim4 -bt obiwan@foo.arda.lan
...
obiwan@foo.arda.lan
  router = local_user, transport = mail_spool

khazad-dum:~ # exim4 -bt obiwan@meins.de
...
obiwan@meins.de
  router = local_user, transport = mail_spool

khazad-dum:~ # exim4 -bt obiwan@foo.com
R: smarthost for obiwan@foo.com
```

```
obiwan@foo.com cannot be resolved at this time:
  lookup of host "gibtsned.franken.de" failed in
  smarthost router

khazad-dum:~ #
```

Für Dich passt die Smarthost-Variante ja vielleicht, so dass Du wenig weiteren Aufwand haben wirst. Ansonsten ist das zumindest ein leidlich guter Ausgangsstand, von dem aus Du Dir was Eigenes basteln kannst.

Wo Du schon beim Lernen bist, gebe ich Dir noch ein paar weitere Tipps zum Einstieg:

- Wie man herausfindet, welche Router und welche Transports für bestimmte E-Mail-Adressen durchlaufen werden, hast Du gerade schon gesehen (Stichwort Option `-bt`).
- Ebenfalls interessant ist der Test des Rewritings mittels

```
khazad-dum:/etc/exim4 # exim4 -brw obiwan@arda.lan
  sender: obiwan@meins.de
    from: obiwan@meins.de
      to: obiwan@meins.de
      cc: obiwan@meins.de
     bcc: obiwan@meins.de
reply-to: obiwan@meins.de
env-from: obiwan@meins.de
  env-to: obiwan@meins.de
khazad-dum:/etc/exim4 #
```

  mit der Du siehst, was die von Dir angegebenen Rewritings aus verschiedenen Adressen machen.

- Sehr hilfreich ist es, *exim4* mit der Option `-bP` von der Kommandozeile aus aufzurufen. Ohne Argumente werden dann nämlich die Konfigurationsoptionen auf der Shell ausgegeben. Mit Argumenten werden Dir die jeweiligen Werte mitgeteilt:

```
khazad-dum:/etc/exim4 # exim4 -bP configure_file
/etc/exim4/exim4.conf
khazad-dum:/etc/exim4 #
```

  sagt Dir, welche Config-Datei verwendet wird,

```
khazad-dum:/etc/exim4 # exim4 -bP log_file_path
log_file_path = /var/log/exim4/%slog
khazad-dum:/etc/exim4 #
```

verrät Dir, wo die Log-Files liegen. Ein

```
khazad-dum:/etc/exim4 # exim4 -bP +local_domains
domainlist local_domains = @ : localhost : arda.lan : *.arda.lan
: meins.de
khazad-dum:/etc/exim4 #
```

sucht nach der entsprechenden benannten Liste und gibt sie aus. Wenn Du etwas zu Routern oder Transports wissen willst, hilft Dir

```
khazad-dum:/etc/exim4 # exim4 -bP transport cyrus_delivery
no_body_only
current_directory =
debug_print = T: cyrus_delivery for $local_part@$domain
...
khazad-dum:/etc/exim4 #
```

Die Log-Dateien im Verzeichnis `/var/log/exim4` sind sehr ausführlich. Dabei gibt es eine *exim4*-eigene Notation, um Dir zu sagen, was jeweils passiert ist:

- <= E-Mail geht ein
- => E-Mail wird ausgeliefert
- -> E-Mail wird an eine zusätzliche Adresse ausgeliefert
- *> Auslieferung der E-Mail wird unterdrückt
- ** permanenter Zustellungsfehler
- == temporärer Zustellungsfehler

Reichen die Log-Dateien trotzdem nicht aus, so kannst Du *exim4* mit dem Parameter `-d` zur Gesprächigkeit zwingen.

Im Folgenden präsentiere ich Dir meine dokumentierte `/etc/exim4/exim4.conf`-Datei, die ich mir selbst zusammengezimmert habe. Sie tut prima – allerdings fehlt noch der Viren- und Spam-Check. Das wird uns in den folgenden Kapiteln beschäftigen (na gut, Dich weniger – die Klippen habe ich alle schon umschifft).

/etc/exim4/exim4.conf

```
1  # $Id: exim4.conf,v 1.7 2004/12/11 20:00:41 root Exp root $
2
3  ######################################################################
4  #                    MAIN CONFIGURATION SETTINGS                     #
5  ######################################################################
6
```

```
# Konfigurationsverzeichnis
CONFDIR = /etc/exim4

# sollte normalerweise der offizielle FQDN meines Hosts sein -- da der
# aber nur intern ist, reicht es, dass exim4 hier den Namen ermittelt
primary_hostname = meins.de

# Domains, fuer die wir zustaendig sind
domainlist local_domains = @ : localhost : arda.lan : *.arda.lan : \
                           meins.de

# Relay fuer keine weiteren Domains (bin kein MX fuer irgendjemand
# anderen)
domainlist relay_to_domains =

# ueber mich darf das lokale Netz sowie Programme auf mir selbst
# versenden (IPv4 und IPv6)
hostlist   relay_from_hosts = 127.0.0.1 : ::::1 : 192.168.1.0/24

# Access Control List fuer eingehende SMTP-Verbindungen
acl_smtp_rcpt = acl_check_rcpt

# keine unqualifizierten Adressen per SMTP entgegennehmen, daher in
# Kommentar
#sender_unqualified_hosts =
#recipient_unqualified_hosts =

# Domain, die an unqualifizierte Adressen angehaengt werden. Die
# koennen nur lokal vorkommen -- bei SMTP ist das nicht zulaessig.
# Also nehmen wir den Hostnamen und kuemmern uns durch Rewriting
# darum, meine offizielle Domain anzuhaengen
qualify_domain = khazad-dum.arda.lan

# Empfaenger ohne Domainangabe sollen genauso behandelt werden, also
# auf Kommentar
#qualify_recipient =

# an allen Interfaces auf Verbindungswuensche lauern (IPv4 und IPv6)
local_interfaces = 0.0.0.0 : ::::0

# user@[192.168.1.1] soll verboten sein, also auf Kommentar
#allow_domain_literals

# lokale Auslieferung immer ueber Cyrus IMAPd
LOCAL_DELIVERY = cyrus_delivery

```

```
53  # The gecos field in /etc/passwd holds not only the name. see
54  # passwd(5).
55  gecos_pattern = ^([^,:]*)
56  gecos_name = $1
57
58  # niemals Mails direkt an root ausliefern (ein entsprechendes Alias
59  # muss existieren)
60  never_users = root
61
62  # Reverse DNS Lookup fuer alle eingehenden SMTP-Verbindungen
63  # durchführen (kann bei mir -- der Firewall sei Dank -- sowieso nur
64  # intern auftreten, und da ist das billig)
65  host_lookup = *
66
67  # die Abfrage des ident-Service fuer alle eingehenden
68  # SMTP-Verbindungen ist bei mir unnoetig
69  rfc1413_hosts = *
70  rfc1413_query_timeout = 0s
71
72  # Wenn eine Mail weder ausgeliefert noch an den Absender
73  # zurueckgeschickt werden kann, gilt die Mail als "frozen". Sie bleibt
74  # damit fuer immer in der Queue. Das ist natürlich Quatsch fuer mich,
75  # also aendern wir das hier.
76
77  # Unfreeze nach zwei Tagen, noch ein Versuch und dann Fehler
78  # ignorieren
79  ignore_bounce_errors_after = 2d
80  # Frozen Mails nach einer Woche löschen
81  timeout_frozen_after = 7d
82  # postmaster informieren
83  freeze_tell = postmaster
84
85  # User, denen wir vertrauen (damit u. a. das Envelope-From-Setzen
86  # geht)
87  trusted_users = uucp:cyrus
88
89  ##################################################################
90  #                       ACL CONFIGURATION                        #
91  #         Specifies access control lists for incoming SMTP mail  #
92  ##################################################################
93
94  begin acl
95
96  # Wird fuer jedes RCPT-Kommando eingehender SMTP-Verbindungen
97  # verwendet. Die Tests werden sequentiell durchgegangen.
98  acl_check_rcpt:
```

```
99
100     # lokal per SMTP (also nicht per TCP/IP) versendete Mails
101     # akzeptieren, indem nach einem leeren Feld fuer den Sender gesucht
102     # wird
103     accept  hosts = :
104
105     # Mails an lokale Benutzer auf Mails untersuchen, die mit einem .
106     # beginnen oder @, %, !, / oder | enthalten
107     deny    message       = Restricted characters in address
108             domains       = +local_domains
109             local_parts   = ^[.] : ^.*[@%!/|]
110
111     # ausgehende Mails, die mit ., / oder | beginnen oder @, %, ! oder
112     # /../ enthalten.
113     deny    message       = Restricted characters in address
114             domains       = !+local_domains
115             local_parts   = ^[./|] : ^.*[@%!] : ^.*/\\.\\./
116
117     # Mails an meinen postmaster immer annehmen
118     accept  local_parts   = postmaster
119             domains       = +local_domains
120
121     # Wenn der Sender nicht ueberprueft werden kann, dann lassen wir es
122     # sein -- tut auch mit UUCP dank Rewriting
123     # require verify        = sender
124
125     # lokale Adressen annehmen, falls der Empfaenger ueberprueft werden
126     # kann
127     accept  domains       = +local_domains
128             endpass
129             message       = unknown user
130             verify        = recipient
131
132     # Mails an Domains, die wir relayen, annehmen, falls der Empfaenger
133     # ueberprueft werden kann
134     accept  domains       = +relay_to_domains
135             endpass
136             message       = unrouteable address
137             verify        = recipient
138
139     # ab hier ist die Domain weder in +local_domains noch in
140     # +relay_to_domains
141
142     # Mail von Hosts annehmen, fuer die wir relayen
143     accept  hosts         = +relay_from_hosts
144
```

```
  # authentifizierte Hosts akzeptieren
  accept  authenticated = *

  # den Rest ablehnen
  deny    message       = relay not permitted



#############################################################################
#                      ROUTERS CONFIGURATION                                #
#               Specifies how addresses are handled                         #
#############################################################################
#     THE ORDER IN WHICH THE ROUTERS ARE DEFINED IS IMPORTANT!               #
# An address is passed to each router in turn until it is accepted.         #
#############################################################################

begin routers

# UUCP-Router fuer saemtliche ausgehende Mail (quasi ein Smarthost)
uucp_router:
  debug_print = "R: uucp_router for $local_part@$domain"
  driver = accept
  require_files = +/usr/bin/uux
  domains = ! +local_domains
  transport = rsmtp
  no_more

# die verbliebenen Router kuemmern sich um lokale Adressen (daher das
# no_more im UUCP-Router)

# Aliases aufdroeseln
system_aliases:
  debug_print = "R: system_aliases for $local_part@$domain"
  driver = redirect
  domains = +local_domains
  allow_fail
  allow_defer
  data = ${lookup{$local_part}lsearch{/etc/aliases}}
# user = exim
# group = mail
# file_transport = address_file
# pipe_transport = address_pipe
# directory_transport = address_directory

# lokale Auslieferung regeln
local_user:
```

```
  debug_print = "R: local_user for $local_part@$domain"
  driver = accept
  domains = +local_domains
  check_local_user
  local_parts = ! root
  transport = LOCAL_DELIVERY
  cannot_route_message = Unknown user



#############################################################################
#                         TRANSPORTS CONFIGURATION                          #
#############################################################################
#                           ORDER DOES NOT MATTER                           #
#     Only one appropriate transport is called for each delivery.           #
#############################################################################

begin transports

# Auslieferung mittels Cyrus IMAPd
cyrus_delivery:
  debug_print = "T: cyrus_delivery for $local_part@$domain"
  driver = lmtp
  socket = /var/run/cyrus/socket/lmtp
  batch_max = 20
  user = cyrus
  group = mail
  delivery_date_add
  envelope_to_add
  return_path_add

# This transport is used for handling pipe deliveries generated by
# alias or .forward files. If the pipe generates any standard output,
# it is returned to the sender of the message as a delivery error. Set
# return_fail_output instead of return_output if you want this to
# happen only when the pipe fails to complete normally. You can set
# different transports for aliases and forwards if you want to - see
# the references to address_pipe in the routers section above.

address_pipe:
  debug_print = "T: address_pipe for $local_part@$domain"
  driver = pipe
  return_output



# This transport is used for handling deliveries directly to files
```

```
237  # that are generated by aliasing or forwarding.
238
239  address_file:
240    debug_print = "T: address_file for $local_part@$domain"
241    driver = appendfile
242    delivery_date_add
243    envelope_to_add
244    return_path_add
245
246
247  # This transport is used for handling autoreplies generated by the
248  # filtering option of the userforward router.
249
250  address_reply:
251    debug_print = "T: autoreply for $local_part@$domain"
252    driver = autoreply
253
254
255  # batched SMTP fuer UUCP
256  rsmtp:
257    debug_print = "T: rsmtp for $sender_address"
258    driver = pipe
259    command = /usr/bin/uux - -r -a$sender_address -gC chico!rsmtp
260    use_bsmtp
261    return_fail_output
262    user = uucp
263    batch_max = 100
264
265  # Alternative dazu
266  bsmtp:
267    debug_print = "T: bsmtp for $sender_address"
268    driver = pipe
269    command = /usr/bin/uux - -r -a$sender_address -gC chico!bsmtp
270    use_bsmtp
271    return_fail_output
272    user = uucp
273    batch_max = 100
274
275  # rmail fuer UUCP
276  rmail:
277    debug_print = "T: rmail for $pipe_addresses"
278    driver = pipe
279    command = /usr/bin/uux - -r -a$sender_address -gC chico!rmail $pipe_addresses
280    return_fail_output
281    user = uucp
282    batch_max = 20
```

```
283
284
285
286 ##########################################################################
287 #                      RETRY CONFIGURATION                               #
288 ##########################################################################
289
290 begin retry
291
292 # This single retry rule applies to all domains and all errors. It
293 # specifies retries every 15 minutes for 2 hours, then increasing
294 # retry intervals, starting at 1 hour and increasing each time by a
295 # factor of 1.5, up to 16 hours, then retries every 6 hours until 4
296 # days have passed since the first failed delivery.
297
298 # Address or Domain    Error       Retries
299 # -----------------    -----       -------
300
301 *                      *           F,2h,15m; G,16h,1h,1.5; F,4d,6h
302
303
304
305 ##########################################################################
306 #                      REWRITE CONFIGURATION                             #
307 ##########################################################################
308
309 begin rewrite
310
311 # alle lokalen Adressen mit meinem offiziellen Domainnamen maskieren
312 *@*arda.lan               $1@meins.de
313
314 # aus dem Exim-FAQ: Bang-Path-Adressen waehrend des SMTP-Empfangs
315 # korrekt umdrehen (und auch für alle Envelope-Felder), denn
316 # andernfalls funktioniert das Bouncen nicht, falls der User lokal
317 # nicht existiert
318 \N^([^!]+)!(.+)@meins.de$\N  $2@$1  SE
319
320 ##########################################################################
321 #                    AUTHENTICATION CONFIGURATION                        #
322 ##########################################################################
323
324 # There are no authenticator specifications in this default
325 # configuration file.
326
327 begin authenticators
328
```

```
329
330
331  ##########################################################################
332  #                    CONFIGURATION FOR local_scan()                      #
333  ##########################################################################
334
335  # If you have built Exim to include a local_scan() function that
336  # contains tables for private options, you can define those options
337  # here. Remember to uncomment the "begin" line. It is commented by
338  # default because it provokes an error with Exim binaries that are not
339  # built with LOCAL_SCAN_HAS_OPTIONS set in the Local/Makefile.
340
341  # begin local_scan
342
343
344  # End of Exim configuration file
```

Jetzt bleibt nur noch, `/etc/aliases` mit sinnvollen Werten zu besetzen. Der Benutzer `root` wurde schon während der Installation mit einem Alias auf Dich versehen (Du erinnerst Dich sicher daran). Aber vielleicht willst Du auch unter anderen Namen als `obiwan` E-Mail empfangen. Dann musst Du diese in der Datei `/etc/aliases` eintragen. Ich gebe Dir ein Beispiel:

```
ben:              obiwan
obi-wan:          obiwan
obi-wan.kenobi:   obiwan
palm:             obiwan
```

Die ersten drei Einträge sorgen dafür, dass nicht nur `obiwan@meins.de`, sondern auch `ben@meins.de`, `obi-wan@meins.de` und `obi-wan.kenobi@meins.de` verwendet werden können (ebenso natürlich dieselben Namen für die Domain `arda.lan`). Sie werden jeweils in Dein Postfach eingestellt.

Die letzte Zeile ist etwas, worauf ich stehe. Manche Web-Seiten bieten Dienste, die man nur nutzen darf, wenn man seine E-Mail-Adresse und möglicherweise noch ein paar weitere Angaben preisgibt. Weil mich so etwas nervt und ich wissen will, ob die Bande meine Adresse evtl. weiter verkauft, gebe ich immer eine Adresse an, die ich leicht zurückverfolgen kann. Habe ich mich beispielsweise bei *Palm Computing* registriert, verwende ich dafür die E-Mail-Adresse `palm@meins.de`.

Interessanterweise gibt es in `/etc/passwd` Benutzer (z. B. `lp`), die keine Entsprechung in `/etc/aliases` haben. Ich sorge gewohnheitsmäßig dafür, dass jeder Eintrag aus `/etc/passwd`, der keinen menschlichen Benutzer bezeichnet, auch in `/etc/aliases` auf der linken Seite zu finden ist – nicht, dass doch mal eine Mail an so einen Benutzer generiert wird, ohne dass ich das mitbekomme. Für z. B. `lp` steht bei mir daher die Zeile

```
lp:               root
```

in `/etc/aliases`.

Abschließend möchte ich gerne einen „spaßigen“ Effekt an Dich weitergeben, der mich immerhin zwei Stunden lang in Atem gehalten hat: Ich hatte versucht, mit *mutt* eine Mail an mich selbst zu verschicken. Den zugehörigen Ausschnitt aus `/var/log/exim4/mainlog`, allerdings aus Platzgründen ohne Zeitangaben, habe ich Dir im Folgenden abgedruckt:

```
1CcNov-001Nv-TS <= obiwan@meins.de U=obiwan P=local S=1026 \
                   id=20041209125457.GA5318@khazad-dum.arda.lan
1CcNov-001Nv-TS ** --@meins.de: Unknown user
1CcNov-001Nv-TS ** full@meins.de: Unknown user
1CcNov-001Nv-TS ** -r@meins.de <-R@meins.de>: Unknown user
1CcNov-001Nv-TS ** delay@meins.de: Unknown user
1CcNov-001Nv-TS ** failure@meins.de: Unknown user
1CcNov-001Nv-TS *> obiwan <obiwan@meins.de> R=local_user \
                   T=cyrus_delivery
1CcNow-001Ny-04 <= <> R=1CcNov-001Nv-TS U=Debian-exim P=local S=2150
1CcNov-001Nv-TS Completed
```

Diese Meldung verwirrte mich zunächst enorm. Als erstes suchte ich den Fehler natürlich in meiner `/etc/exim4/exim4.conf`, bis ich es irgendwann mit *mail* versuchte und es damit zu meiner Erleichterung einwandfrei klappte:

```
1CcNpR-001O2-K5 <= obiwan@meins.de U=obiwan P=local S=342
1CcNpR-001O2-K5 => obiwan <obiwan@meins.de> R=local_user \
                   T=cyrus_delivery
1CcNpR-001O2-K5 Completed
```

Damit stand fest, dass es nicht an meiner `/etc/exim4/exim4.conf` lag. Woran aber sonst?

Es stellte sich heraus, dass zwei gesetzte Optionen in meiner `~/.muttrc` daran schuld waren. Ich hatte

```
set dsn_notify='failure,delay'
```

und

```
set dsn_return=full
```

darin gesetzt. Mit Sendmail und Postfix (Version 2.0.6) tat das einwandfrei, bloß *exim4* ist darüber gestolpert. Andererseits ist der Satz *“you should not enable this unless you are using Sendmail 8.8.x or greater”* aus dem *mutt*-Manual auch sehr eindeutig – man muss sich nur daran erinnern . . .

11.4 SpamAssassin

SpamAssassin ist ein in *Perl* geschriebener Spam-Filter, der für die Bestimmung des Spam-Levels einer E-Mail verschiedene Kriterien heranzieht, deren Bewertung Du leicht

an Deine Vorlieben anpassen kannst. Du kannst auch die Verwendung des *Bayes*-Algorithmus* aktivieren, ebenso die Nutzung verschiedene Dienste aus dem Internet wie *Vipul's Razor* (ein Netzwerkverbund zur Erkennung von Spam) oder *Pyzor* (dasselbe in grün, aber mit freier Serversoftware). Ein absolutes *must have* also.

Installiere *spamassassin* und ggf. *razor* sowie *pyzor*. Wenn Du anschließend die Datei `/etc/default/spamassassin` anschaust, so verzichte darauf, den Daemon *spamd* zu aktivieren. Wir werden *AMaViS* benutzen, das seinerseits in *Perl* geschrieben ist und somit direkt auf den Kern von *SpamAssassin* zugreift.

Du darfst `/etc/spamassassin/local.cf` bearbeiten, um die Konfiguration an Deine Gegebenheiten anzupassen. Wenn Du darin den Bayes-Algorithmus aktivierst (was Du solltest), so musst Du jeweils mindestens 200 Spam- und Ham-E-Mails anlernen, damit dieser verwendet wird.

/etc/spamassassin/local.cf

```
# Add your own customisations to this file.  See 'man Mail::SpamAssassin::Conf'
# for details of what can be tweaked.
#
# rewrite_subject 0
# report_header 1
use_terse_report 1

# Enable the Bayes system
use_bayes 1

# Enable Bayes auto-learning
auto_learn 1

# Enable or disable network checks
skip_rbl_checks         0
use_dcc                 1
use_pyzor               1
use_razor2              1
razor_config                        /etc/razor/razor-agent.conf

num_check_received 8

# Mail using languages used in these country codes will not be marked
# as being possibly spam in a foreign language.
# - english german
ok_languages            de en

# Mail using locales used in these country codes will not be marked
# as being possibly spam in a foreign language.
ok_locales              de en

# Bayes-Scores hochsetzen
score BAYES_50 5.0
score BAYES_56 5.5
score BAYES_60 6.0
score BAYES_70 7.0
score BAYES_80 8.0
score BAYES_90 9.0
score BAYES_99 10.0

# neues Locking von SpamAssassin 3.0 nutzen
lock_method flock
```

* http://de.wikipedia.org/wiki/Bayesscher_Filter

Das Anlernen machst Du folgendermaßen: Als Benutzer `root` wechselst Du in Dein Spam-Verzeichnis innerhalb der Cyrus-eigenen Ablagehierarchie und rufst dort das Programm auf, das *SpamAssassin* die Unterscheidung von Ham und Spam lehrt (ersetze in diesem Kommando `--spam` durch `--ham` für Ham, der logischerweise in einem anderen Verzeichnis liegt):*

```
khazad-dum:/var/spool/cyrus/mail/o/user/obiwan/spam # sa-learn \
            --showdots --spam --file *.
```

Das erzeugt bzw. aktualisiert Dir die Bayes-Datenbank in `~root/.spamassassin`. Die darin enthaltenen Dateien `bayes_*` musst Du nun nach `~amavis/.spamassassin` kopieren und ggf. die Rechte so anpassen, dass der Benutzer `amavis` – denn unter diesem läuft der *SpamAssassin* – darauf zugreifen darf.

Mehr ist erstmal nicht zu tun damit. Um den Rest kümmert sich *AMaViS*.

11.5 AMaViS

Lass mich kurz den aktuellen Status festhalten: Dein Mailserver tut bereits prima. Per UUCP eingehende Mail wird an den Cyrus verfüttert, ausgehende Mail (entweder lokal erzeugt oder per SMTP aus dem lokalen Netz eingeliefert) per UUCP verschickt. Auch *SpamAssassin* steht schon Gewehr bei Fuß (welch ein Wortspiel). Was noch fehlt, ist der Leim für all das.

Es wäre überaus nett, wenn man Schädlinge schon auf dem Mailserver abfangen könnte. Und das kann man auch. Du könntest dazu *exiscan* verwenden. Das ist ein Patch für *exim4*, der im Paket *exim4* für Debian GNU/Linux bereits enthalten ist. Das hat in meinen Augen jedoch den Nachteil, dass nur *ein* Virenscanner eingesetzt werden kann. Und zwei oder drei sind doch besser, denn die Reaktionszeit auf neue Viren streut in den unterschiedlichen Labors doch sehr. „Aha“, sagt der geneigte Leser nun vielleicht, „dann bastle doch einfach ein Skript, das mehrere Virenscanner aufruft und mit einer entsprechenden Meldung terminiert, falls einer davon einen Virus findet“, was zweifellos eine clevere Idee ist. Nur warum soll ich etwas basteln, das es bereits gibt? *amavisd-new* bietet genau so etwas. Außerdem lässt sich auch der *SpamAssassin* sehr gut darin integrieren (was übrigens mit *exiscan* auch ginge).

Also installierst Du *amavisd-new*. Du wirst feststellen, dass der ziemlich viel per *depends* reinzieht, andererseits recht viel empfiehlt. Ich schlage daher vor, dass Du *aptitude* verwendest, um das Paket einzuspielen. Dabei nimmst Du am besten alle Empfehlungen rekursiv mit. Vergiss nicht, auch *bzip2* einzuspielen. Das fehlt nämlich in der *suggests*-Liste (warum auch immer).

Sobald Du Deine Dateien `/etc/apt/preferences` und `/etc/apt/sources.list` wie in Kapitel 15 auf Seite 123 vorgestellt aufgebohrt hast, solltest Du noch die Pakete *lha* und *unrar* nachinstallieren.

* Auf meinem PIII mit 600 MHz brauche ich 93 min für 17 188 Spam-Mails. Die CPU-Temperatur steigt dabei von 23 °C auf 46 °C.

In der Konfiguration von *clamav-freshclam* gibst Du an, dass Du Deine Virensignaturen per *cron*-Job aktualisieren möchtest.

Nach der Installation sorgst Du mittels *vigr* dafür, dass `clamav` der Gruppe `amavis` angehört. Vergiss nicht, den *clamd* danach neu zu starten, denn ansonsten hat er leider keine Rechte, die von AMaViS abgelegten Dateien zu scannen. Demzufolge mault er und Du wunderst Dich, woran das wohl liegen könnte.

Jetzt brauchst Du noch weitere Virenscanner, beispielsweise *F-Prot*, das Du mittels

```
khazad-dum:~ # apt-get -u install f-prot-installer zip \
                          libhtml-format-perl
```

einspielst. Anschließend sorgst Du dafür, dass Signatur- und Programm-Updates automatisch geholt werden, indem Du die Datei `/etc/cron.d/f-prot-installer` nachbearbeitest. Leider enthält das Paket für Workstations nicht den zugehörigen Daemon. Also musst Du den F-Prot-Abschnitt aus der Liste der Secondary Scanners in den der Primary Scanners hochschieben.

H+BEDV AntiVir UNIX Workstation ist ein anderes nettes Produkt, das Du nichtkommerziell nach Registrierung auch kostenlos nutzen darfst. Das kannst Du Dir von http://www.antivir.de/ holen. Das Archiv packt man irgendwo aus und macht das, was im `README` steht. Ich muss Dir das jetzt nicht vorbeten, oder?

Als nächstes ist die `/etc/exim4/exim4.conf` an der Reihe. Du ergänzt die Datei um die folgenden Angaben:*

```
...
local_interfaces = 0.0.0.0.25 : ::::0 : 0.0.0.0.10025
...
begin routers

# zuerst der Scan mit AMaViS
amavis:
  debug_print = "R: amavis for $local_part@$domain"
  driver = manualroute
# nicht ausfuehren, falls von Port 10025, bereits gescannt oder
# Bounce
  condition = "${if or {{eq {$interface_port}{10025}} \
                        {eq {$received_protocol}{spam-scanned}} \
                        {eq {$sender_address}{}} \
                       }{0}{1}}"
  transport = amavis
  route_list = "* localhost byname"
```

* Beachte bitte, dass ich im UUCP-Router eine Ergänzung vorgenommen habe: Ich sorge dafür, dass der von *AMaViS* eingefügte Header `X-Virus-Scanned` für ausgehende E-Mail entfernt wird. Ich habe gelesen (ich glaube, es war in einer *c't*), dass ein solcher Header vom Empfänger als Garantie für Virenfreiheit von meiner Seite aus aufgefasst werden könne. Sollte doch mal einer von einem infizierten System in meinem Netz verschickt werden, den meine Virenscanner nicht finden (Du weißt: *shit happens*), so könnte man mich daher theoretisch juristisch dafür belangen. Und darauf bin ich nicht sonderlich scharf. Also weg damit.

```
  self = send

# UUCP-Router fuer saemtliche ausgehende Mail (quasi ein Smarthost)
uucp_router:
  debug_print = "R: uucp_router for $local_part@$domain"
  driver = accept
  headers_remove = X-Virus-Scanned
...
begin transports

# Scan mittels AMaViS
amavis:
  driver = smtp
  port = 10024
  allow_localhost
...
```

Jetzt musst Du noch einige Eintragungen in `/etc/amavis/amavisd.conf` abändern. Diese Konfigurationsdatei ist recht garstig, weil sehr umfangreich. Im Folgenden habe ich das Ergebnis eines `rcsdiff amavisd.conf` für Dich abgedruckt.

Unterschiede zu `/etc/amavis/amavisd.conf`

```
5c5
< # $Id: amavisd.conf,v 1.1 2004/12/09 20:32:52 root Exp $
---
> # $Id: amavisd.conf,v 1.3 2004/12/10 09:43:36 root Exp root $
66c66
< $mydomain = 'example.com';      # (no useful default)
---
> $mydomain = 'domain.de';      # (no useful default)
80,81c80,81
< $TEMPBASE = $MYHOME;           # (must be set if other config vars use is)
< #$TEMPBASE = "$MYHOME/tmp";     # prefer to keep home dir /var/amavis clean?
---
> #$TEMPBASE = $MYHOME;           # (must be set if other config vars use is)
> $TEMPBASE = "$MYHOME/tmp";     # prefer to keep home dir /var/amavis clean?
105,106c105,106
< #$forward_method = 'smtp:127.0.0.1:10025';  # where to forward checked mail
< #$notify_method = $forward_method;          # where to submit notifications
---
> $forward_method = 'smtp:127.0.0.1:10025';  # where to forward checked mail
> $notify_method = $forward_method;          # where to submit notifications
161c161
< @bypass_spam_checks_acl  = qw( . );    # No default dependency on spamassassin
---
> #@bypass_spam_checks_acl  = qw( . );    # No default dependency on spamassassin
183,184c183,184
< @local_domains_acl = ( ".$mydomain" );  # $mydomain and its subdomains
< # @local_domains_acl = ( ".$mydomain", "my.other.domain" );
---
> # @local_domains_acl = ( ".$mydomain" );  # $mydomain and its subdomains
> @local_domains_acl = ( ".$mydomain", "domain.de", ".arda.lan" );
278c278
< $DO_SYSLOG = 1;                # (defaults to false)
---
> $DO_SYSLOG = 0;                # (defaults to false)
343c343
```

```
< read_l10n_templates('en_US', '/etc/amavis');
---
> read_l10n_templates('de_DE', '/etc/amavis');
401c401
< $final_spam_destiny       = D_REJECT;  # (defaults to D_REJECT)
---
> $final_spam_destiny       = D_PASS;  # (defaults to D_REJECT)
484a485
> # $spam_admin = "postmaster\@$mydomain";
615c616
< $spam_quarantine_to = 'spam-quarantine';
---
> #$spam_quarantine_to = 'spam-quarantine';
635,636c636,637
< $remove_existing_x_scanned_headers = 0; # leave existing X-Virus-Scanned alone
< #$remove_existing_x_scanned_headers= 1; # remove existing headers
---
> #$remove_existing_x_scanned_headers = 0; # leave existing X-Virus-Scanned alone
> $remove_existing_x_scanned_headers= 1; # remove existing headers
1035c1036,1037
< $MAXLEVELS = 14;                # (default is undef, no limit)
---
> # $MAXLEVELS = 14;                # (default is undef, no limit)
> $MAXLEVELS = 0;                # (default is undef, no limit)
1038c1040,1041
< $MAXFILES = 1500;                # (default is undef, no limit)
---
> # $MAXFILES = 1500;                # (default is undef, no limit)
> $MAXFILES = 0;                # (default is undef, no limit)
1105c1108
< #$sa_auto_whitelist = 1;    # turn on AWL (default: false)
---
> $sa_auto_whitelist = 1;    # turn on AWL (default: false)
1113c1116
< # $sa_auto_whitelist = 1;   # defaults to undef
---
> $sa_auto_whitelist = 1;   # defaults to undef
1120,1121c1123,1124
< $sa_tag_level_deflt  = 4.0; # add spam info headers if at, or above that level
< $sa_tag2_level_deflt = 6.3; # add 'spam detected' headers at that level
---
> $sa_tag_level_deflt  = 0.0; # add spam info headers if at, or above that level
> $sa_tag2_level_deflt = 5.0; # add 'spam detected' headers at that level
1126c1129
< $sa_dsn_cutoff_level = 10;  # spam level beyond which a DSN is not sent,
---
> $sa_dsn_cutoff_level = undef;  # spam level beyond which a DSN is not sent,
1143c1146
< $sa_spam_subject_tag = '***SPAM*** ';        # (defaults to undef, disabled)
---
> # $sa_spam_subject_tag = '***SPAM*** ';        # (defaults to undef, disabled)
1256a1260,1265
>   ### http://www.f-prot.com/
>   ['FRISK F-Prot Antivirus', ['f-prot','f-prot.sh'],
>     '-dumb -archive -packed {}', [0,8], [3,6],
>     qr/Infection: (.+)/ ],
>
>
1452,1454c1461,1463
<   ['FRISK F-Prot Antivirus', ['f-prot','f-prot.sh'],
<     '-dumb -archive -packed {}', [0,8], [3,6],
<     qr/Infection: (.+)/ ],
---
> #  ['FRISK F-Prot Antivirus', ['f-prot','f-prot.sh'],
```

```
100   > #    '-dumb -archive -packed {}', [0,8], [3,6],
101   > #    qr/Infection: (.+)/ ],
```

Jetzt musst Du als Benutzer `amavis` das Verzeichnis `~amavis/.spamassassin` anlegen und dort einen Soft-Link auf die globale *SpamAssassin*-Konfigurationsdatei setzen:

```
khazad-dum:~ # ln -s /etc/spamassassin/local.cf \
                     ~amavis/.spamassassin/user_prefs
```

Der Link ist erforderlich, da nur dann der vom Benutzer `amavis` durchgeführte Spam-Check die global festgelegten Einstellungen verwendet. Es gibt jedoch eine Ausnahme: *AMaViS* ignoriert beharrlich Deinen in der *SpamAssassin*-Konfigurationsdatei festgelegten Wunsch, bei Internetdiensten wie `razor` nachzufragen, außer, Du setzt in der *AMaViS*-Konfigurationsdatei die Variable `sa_local_tests_only` auf `0`.

Ein Tipp noch: Willst Du im laufenden Betrieb Änderungen an der Konfigurationsdatei `/etc/spamassassin/local.cf` durchführen, musst Du *AMaViS* dazu veranlassen, die *SpamAssassin*-Konfiguration erneut einzulesen. Das machst Du wie üblich mittels des Kommandos

```
khazad-dum:~ # /etc/init.d/amavis reload
```

Vielleicht wunderst Du Dich nun ein wenig, warum Du ausgerechnet die *AMaViS*-Konfiguration neu einlesen sollst, wenn Du doch an der *SpamAssassin*-Konfigurationsdatei gefummelt hast. Der Grund dafür ist einfach: *AMaViS* ist in *Perl* geschrieben. *SpamAssassin* besteht aus einigen *Perl*-Modulen und einem Front-End zum Aufruf derselben. Du ahnst es jetzt schon: Die Konfigurationsdatei `/etc/spamassassin/local.cf` wird natürlich genau dann gelesen, wenn die *SpamAssassin*-Module zum ersten Mal geladen werden. Danach verbleibt die Konfiguration im Speicher, bis der zugehörige Prozess beendet und erneut gestartet oder eben ein `reload` angefordert wird.

Nach einem Neustart von *AMaViS* solltest Du in `/var/log/amavis.log` nachprüfen, ob der Start fehlerfrei erfolgte und alle nötigen Programme gefunden wurden. Jetzt kannst Du Dir von `http://www.eicar.org/download/eicar.com` die *EICAR*-Testsignatur herunterladen und an Dich selbst mailen. Du solltest per Mail vor einem Virus gewarnt werden. Wenn das klappt, solltest Du der Reihe nach alle Deine Virenscanner testen, indem Du jeweils alle anderen in `/etc/amavis/amavisd.conf` auskommentierst. Nur so kannst Du sicher gehen, dass das wie gewünscht funktioniert.

Außerdem solltest Du Dir eine Spam-E-Mail schicken und prüfen, ob diese auch korrekt erkannt wird. Dein *Cyrus IMAPd* sollte diese – so Du den *Sieve*-Filter angelegt hast – brav in Deinen Spam-Folder verschieben.

11.6 Fetchmail

Du hast bestimmt mindestens ein Postfach bei GMX, WEB.DE oder sonst wo. Ich habe neben meiner `franken.de`-E-Mail-Adresse – meiner Flatrate wegen – ein Postfach bei

T-Online. Die Mail dort frage ich per POP3 ab. Diesen Vorgang kann man aber auch von einem netten Daemon automatisiert ablaufen lassen: *Fetchmail*.

Erst einmal installierst Du das zugehörige Paket *fetchmail*. Das legt u. a. ein Init-Skript `/etc/init.d/fetchmail` an, welches *fetchmail* als Daemon startet. Dieser Daemon holt per Default alle fünf Minuten Deine Mail ab. Dich wird dabei folgendes interessieren (oder sollte es zumindest): Die Mail auf dem abgefragten Server wird erst dann gelöscht, wenn sie erfolgreich an Deinen lokalen MTA übergeben werden konnte. Wenn der beispielsweise nicht läuft, geht auch keine Mail ins Nirwana. Nett.

Erstelle eine Datei `/etc/fetchmailrc`. Da darin Passwörter im Klartext stehen, solltest Du darauf achten, sie nur für den Benutzer `fetchmail` lesbar zu machen. Bei mir sieht die Datei folgendermaßen aus:

/etc/fetchmailrc

```
1  set daemon 43200 # alle 12 Stunden
2  set postmaster "postmaster"
3  set syslog
4  set bouncemail
5  set no spambounce
6  set properties ""
7  poll pop.t-online.de with proto POP3
8       user 'ow.kenobi' there with password 'mtFbwy!' is 'obiwan' here \
9       options fetchall
```

Wie Du siehst, sind mir die fünf Minuten Poll-Abstand zu kurz. Außer den Vertragsbriefen und einem T-Online-Newsletter geht dort keinerlei Mail ein. Mir reichen daher zwölf Stunden locker aus.

Übrigens gibt es auch ein graphisches Konfigurationsprogramm, das sich im Paket *fetchmailconf* befindet und Dir eine `.fetchmailrc` erzeugt. Auch das leistet Dir gute Dienste, falls Du eine graphische Benutzeroberfläche laufen hast.

Solltest Du mit POP3/S oder IMAP4 bzw. IMAP4/S arbeiten, musst Du das als zu nutzendes Protokoll angeben. Hast Du mehrere Accounts, die Du auf diese Art und Weise abfragen möchtest, brauchst Du für jeden dieser Accounts einen solchen `poll`-Eintrag. Am besten guckst Du mal in die Man-Page.

Nun kannst Du *fetchmail* mit

```
khazad-dum:~ # /etc/init.d/fetchmail start
```

starten. Beim nächsten Systemstart erfolgt das automatisch. Willst Du zwischendurch mal E-Mail holen, kannst Du – ebenfalls als Benutzer `root` – kurz

```
khazad-dum:~ # /etc/init.d/fetchmail awaken
```

aufrufen, und sofort wird gepollt.

KAPITEL 12

Proxy-Dienste

In diesem Kapitel zeige ich Dir, wie Du einen Proxy-Cache und einen Werbefilter einrichten kannst.

12.1 Squid

Je mehr Bandbreite Du hast, desto besser ist es. Aber es ist doch Blödsinn, Bandbreite für den Download von etwas zu opfern, das Du schon mal geholt hast, nicht wahr? Eben. Warum sollst Du ein und dasselbe langweilige Bild von PRINCE CHARLES dreimal runterladen, nur weil Du alle halbe Stunde auf http://www.spiegel.de/ gucken willst? Einmal ist schon zu viel, aber das ist ein anderes Problem.

Also, was tun? Klar, Dein Browser hat einen Cache. Aber Du weißt, dass der Cache nicht greift, sobald Du einen Deiner anderen Browser einsetzt – oder ist Dir noch nie eine Web-Site begegnet, die Deinen Lieblings-Browser ausgrenzt? Ach, Du verwendest den *Internet Explorer*? Dann ist es besonders clever von Dir, den *Squid* und vor allem den *Privoxy* zu installieren, um möglichst viel Unheil von Dir abzuhalten. Internet Explorer, igitt ...

Also, installiere das Teil zuerst mal mittels

```
khazad-dum:~ # apt-get install squid
```

Dann musst Du Dich der Konfigurationsdatei `/etc/squid/squid.conf` annehmen, da standardmäßig nur `localhost` Zugriff auf *Squid* hat.

Die Änderungen, die Du durchzuführen hast, sind minimal. Im Folgenden findest Du meine Änderungen in Form eines *rcsdiff*.

Unterschiede zu `/etc/squid/squid.conf`

```
1  483a484
2  > cache_mem 32 MB
3  1806a1808
4  > acl allowed_hosts src 192.168.1.0/255.255.255.0
5  1863c1865
```

```
< #http_access deny to_localhost
---
> http_access deny to_localhost
1871a1874
> http_access allow allowed_hosts
2061a2065
> cache_mgr obiwan@meins.de
```

Das Teil lauert übrigens auf Port 3128 auf HTTP-Anfragen. Wenn Du das umbiegen oder auch HTTPS aktivieren willst, musst Du das noch nachträglich abändern.

12.2 Privoxy

Der *Privoxy* ist ein besonders nettes Teil. Nicht nur, dass er Werbebanner sehr zuverlässig unterdrückt, sondern er vermag es oftmals auch, bösartigen JavaScript-Code zu entfernen, bevor er Web-Seiten an Deinen Browser ausliefert.

Die Konfiguration geht sehr fix. Im Folgenden findest Du meine Änderungen in Form eines *rcsdiff*.

Unterschiede zu `/etc/privoxy/config`

```
1  ===================================================================
2  RCS file: RCS/config,v
3  retrieving revision 1.1
4  diff -r1.1 config
5  473a474
6  > admin-address obiwan@meins.de
7  661c662,663
8  < listen-address  127.0.0.1:8118
9  ---
10 > #listen-address  127.0.0.1:8118
11 > listen-address  192.168.1.3:8118
12 949a952
13 > forward   /                  localhost:3128
```

Zeile 11 bewirkt, dass Du Deinen *Privoxy* von Deinem lokalen Netz aus über Port 8118 ansprechen kannst (Du setzt in Deinem Browser also `proxy.arda.lan:8118` als Proxy für HTTP ein). Zeile 13 ist dafür gut, Privoxy mit Deinem Squid in Reihe zu schalten. Auch Privoxy nutzt damit den vorgeschalteten Cache von Squid. Krass, nicht wahr? Eben. Das ist LINUX.

Wenn Du selbst weitere Ausblendungen aus Web-Seiten vornehmen möchtest, so bemühe dazu bitte `user.action`.

KAPITEL 13

Datei-Dienste

In diesem Abschnitt lernst Du, Deinen Server zu einem File-Server aufzuwerten.

13.1 NFS

Eingangs erwähnte ich, dass ich meinen Server auch als Fileserver nutzen möchte. Es gibt dazu einige Verfahren – ich nutze allerdings nur zwei davon: Das eine namens *NFS* bespreche ich in diesem Kapitel, während ich auf das andere namens *Samba* in Abschnitt 13.2 auf der nächsten Seite zu sprechen kommen werde.

Das *Network File System*, kurz NFS, ist ein von SUN MICROSYSTEMS entwickeltes Protokoll, das den transparenten Zugriff auf Dateien über das Netzwerk ermöglicht – genau so wie den Zugriff auf Dateien auf Deiner Festplatte. Großer Vorteil: Aufgrund des geringeren Protokolloverheads ist es ein ganzes Stück flotter als Samba.

Die Installation ist sehr einfach: Du brauchst lediglich das Paket *nfs-kernel-server* zu installieren und danach die Datei `/etc/exports` zu bearbeiten. Näheres zum Format dieser Datei findest Du mittels

```
obiwan@khazad-dum:~ $ man 5 exports
```

heraus.

Eine beispielhafte `/etc/exports`, die das Verzeichnis `/export/foo` zum Lesen an das lokale Netz freigibt, sieht folgendermaßen aus:

/etc/exports

```
1 /export/foo   *.arda.lan(async,root_squash,ro)
```

Nachdem Du Deine `/etc/exports` erstellt hast, startest Du NFS mittels

```
khazad-dum:~ # /etc/init.d/nfs-kernel-server restart
```

neu.

Von den Clients aus kannst Du diese Freigabe jetzt unter einem beliebigen Mount-Point (ich nenne ihn im Beispiel mal `/mnt/khazad-dum`) einbinden:

```
khazad-dum:~ # mkdir -p /mnt/khazad-dum
khazad-dum:~ # mount -t nfs khazad-dum:/export/foo /mnt/khazad-dum
khazad-dum:~ # ls /mnt/khazad-dum
...
```

Eine besonders raffinierte Methode ist es, das Einbinden auf Client-Seite mittels Automounter vornehmen zu lassen. Dann werden die Freigaben *on demand* eingebunden. Allerdings ist die clientseitige Konfiguration nicht Gegenstand dieser Anleitung. Aber ich gebe Dir einen Tipp: Auf *Unstable* benötigst Du dazu das Paket *autofs*.

13.2 Samba

Samba implementiert das Server-Message-Protokoll (SMB). Samba leistet damit etwas ähnliches wie ein Windows-Server. Anders als das in Abschnitt 13.1 auf der vorherigen Seite beschriebene NFS bietet Samba auch Druck- und Zeitdienste an.

Spiel einfach *samba* und *samba-doc* ein. Das zieht alles rein, was Du so brauchst. Die Konfiguration beschränkt sich auf ein paar Änderungen in `/etc/samba/smb.conf`. Es schadet aber nicht, wenn Du Dir die umfangreiche Dokumentation zu Gemüte führst.

Da ich nur einen Windows XP-Rechner in meinem Netz habe, nutze ich die verschlüsselten Passwörter von Samba und muss somit mittels

```
khazad-dum:/etc/samba # smbpasswd -a obiwan
```

einen Samba-Nutzer meines Namens samt zugehörigem Passwort anlegen. Das kann ich aber nur dann machen, wenn mein UNIX-Benutzer schon angelegt ist, also UNIX- und Samba-Benutzername *identisch* sind.

Im Folgenden habe ich Dir meine Konfigurationsdatei abgedruckt.

/etc/samba/smb.conf

```
1  [global]
2     workgroup = <Name Deiner Arbeitsgruppe>
3     server string = %h server (Samba %v)
4     dns proxy = no
5     log file = /var/log/samba/log.%m
6     max log size = 1000
7     syslog = 0
8     panic action = /usr/share/samba/panic-action %d
9     security = user
10    encrypt passwords = true
11    passdb backend = tdbsam guest
12    obey pam restrictions = yes
13    invalid users = root
14    passwd program = /usr/bin/passwd %u
```

```
15      passwd chat = *Enter\snew\sUNIX\spassword:* \
16                    %n\n *Retype\snew\sUNIX\spassword:* %n\n .
17      load printers = yes
18      printing = cups
19      printcap name = cups
20      show add printer wizard = no
21      mangled names = no
22      os level = 33
23      time server = Yes
24      unix extensions = Yes
25      map to guest = Bad User
26      socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
27      display charset = iso-8859-15
28      unix charset = iso-8859-15
29
30    [homes]
31      comment = Home Directories
32      browseable = no
33      writable = yes
34      create mask = 0700
35      directory mask = 0700
36
37    [printers]
38      comment = All Printers
39      browseable = no
40      guest ok = yes
41      path = /tmp
42      printable = yes
43      create mode = 0700
44      use client driver = yes
45
46    [print$]
47      comment = Printer Drivers
48      path = /var/lib/samba/printers
49      browseable = yes
50      read only = yes
51      guest ok = yes
52
53    [data]
54      comment = Datenverzeichnis
55      path=/export/data
56      read only = No
57      create mask = 0640
58      directory mask = 0750
59      browseable = Yes
```

KAPITEL 14

Weitere Dienste

Hier folgen einige Dienste, für die mir keine sinnvolle Kategorie eingefallen ist.

14.1 Network Time Protocol

Du hast alle Zeit der Welt. Klar, Mann! Alles andere wäre uncool. Aber es ist ebenso uncool, falsche Zeiten in Logfiles vorzufinden, weil das das Debugging sehr erschweren kann. Mega-uncool ist es, E-Mails zu verschicken, die aus der Zukunft kommen – denn das merkt Dein Empfänger, wenn er den Verlauf der Mail anhand der `Received:`-Header verfolgt. Und Rechneruhren pflegen nun mal mehr, mal weniger genau mit der „offiziellen“ Zeit überein zu stimmen, wenn man nicht automatisch nachführen lässt. Doch genau das ist sehr einfach hinzukriegen, weshalb Du das natürlich machen möchtest.

Aber Vorsicht: Dazu wird immer wieder mal eine Verbindung ins Internet aufgebaut. Wenn Du das nicht willst, ist die hier vorgeschlagene Lösung nichts für Dich![*]

Ansonsten tippst Du die folgende Zauberformel ein:

```
khazad-dum:~ # apt-get -u install ntp ntp-simple ntpdate \
                    ntp-doc ntp-server
```

Wenn Du keinen bestimmten Zeitserver verwenden willst und Deine Firewall UDP-Port 123 passieren lässt, bist Du damit schon fertig (welch angenehme Überraschung).

Da ich über einen in meinen Router integrierten NTP-Server verfüge, möchte ich diesen natürlich auch benutzen. Dazu muss ich in zwei Dateien herumeditieren: In `/etc/default/ntpdate` stelle ich meinen lokalen, also den in den Router integrierten, NTP-Server ein, statt `pool.ntp.org` zu verwenden. Und diesen trage ich auch als Server in die Datei `/etc/ntp.conf` ein. Wundere Dich dabei nicht über den Namen `ntp.arda.lan` – warum mein Router namens `morannon.arda.lan` zusätzlich unter diesem „sprechen-

[*] Das Paket *chrony* ist dann vielleicht besser geeignet. Eine unschöne, aber immerhin gangbare Alternative ist es, bei jedem Verbindungsaufbau ein `ntpdate pool.ntp.org` durchzuführen. Du hast dann keine streng monotone Zeit, aber vielleicht stört Dich das ja nicht.

den Namen“ bekannt ist (und wie man das macht), erläutere ich im Abschnitt 10.1 auf Seite 65.

/etc/default/ntpdate

```
NTPSERVERS="ntp.arda.lan"
NTPOPTIONS="-u"
```

/etc/ntp.conf

```
# /etc/ntp.conf, configuration for ntpd

driftfile /var/lib/ntp/ntp.drift
statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

server ntp.arda.lan

server 127.127.1.0
fudge 127.127.1.0 stratum 13

restrict default kod notrap nomodify nopeer noquery

restrict 127.0.0.1 nomodify
```

Jetzt noch neu starten, dann haben wir es auch schon geschafft. Mittels

```
khazad-dum:~ # ntpq -p
```

kannst Du Dir ansehen, mit wem Du Dich mit welcher Genauigkeit synchronisierst. Es dauert etwas, bis sich das Verfahren eingeschwungen hat. Also nur Geduld.

14.2 CUPS

Um einen Print-Server aufzusetzen, muss im einfachsten Fall ein Drucker direkt an den Rechner angeschlossen sein. Komplizierter sind Netzwerkdrucker – aber auch diese bereiten normalerweise keine Probleme.

Mein Drucker ist ein *HP LaserJet 5MP*, der mir schon seit vielen Jahren treue Dienste leistet. Wo ich so darüber nachdenke, ist das der beste Drucker, den ich je besaß. Zum Anschluss nutze ich die parallele Schnittstelle. Solange Du keinen dieser merkwürdigen GDI-Drucker besitzt, kannst Du unter LINUX die meisten Modelle nutzen, egal, ob sie per paralleler Schnittstelle, USB oder Netzwerk angeschlossen sind.

Fehlt nur noch die nötige Software. Hier kommt *CUPS*[*] ins Spiel. Nachdem es da recht viele *suggested* und *recommended* Pakete gibt, installierst Du *cupsys* mit *aptitude* und nimmst alles mit, was interessant erscheint – also alles bis vielleicht auf die asiatischen *xpdf*-Varianten. Wichtig ist dabei insbesondere, das Paket *foomatic-filters-ppds* mitzunehmen. Dieses enthält eine Druckerdatenbank, die die Anbindung des anzuschließenden Druckers deutlich erleichtert.

Während der Installation wirst Du nach einigem gefragt, beispielsweise nach der Papiergröße, die Du bevorzugst. Ich denke, Du schaffst es, alle Fragen zu beantworten, ohne dass ich das groß kommentieren muss.

Als nächstes machst Du Dich mit einem Editor an der Datei `/etc/cups/cupsd.conf` zu schaffen. Suche darin nach

```
#DefaultLanguage en
```

Dupliziere die Zeile, entferne das Kommentarzeichen (`#`) und ersetze das `en` durch `de`:

```
#DefaultLanguage en
DefaultLanguage de
```

Nach einem Neustart des Drucksystems mittels

```
khazad-dum:~ # /etc/init.d/cupsys restart
```

spricht CUPS deutsch mit Dir.

Möchtest Du, dass andere Rechner, die auch per CUPS drucken (z. B. andere Rechner unter LINUX, Windows XP oder Mac OS X), auf diesen Drucker zugreifen können, musst Du in `/etc/cups/cupsd.conf` (ziemlich am Ende der Datei) den Abschnitt `<Location />` folgendermaßen abändern:

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.1
</Location>
```

Ergänzt Du die vorletzte Zeile auch im Abschnitt `<Location /admin>`, so kannst Du die Konfiguration per Browser auch von anderen Rechnern in Deinem lokalen Netz aus durchführen. Die Benutzerkennung ist übrigens `root`, das Passwort das `root`-Passwort auf dem Server.

Und wenn die anderen Rechner in Deinem lokalen Netz, deren Drucksystem ebenfalls *CUPS* ist, die an Deinen Server angeschlossenen Drucker *automatisch* nutzen sollen, so musst Du die Datei `/etc/cups/cupsd.conf` nochmals mit einem Editor bearbeiten (das letzte Mal in diesem Abschnitt, versprochen!). Suche nach dem Abschnitt, in dem die Direktive

```
BrowseAddress
```

[*] **C**ommon **U**nix **P**rinting **S**ystem

beschrieben ist. Dort fügst Du ein

```
BrowseAddress @LOCAL
```

ein und startest anschließend *CUPS* neu. Mein Mac OS X-System hat den Drucker daraufhin auf Anhieb gefunden und korrekt eingebunden.

Ältere Windows-Versionen wollen dagegen per Samba drucken. Auch das ist kein Problem. Hierzu stellst Du sicher, dass in `/etc/cups/mime.convs` die Zeile

```
application/octet-stream    application/vnd.cups-raw    0    -
```

nicht in Kommentar steht. Selbiges machen wir für die Zeile

```
application/octet-stream
```

in `/etc/cups/mime.types`.* Die Konfiguration von *Samba* hast Du ja bereits vorgenommen; in meiner Konfiguration habe ich weitsichtigerweise bereits dafür gesorgt, dass *CUPS* als Drucksystem genutzt wird. Die unter *CUPS* konfigurierten Drucker werden – zumindest in meiner Konfiguration – im lokalen Netz angeboten. Du kannst darauf drucken, ohne ein Passwort angeben zu müssen (wegen `guest ok = yes` in der `printers`-Sektion).

Nach diesen Vorarbeiten musst Du den Drucker noch konfigurieren. Dazu brauchst Du auf einem Deiner Rechner in Deinem lokalen Netz einen beliebigen Browser, den Du mit dem URL `http://khazad-dum.arda.lan:631/admin` versorgst. Nach der Eingabe Deines `root`-Passworts steht Dir der Administrationsbereich von *CUPS* offen. Dort wählst Du die Funktion `Neuen Drucker hinzufügen` und arbeitest Dich durch die Masken durch, die Dir vorgelegt werden.

Abschließend lässt Du eine Testseite ausgeben. Wenn Du alles richtig gemacht hast, funktioniert das auf Anhieb.

14.3 Lieblingscomics täglich herunterladen

Die meisten Informatiker mögen DILBERT- und GENERAL PROTECTION FAULT-Cartoons, die sie sich täglich zu Gemüte führen. Damit Du nicht täglich die gleichen Seiten anzusurfen brauchst (und Dir nicht zu merken brauchst, wo Du zuletzt zu lesen aufgehört hast), kannst Du Dir das Paket *dailystrips* installieren. Dies lädt die gewünschten Cartoons auf die lokale Maschine herunter und pflegt ein Archiv aller je geholten Comic-Strips.

In Deinem `$HOME`-Verzeichnis legst Du Dir eine Datei `.dailystrips.defs` an, die die Comic-Strips angibt, die Du gerne magst. Die Version in meinem `$HOME` sieht so aus:

beispielhafte `.dailystrips.defs`

```
1 group ObiWan
2     desc ObiWans Lieblings-Comics
3     include harmbengen perscheid quirit weyershausenwelt
4     include nichtlustig calvinandhobbes dilbert garfield gpf
```

* *CUPS* danach sicherheitshalber neu starten.

```
5       include hagarthehorrible peanuts shermanslagoon userfriendly
6       include apodbig
7   end
```

Wenn Du Dich fragst, wie Du Deine Lieblingscomics eintragen kannst, solltest Du Dir die entsprechenden Identifizierer aus `/usr/share/dailystrips/strips.def` heraussuchen.

Abschließend musst Du noch dafür sorgen, dass Deine Comics regelmäßig geholt und lokal abgelegt werden. Dazu benutzt Du zweckmäßigerweise *Cron*. Meinen Aufruf siehst Du im Folgenden:

```
5 10,15  * * *  dailystrips -q --stripnav -l -a --stripdir \
                --basedir /export/dailystrips @ObiWan > /dev/null 2>1
```

Er lädt die Comics in Verzeichnisse herunter, die nach dem Strip benannt sind, erzeugt eine Navigation für die generierten Seiten und pflegt ein Archiv aller je heruntergeladenen Strips. Als Basispfad für das Ganze habe ich `/export/dailystrips` gewählt. Das Verzeichnis muss vor dem erstmaligen Aufruf des Kommandos natürlich existieren und für Dich Schreibrechte gesetzt haben.

14.4 Web-Server

Wenn Du *dailystrips* einsetzt, willst Du Dir Deine Lieblingscomics gerne in Deinem Web-Browser ansehen. Aber das ist nur eine ganz einfache Verwendung eines Web-Servers. Interessantere findest Du sicher auch, wenn Du in dieser Richtung etwas vorhast. Digitale Fotoalben lassen sich so beispielsweise vortrefflich vorhalten.

Nachdem Du die Debian-Pakete *apache2-mpm-worker* und – so Du gerne Dokumentation liest – *apache2-doc* installiert hast, führst Du einige kleine Anpassungen durch, um Deine Comic-Strips per Web-Browser lesen zu können.

Ich möchte in meinem Web-Browser den URL `http://www.arda.lan:4711/` eingeben und dann auf der aktuellen Comicseite abgeworfen werden. Dazu ergänze ich in meiner `/etc/bind/db.arda.lan` zunächst den CNAME `www`, den ich auf `khazad-dum` zeigen lasse*. Jetzt musst Du die Datei `/etc/apache2/ports.conf` um eine Zeile mit Inhalt

```
Listen 4711
```

ergänzen, damit der *Apache2* auch auf dem gewünschten Port auf Verbindungen lauert. Abschließend erstellst Du eine Datei beliebigen Namens im Verzeichnis `/etc/apache2/sites-available`, die ich `4711_dailystrips` genannt habe, damit ich sie leicht zuordnen kann.

`/etc/apache2/sites-available/4711_dailystrips`

```
1   NameVirtualHost *:4711
2   <VirtualHost *:4711>
```

* Vergiss nicht, die Serial Number zu erhöhen und den Nameserver mittels `rndc reload` zum erneuten Einlesen seiner Konfiguration zu veranlassen.

```
3           ServerAdmin obiwan@arda.lan
4
5           DocumentRoot /export/dailystrips
6           <Directory /export/dailystrips/>
7                   Options Indexes FollowSymLinks MultiViews
8                   AllowOverride None
9                   Order allow,deny
10                  allow from all
11          </Directory>
12
13          ErrorLog /var/log/apache2/error.log
14          LogLevel warn
15          CustomLog /var/log/apache2/access.log combined
16          ServerSignature On
17  </VirtualHost>
```

Bevor Du *Apache2* neu startest, legst Du noch einen Soft-Link an, damit Deine neue Konfiguration überhaupt verwendet wird (darum ist es eine prima Idee, die in den Paketen enthaltene Dokumentation, zumindest aber das jeweilige `README.Debian`, auch zu lesen):

```
khazad-dum:~ # ln -s /etc/apache2/sites-available/4711_dailystrips \
                     /etc/apache2/sites-enabled/4711_dailystrips
```

14.5 Router-Überwachung mit SNMP

Bisher habe ich es Dir gegenüber als das Beste überhaupt dargestellt, den Internetzugang nicht vom Server abwickeln zu lassen, sondern von einem dedizierten Router mit integrierter Firewall. Dazu stehe ich auch nach wie vor. Allerdings stellt Dich diese Netzwerkarchitektur vor ein gewisses Dilemma: Denn sobald sowieso eine Internetverbindung aufgebaut wird, soll der Server geschickterweise gleich E-Mail holen und in der Warteschlange befindliche verschicken. Nur – wie bekommst Du den Verbindungsaufbau mit, wenn der Server das nicht selbst handhabt?

Wenn Du einen Router der Baumarkt-Klasse hast, dann hast Du dafür schlechte Karten. Wenn Du dagegen einen Router der gehobenen Klasse Dein Eigen nennen darfst, bist Du fein raus, denn der „spricht“ normalerweise *SNMP*[*].

Man nutzt SNMP u. a. dafür, den Status von Netzwerkgeräten abzufragen. Und genau das gibt Dir ein Instrument zur Hand, Verbindungsauf- und -abbauten festzustellen (und einigen anderen Schabernack zu treiben, auf den ich hier allerdings nicht einzugehen gedenke).

Ich beispielsweise lasse mir von meinem Router einen SNMP-Trap schicken, sobald dieser eine Verbindung ins Internet aufbaut. Bei dieser Gelegenheit lasse ich per *UUCP* E-Mail holen bzw. verschicken und sorge dafür, dass der für mich zuständige *DynDNS*-

[*] **S**imple **N**etwork **M**anagement **P**rotocol

Server meine aktuelle IP-Adresse erfährt. Letzteres kann der Router bei den üblichen DynDNS-Anbietern auch selbst; allerdings unterstützt er nicht das Verfahren, das der KNF e. V.* verwendet, nämlich ein Login per *ssh*.

Beim Verbindungsabbau lasse ich mir ebenfalls einen Trap schicken und kann so protokollieren, wie lange ich online war. Interessant für die Statistik.

Falls Dein Router keine Traps zu schicken vermag, aber grundsätzlich SNMP beherrscht, musst Du ggf. pollen, also regelmäßig ein *snmpget* aufrufen, um den entsprechenden Wert auf Änderungen zu überprüfen. Unschön, geht aber auch.

Am besten wird es sein, Du guckst in die Datenblätter oder einfach in das Konfigurationsmenü Deines Routers und siehst nach, ob und inwieweit er SNMP beherrscht. Dann aktivierst Du die SNMP-Funktionalität (*Achtung: Mach das in Deiner Firewallkonfiguration von außen her dicht!*). Bei der Gelegenheit stellst Du das Ganze gleich so ein, dass die Traps an Deinen Server geschickt werden.

Dann installierst Du die Pakete *snmp*, *snmpd* und *snmptrapfmt*. *snmpd* enthält unter anderem den *snmptrapd*, der eingehende Traps auffängt und über einen Trap-Handler protokolliert. Per Default ist dort der *snmptrapfmt* eingetragen. Du findest dessen Protokolldatei unter `/var/log/snmptrapfmt.log`. Darin siehst Du leicht, welche Traps für Dich interessant sind.† Für diese musst Du Deinen eigenen Trap-Handler schreiben und in `/etc/snmp/snmptrapd.conf` eintragen. Ich habe eine Zeile ähnlich der folgenden vor der `default`-Zeile eingefügt:

```
traphandle SNMPv2-SMI::enterprises.5001.815.4.1533.1.27 \
           /root/bin/ConnectionHandler
```

Da Du einen anderen Router als ich hast (und die obige Zeile nicht mal zu meinem Router passt, denn Du brauchst nicht zu wissen, welchen ich einsetze), nutzt Dir diese Zeile nur insofern etwas, als die Syntax klar wird.

Das darin verwendete Skript `ConnectionHandler` wiederum muss die eingehenden Traps genauer untersuchen und feststellen, ob es sich um einen Verbindungsauf- oder -abbau oder etwas ganz anderes handelt.

Mein Skript, allerdings mit abgefälschten OIDs, habe ich Dir im Folgenden abgedruckt (eine elegantere Version bleibt dem Leser zu Übungszwecken überlassen):

/root/bin/ConnectionHandler

```
1  #!/bin/sh
2  # Skript, das von smtptrapd aufgerufen werden kann
3
4  # Stellt fest, ob ein Verbindungsauf- oder -abbau stattgefunden hat,
```

* http://www.franken.de

† Wenn Du das nicht so leicht sehen kannst, musst Du die in *snmp* enthaltenen Programme *snmpwalk* und *snmpget* einsetzen, um Land zu sehen. Ein guter Ausgangspunkt dafür ist das Kommando `snmpwalk -Cc -c public -v 1 morannon`. Allerdings ist die für Dich interessante Information wahrscheinlich im herstellerspezifischen Zweig versteckt, so dass Du als Startknoten noch eine entsprechende OID dahinter schreiben musst, die Du Dir allerdings aus dem Trap-Log erschließen kannst. Ach so: Eine vom Hersteller Deines Routers mitgelieferte *MIB* ist dabei unheimlich hilfreich.

```
# protokolliert und ruft im Fall des Verbindungsaufbaus ein Skript auf.

read host
read ip
# 1: Verbindungsaufbau
# 2: Verbindungsabbau
action=0
connection=
connTime=

while read oid val
do
# State überprüfen, ob ein Trap zum Verbindungsauf- oder -abbau kam
  if [ "$oid" = "SNMPv2-SMI::enterprises.5001.815.4.1533.1.27.1.4.3" ]
  then
  # Verbindungsaufbau
  if [ "$val" = "foo" ]
  then
    action=1
  else
    # Verbindungsabbau
    if [ "$val" = "bar" ]
    then
      action=2
    fi
  fi
  fi
# Name der Verbindung auslesen
  if [ "$oid" = "SNMPv2-SMI::enterprises.5001.815.4.1533.1.27.1.11.3" ]
  then
    connection=$val
  fi
# Verbindungszeit auslesen
  if [ "$oid" = "SNMPv2-SMI::enterprises.5001.815.4.1533.1.27.1.12.3" ]
  then
    connTime=$val
  fi
done

if [ $action -eq 1 ]
then
  /usr/bin/logger -i -t ConnectionHandler \
    "Verbindungsaufbau über $connection"
  /root/bin/ConnUp
else
  if [ $action -eq 2 ]
```

```
51    then
52      /usr/bin/logger  -i -t ConnectionHandler \
53        "Verbindungsabbau von $connection (verbunden: $connTime Sek.)"
54    fi
55  fi
```

Wie Du siehst, wird darin ein `ConnUp`-Skript aufgerufen. Dieses erledigt die eigentliche Arbeit wie den *UUCP*-Aufruf und die Eintragung in den DynDNS. Aber das bekommst Du auch ohne meine Hilfe hin, so ein einfaches Skript zu schreiben. Ansonsten laufen bestimmt irgendwo im Fernsehen die Teletubbies …

Teil IV

Verwaltung des Systems

KAPITEL 15

Aktualisierung des Systems

Gerade dann, wenn Dein System von außen erreichbar ist, ist es wichtig, dass Du die Software auf Deiner Maschine aktuell hältst. Nur dann ist gewährleistet, dass bekannte Sicherheitslücken so rasch wie möglich gestopft werden. Unter *Testing* und *Unstable* sorgen Aktualisierungen nicht nur für die Beseitigung von allerlei Fehlern, sondern bringen oft auch neue Features mit. Du siehst, ein aktuelles System ist eine tolle Sache.

Wenn Du fragst, wie oft Du aktualisieren sollst, muss ich Dir sagen, dass das Geschmackssache ist. In *Testing* und *Unstable* bekommst Du so ziemlich jeden Tag einen ganzen Schwall an Updates geliefert. In *Stable* sollte sich deutlich weniger tun. Entscheide selbst ...

Aber zuerst musst Du Dein *APT* vernünftig konfigurieren. Da ich *Stable* verwende, aber auch in der Lage sein möchte, einzelne Pakete aus *Testing* oder *Unstable* einzuspielen, verwende ich *APT-Pinning*. Man weist damit einzelnen *Paketen* oder, wie in meinem Beispiel, ganzen *Zweigen* unterschiedliche Prioritäten zu. Diese Prioritäten sorgen dafür, dass Du genau beeinflussen kannst, welche Pakete in Dein System eingespielt werden: Es werden immer die Pakete genommen, die die höchste Priorität haben – selbst dann, wenn es neuere Pakete gibt, die jedoch mit niedrigerer Priorität gelistet sind. Gibt es also ein Paket sowohl in *Stable* als auch in *Unstable*, so kannst Du festlegen, dass immer das aus *Stable* genommen wird. Gibt es das Paket dagegen nur in *Unstable*, so wird dieses verwendet. Dazu gebe ich Dir noch ein Beispiel an, sobald ich Dir meine `/etc/apt/sources.list` vorgestellt habe.

Für das *Pinning* in der gerade von mir vorgestellten Form erzeugst Du eine Datei `/etc/apt/preferences` folgenden Inhalts:

```
Package: *
Pin: release a=stable
Pin-Priority: 900

Package: *
Pin: release a=sarge
Pin-Priority: 900

Package: *
```

```
Pin: release a=testing
Pin-Priority: 800

Package: *
Pin: release a=unstable
Pin-Priority: 700

Package: *
Pin: release o=Debian
Pin-Priority: -10
```

Kurz zur Erklärung (mehr findest Du mittels

```
obiwan@khazad-dum:~ $ man apt_preferences
```

heraus): Der letzte Block verhindert durch die extrem niedrige Priorität von -10, dass Pakete aus anderen Debian-Distributionen eingespielt werden. Der erste Block weist dem *Stable*-Zweig eine sehr hohe Priorität von 900 zu. Der zweite bezieht sich auf *Volatile*[*], das Updates von solchen *Stable*-Paketen anbietet, die sich häufig ändern (Virenscanner, Spam-Filter etc.), und vergibt eine ebenso hohe Priorität wie für *Stable*. Dadurch werden bevorzugt Pakete von *Volatile* verwendet. Die restlichen Blöcke vergeben an *Testing* und *Unstable* niedrigere Prioritäten, so dass diese nur dann verwendet werden, wenn es keine Kandidaten höherer Priorität gibt.

So kann ich meine `/etc/apt/sources.list` ohne weiteres auch mit Quellen[†] für *Testing* und *Unstable* bestücken; standardmäßig werden jedoch *Stable* und *Volatile* verwendet.

Vielleicht sollte ich noch ein paar Worte zum Aufbau der `/etc/apt/sources.list` verlieren, auch wenn Du Dir das leicht selbst aneignen könntest. Aber ich will mal nicht so sein. Jeder Eintrag hat entweder die Form

```
deb [URI] [Distribution] [Komponenten]
```

oder

```
deb-src [URI] [Distribution] [Komponenten]
```

Letztere bezeichnet Quelldateien, Sourcen, jedoch keine bereits fertig paketierten Binaries. Außerdem kennzeichnen mit '#' eingeleitete Zeilen – wie üblich – einen Kommentar. Aber ich muss ja die Man-Page zu `sources.list` nicht für Dich abschreiben. Lies ruhig selbst ein bisschen nach. Nur soviel: Neben den Zugriffsmethoden `http`, `ftp` und `cdrom`[‡] kannst Du u. a. auch `file` und `ssh` verwenden.

---

[*] siehe http://volatile.debian.net

[†] Eine Quelle im Zusammenhang mit *APT* meint keine Source-Codes, sondern Orte, von denen man Debian-Pakete herunterladen kann. *Quellpakete* sind das, was Du zuerst vermutet hast: Debian-Pakete mit den Sourcen, aus denen die sonst üblichen Binärpakete erst noch erzeugt werden müssen.

[‡] Du willst *apt-cdrom* nutzen, um solche Zeilen hinzuzufügen. Auch dann, wenn Du ein DVD-ROM-Laufwerk einsetzt.

Damit das Ganze jetzt nicht so leer im Raum steht, habe ich Dir im Folgenden meine `/etc/apt/sources.list` abgedruckt:

/etc/apt/sources.list

```
# Stable
deb     ftp://ftp.de.debian.org/debian        stable   main contrib non-free
deb-src ftp://ftp.de.debian.org/debian        stable   main contrib non-free

# Volatile
deb     http://volatile.debian.net/debian-volatile   sarge/volatile   main
deb-src http://volatile.debian.net/debian-volatile   sarge/volatile   main

# Testing
deb     ftp://ftp.de.debian.org/debian        testing   main contrib non-free
deb-src ftp://ftp.de.debian.org/debian        testing   main contrib non-free

# Unstable
deb     ftp://ftp.de.debian.org/debian        unstable   main contrib non-free
deb-src ftp://ftp.de.debian.org/debian        unstable   main contrib non-free

# Security
deb     http://security.debian.org   stable/updates   main contrib non-free
deb-src http://security.debian.org   stable/updates   main contrib non-free
#deb     http://security.debian.org   testing/updates   main contrib non-free
#deb-src http://security.debian.org   testing/updates   main contrib non-free
#deb     http://security.debian.org   unstable/updates   main contrib non-free
#deb-src http://security.debian.org   unstable/updates   main contrib non-free

# DVD-Spieler etc.
deb     ftp://ftp.nerim.net/debian-marillat   stable   main
deb     ftp://ftp.nerim.net/debian-marillat   testing   main
deb     ftp://ftp.nerim.net/debian-marillat   unstable   main

```

Als nächstes bringst Du Deine Paketliste auf den aktuellen Stand:

```
khazad-dum:~ # apt-get update
```

Damit hast Du das Rüstzeug, den Erfolg des APT-Pinning zu überprüfen. Um gleich den kompliziertesten und damit interessantesten Fall auszuprobieren, schauen wir uns *clamav* an:

```
obiwan@khazad-dum:~ $ apt-cache policy clamav
clamav:
  Installiert:0.85.1-0volatile1
  Mögliche Pakete:0.85.1-0volatile1
  Versions-Tabelle:
     0.85.1-2 0
        800 ftp://ftp.de.debian.org testing/main Packages
        700 ftp://ftp.de.debian.org unstable/main Packages
 *** 0.85.1-0volatile1 0
        900 http://volatile.debian.net sarge/volatile/main Packages
        100 /var/lib/dpkg/status
     0.84-2 0
        900 ftp://ftp.de.debian.org stable/main Packages

obiwan@khazad-dum:~ $
```

Du siehst: Das gibt es mit drei unterschiedlichen Versionsnummern:

- 0.85.1-2 (*Unstable*)
- 0.85.1-0volatile1 (*Volatile*)
- 0.84-2 (*Stable*)

Das Paket aus *Unstable* hat die höchste Versionsnummer. Allerdings hat es zugleich die niedrigste Priorität der Kandidaten, weshalb es *nicht* gewählt wird. Die höchste Priorität haben mit 900 die Kandidaten aus *Stable* und aus *Volatile*. Da die höchste Versionsnummer in *Volatile* vorliegt, wird das Paket aus diesem Zweig genommen. Diese Entscheidung kannst Du übrigens immer in der Zeile

```
Mögliche Pakete:
```

ablesen.

Nachdem Du Dich einmalig vom korrekten Verhalten des Pinning überzeugt hast, aktualisierst Du alle Pakete mit einem beherzten

```
khazad-dum:~ # apt-get -u upgrade
```

Das zeigt Dir zunächst an, welche Pakete aktualisiert werden sollen, was Du bestätigen musst.* Wenn Du die Option `-u` weglässt, entfällt die Bestätigung. Dann werden die Updates heruntergeladen. Hast Du *apt-listbugs* installiert (was Du solltest), werden Dir derzeit bekannte Probleme mit den einzuspielenden Paketen angezeigt, die Du abnicken kannst, worauf die Installation weiter läuft, die Du aber auch verhindern kannst.

Wenn Du einmalig alles korrekt eingerichtet hast, beschränken sich alle weiteren Aktualisierungsläufe auf den Aufruf der beiden Kommandos

```
khazad-dum:~ # apt-get update
...
khazad-dum:~ # apt-get -u upgrade
```

* Dabei kann es sein, dass ein paar Pakete als „zurückgehalten“ oder “kept back” gekennzeichnet werden. Das braucht Dich nicht zu stören: Diese Pakete können deshalb nicht aktualisiert werden, weil dazu die Neuinstallation weiterer Pakete erforderlich wäre. Willst Du das gestatten, so gib nach dem derzeit laufenden Update ein `apt-get -u dist-upgrade` ein.

KAPITEL 16

Systemüberwachung

Von großer Bedeutung ist das *Monitoring* Deines Systems. Während manche Software, etwa die *smartmontools*, Dir eine E-Mail schicken, wenn etwas im Argen ist, protokollieren andere Programme Probleme nur in die Logdateien Deines Rechners. Dieses Kapitel wird sich damit befassen, letztere automatisiert auszuwerten.

Ein anderes Thema dieses Kapitels wird die Integrität Deines Servers sein – Du willst schließlich wissen, wenn jemand in unlauterer Absicht in Deinen Rechner eingedrungen ist, um Chaos und Verderben zu säen.

16.1 Überwachung der Logdateien

Die Überwachung von Logdateien ist langweilig. Der Mensch ist nicht dafür gemacht, sich Zahlen- und Textwüsten anzuschauen. Punkt. Allerdings stehen manchmal verdammt interessante Ereignisse im Systemprotokoll. Gesucht ist also ein Mittel, nur genau diese Ereignisse an Dich weiterzugeben, den Rest dagegen wegzufiltern – das berühmte Silbertablett also, das Dir Dein persönlicher James bei Bedarf vor das Näschen hält. Ach so, Dein James nennt sich in dem Fall *Logcheck*, weshalb Du das zugehörige Paket *logcheck* auch gleich installieren solltest.

Ohne weitere Konfiguration durch Dich wird `root` jetzt alle zwei Stunden per E-Mail mit wichtigen Systemereignissen konfrontiert. Möchtest Du darüber hinaus eine kurze Zusammenfassung der Inhalte der Protokolldateien, so kannst Du `SYSLOGSUMMARY=1` setzen und das Paket *syslog-summary* installieren. Aber lass es – es kommt meiner Erfahrung nach nichts Wesentliches dabei heraus.

Benutze das Kommando

```
khazad-dum:/ # su -s /bin/bash -c "/usr/sbin/logcheck" logcheck
```

um von Hand einen Check zu initiieren. Das Ergebnis des Laufs wird an `root` und damit an `obiwan` gemailt, wenn Du zugestimmt hast, dass E-Mail an `root` zusätzlich an diesen Benutzer ausgeliefert wird. Dabei kannst Du drei Arten von E-Mails bekommen:

1. *Attack Alerts* betreffen Einbruchsversuche in Deine Maschine. Diese werden durch Regeln im Verzeichnis `/etc/logcheck/cracking.d` spezifiziert. Willst Du davon

welche ignorieren, so musst Du das in der Konfigurationsdatei `/etc/logcheck/logcheck.conf` explizit so einstellen und dann eine entsprechende Regel in `/etc/logcheck/cracking.ignore.d` eintragen, falls die nicht schon darin steht – schau insbesondere dann nach, falls Du in der Konfigurationsdatei das Flag zum Ignorieren bisher abgeschaltet hattest.

2. *Security Events* behandeln weniger kritische Ereignisse, die man aber trotzdem beachten sollte. Diese werden in `/etc/logcheck/violations.d` erfasst, während `/etc/logcheck/violations.ignore.d` dazu dient, bestimmte davon zu ignorieren.

3. *System Events* schließlich befassen sich mit dem ganzen Rest. Und hier landet all das, was nicht schon bisher behandelt worden ist. Um hier etwas zu ignorieren, kommt der in `/etc/logcheck/logcheck.conf` zu definierende `REPORTLEVEL` ins Spiel. Dieser arbeitet folgendermaßen: Erst werden die regulären Ausdrücke in der Datei `/etc/logcheck/ignore.d.paranoid` herangezogen, um etwas herauszufiltern. Anschließend werden die Muster in `/etc/logcheck/ignore.d.server` verwendet, falls Dein `REPORTLEVEL` entweder auf `server` oder `workstation` gesetzt ist. Falls Dein `REPORTLEVEL` `workstation` lautet, wird zum Schluss `/etc/logcheck/ignore.d.workstation` angewandt. Es werden also erst die speziellen Meldungen gefiltert und dann immer weiter abgeschwächt. Der `REPORTLEVEL` `server` erscheint mir recht vernünftig.

Leider wird noch etwas Handarbeit nötig sein, um Ereignisse auszuschließen, die Dich nicht weiter stören. Ich habe je eine Datei `mkspecial` in `/etc/logcheck/violations.ignore.d` und `/etc/logcheck/ignore.d.server` angelegt und alles für mich Uninteressante darin platziert.

Aber Vorsicht: Eine leere Zeile in Deiner Datei passt als Muster auf *alles* – das ist fatal, wenn die Datei hergezogen wird, um Einträge aus der Ergebnisliste auszufiltern. Die ist hinterher nämlich leer, und Du bekommst gar nix gemailt. Nicht der Sinn der Sache, würde ich meinen ...

16.2 Intrusion Detection Systems

Da meine Maschine nicht direkt am Internet hängt, sondern durch einen Router mit Firewall davor abgeschirmt wird, und auch keine Dienste ins Internet hin bereitstellt, brauche ich nicht unbedingt ein *IDS*, um eventuelle Einbrüche in mein System festzustellen.

Nichtsdestotrotz schadet es nicht, ein solches Tool laufen zu haben. Ich verwende dafür traditionell *AIDE*. Einige weitere findest Du per

```
obiwan@khazad-dum:~ $ apt-cache search intrusion detection
```

bzw.

```
obiwan@khazad-dum:~ $ apt-cache search integrity checker
```

AIDE errechnet unter anderem eine Prüfsumme über alle Dateien auf Deinem System. Wenn sich eine davon ändert (beispielsweise, weil ein Einbrecher Dateien modifiziert), erkennt es das.

Während der Konfigurationsphase der Installationsprozedur wirst Du gefragt, ob Du die Datenbank initialisieren möchtest. Wenn Du das bejahst, läuft dieser Vorgang im Hintergrund ab.

Wird zuviel geloggt, kannst Du entweder die Datei `/etc/aide/aide.conf` bearbeiten oder versuchen, mit `/etc/default/aide` zurechtzukommen. Die Vorgaben sind eigentlich prima.

Die erstellte Datenbank solltest Du, nachdem sie sich stabilisiert hat (anfangs vergisst Du garantiert das eine oder andere sich häufig ändernde Verzeichnis), auf ein *read-only*-Medium packen. Sonst könnte ein Einbrecher nach seinen Modifikationen einfach wieder ein

```
khazad-dum:~ # aide --init
```

machen, und das Ding würde nie Alarm schlagen ...

Um Änderungen im Dateisystem in Deine Datenbank zu übernehmen, musst Du ein

```
khazad-dum:~ # aide --update
```

laufen lassen und im Verzeichnis `/var/lib/aide` die Datei `aide.db.new` nach `aide.db` kopieren, wenn Du Dich von der Korrektheit der Änderungen überzeugt hast. Per Default übernimmt das Update ein täglich einmal ablaufender *cron*-Job für Dich.

16.3 Rootkit Detection

Eine nette Methode, um festzustellen, ob Dein System „sauber“ ist, besteht darin, ein Programm laufen zu lassen, das die üblichen *Rootkits* erkennt. Unter Rootkits versteht man Programmsammlungen, die einen einzigen Zweck haben: Den unbemerkten Zugriff eines Fremden auf Dein System zu gestatten und dabei Eingaben, E-Mails etc. von Dir mitzuschneiden.

Ein solches Programm ist *Rootkit Hunter*. Geschickterweise gibt es davon auch ein Debian-Paket, das sich derzeit jedoch noch im *Unstable*-Zweig versteckt. Wenn Du jedoch die in Kapitel 15 auf Seite 123 vorgeschlagene Konfiguration zum Apt-Pinning auf Deinem System vorgenommen hast, müsstest Du das Paket problemlos auf Deinem System einspielen können.

Gib also

```
khazad-dum:~ # apt-get install rkhunter
```

ein.

Anschließend bearbeitest Du die Datei `/etc/default/rkhunter` so, dass der *Rootkit Hunter* einmal täglich seinen Suchlauf durchführt und einmal wöchentlich nach Updates für seine Datenbanken Ausschau hält. Dieses Update solltest Du gleich zu Beginn händisch ausführen, indem Du folgendes eingibst:

```
khazad-dum:~ # rkhunter --update
```

```
Running updater...

Mirrorfile /var/lib/rkhunter/db/mirrors.dat rotated
Using mirror http://www.rootkit.nl/rkhunter
[DB] Mirror file                      : Up to date
[DB] MD5 hashes system binaries       : Update available
  Action: Database updated
[DB] Operating System information     : Update available
  Action: Database updated
[DB] MD5 blacklisted tools/binaries   : Up to date
[DB] Known good program versions      : Update available
  Action: Database updated
[DB] Known bad program versions       : Update available
  Action: Database updated
Ready.
khazad-dum:~ #
```

Nun bietet es sich an, das System auf der Stelle interaktiv zu untersuchen:

```
khazad-dum:~ # rkhunter -c
Rootkit Hunter 1.2.7 is running
Determining OS... Ready

Checking binaries
* Selftests
     Strings (command)                [ OK ]

* System tools
  Performing 'known bad' check...
   /bin/cat                           [ OK ]
...
khazad-dum:~ #
```

Nach jedem Test musst Du die Enter-Taste drücken, um die Ausgabe fortzusetzen. Wenn Dich das stört, kannst Du das mittels der Option `--skip-keypress` natürlich unterdrücken.

Wenn Du *udev* oder *Java* laufen hast, wird Dich der Test auf versteckte Verzeichnisse hinweisen. Um diese *false positives* zu unterdrücken, solltest Du folgende Zeilen in die Konfigurationsdatei `/etc/rkhunter.conf` einfügen:

```
ALLOWHIDDENDIR=/etc/.java
ALLOWHIDDENDIR=/dev/.udevdb
ALLOWHIDDENDIR=/dev/.static
```

Solltest Du Dich bei der Gelegenheit in der Konfigurationsdatei etwas umsehen, sollte Dir die Option ins Auge stechen, im Fall des Falles eine Warn-E-Mail von *Rootkit Hunter* zugschickt zu bekommen. Sei darauf hingewiesen, dass die *Cron*-Jobs so intelligent geschrieben sind, dass sie dies bereits leisten. Die Aktivierung dieser Option bringt Dir also nichts Neues.

KAPITEL 17

Backups

Backups sind, wie ich bereits erwähnte, von enormer Bedeutung. Selbst wenn Du ein RAID-Array an Deinem Server hängen hast, brauchst Du Backups – ein RAID schützt Dich nur gegen Ausfälle, aber nicht gegen ein versehentliches

```
khazad-dum:~ # rm -rf /
```

Doch auch das tollste Backup nutzt nichts, wenn es sich im Fall des Falles nicht einlesen lässt (weil z. B. das verwendete Bandlaufwerk dejustiert ist) oder die Daten immer in einem inkonsistenten Zustand gesichert wurden. Prüfe Deine Backups also periodisch auf ihre Verwendbarkeit. Und behalte genügend alte Stände Deiner Backups zurück, denn wenn beispielsweise eine Anwendung eine Datei falsch oder unvollständig schreibt, weil sie während des Schreibvorgangs abstürzt, so hast Du eine wunderbare Sicherung einer Datei, mit der Du nichts mehr anfangen kannst. Du brauchst stattdessen den Stand *davor*.

Man kann noch viele Ratschläge zu diesem Thema geben. Glücklicherweise gibt es massenhaft Literatur, die sich mit dem Thema Datensicherung beschäftigt. Du solltest Dich ein wenig einlesen – und wenn es nur das kurzweilige *"Tao of Backup"*[*] ist.

Doch wie fertigt man Backups am besten an? Klar ist: Der Vorgang muss vollständig automatisierbar sein. Sobald Du etwas händisch anstoßen musst, vergisst Du es oder bist einfach zu faul dazu. Und dann ist hinterher das Geschrei groß. Daher werden wir für die Automatisierung den allseits beliebten *Cron* heranziehen.

Welche Daten Du sichern willst, musst Du selbst entscheiden. Beispiele dafür, was *ich* sichere, findest Du im Folgenden.

Aufpassen musst Du vor allem mit Datenbanken. Wenn Du eine Datenbank im laufenden Betrieb abziehst, ist es wahrscheinlich, dass das Backup der zugehörigen Daten Dir wenig bis gar nichts nutzt, da sie sich in einem inkonsistenten Zustand befinden dürften. Um das zu vermeiden, solltest Du die Datenbank vor der Anfertigung des Backups herunterfahren und danach wieder starten.[†] *Backup2l* (vgl. Abschnitt 17.2 auf Seite 135) bietet Dir hierfür Hooks an, in die Du Deine Skripte einhängen kannst.

---

[*] http://www.taobackup.com/

[†] Datenbanken erkennt man jedoch nicht immer als solche. Nur ein Beispiel dazu: Der in dieser Anleitung verwendete *Cyrus IMAPd* verwendet eine Datenbank, obwohl man das möglicherweise nicht erwartet. Du solltest Deinen *Cyrus IMAPd* also beenden, bevor Du sicherst. Und es ist auch unproblematisch,

Da ich meine privaten Projekte mittels *CVS* und *Subversion* verwalte und letzteres ebenfalls auf einer Datenbank aufsetzt, will ich Dir kurz mein Skript vorstellen, das mir vor dem Backup-Lauf einen Dump der Inhalte des Repositorys anfertigt.* Mit diesem Dump habe ich alles, was ich brauche, um das Repository vollständig wiederherzustellen. Wenn Du magst, darfst Du das Skript natürlich gerne noch verbessern; für mich tut es, was es soll.

/root/backupSVN.sh

```
1  #!/bin/bash
2
3  BACKUP_HOME=/var/backups/svn
4  ERROR=0
5
6  function usage() {
7    echo `basename $0` erstellt ein Dump-File der als Parameter
8    echo angegebenen SVN-Repositorys im Verzeichnis $BACKUP_HOME.
9  }
10
11 if test $# -lt 1 ; then
12   usage
13   exit 1
14 fi
15
16 SVNLOOK=`which svnlook 2> /dev/null`
17 if test ! "$SVNLOOK" -o ! -x "$SVNLOOK" ; then
18   echo svnlook nicht gefunden
19   exit 1
20 fi
21
22 SVNADMIN=`which svnadmin 2> /dev/null`
23 if test ! "$SVNADMIN" -o ! -x "$SVNADMIN" ; then
24   echo svnadmin nicht gefunden
25   exit 1
26 fi
27
28 while test $# -gt 0 ; do
29   REPOSITORY=$1
30   echo überprüfe $REPOSITORY...
31
32   REP_DIRNAME=`basename $REPOSITORY`
33   REP_PATH=`dirname $REPOSITORY`
34   if test "$REPOSITORY" != "$REP_PATH/$REP_DIRNAME" ; then
35     echo kann $REPOSITORY nicht korrekt zerlegen
```

wenn währenddessen Mails eingehen – der *Exim4* behält diese solange in seiner Queue und versucht periodisch, sie erneut zuzustellen. Also keine Sorge, Du verlierst dadurch keine einzige E-Mail.

* Ein Shutdown entfällt, da kein Daemon beteiligt ist und die Dump-Operation atomar abläuft.

```
36      ERROR=1
37      shift
38      continue
39    fi
40
41    YOUNGEST=‘$SVNLOOK youngest $REPOSITORY 2> /dev/null‘
42    if test $? -ne 0 ; then
43      echo $REPOSITORY ist kein SVN-Repository
44      ERROR=1
45      shift
46      continue
47    fi
48    echo jüngste Version: $YOUNGEST
49
50    BACKUP_NAME=$BACKUP_HOME/BackUp_${REP_DIRNAME}_Revision_0_${YOUNGEST}
51    if test -f $BACKUP_NAME ; then
52      echo Version $YOUNGEST von $REPOSITORY ist bereits gesichert
53      shift
54      continue
55    fi
56
57    echo sichere $REPOSITORY nach $BACKUP_NAME
58    $SVNADMIN dump $REPOSITORY > $BACKUP_NAME
59    if test $? -ne 0 ; then
60      echo Sicherung von $REPOSITORY fehlgeschlagen
61      ERROR=1
62      shift
63      continue
64    fi
65
66    echo Sicherung von $REPOSITORY in Version $YOUNGEST erfolgreich.
67    shift
68  done
69
70  if test $ERROR -eq 0 ; then
71    exit 0
72  fi
73
74  echo Während der Ausführung traten Fehler auf.
75  exit 1
```

Für die eigentliche Datensicherung gibt es – wie fast immer – mehrere Wege. Da ich kein Bandlaufwerk besitze, ziehe ich es vor, die Daten im Rechner zu belassen, allerdings auf einer zusätzlichen Festplatte. Außerdem erstelle ich gelegentlich Kopien der Backups auf einem weiteren Rechner.

Klar sein muss dabei: Wenn der Keller, in dem der Rechner steht, überschwemmt wird oder es dort brennt, ist alles fort. Ich bin mir dessen bewusst und denke, dass ich nach einem Feuer andere Sorgen habe als meine Mails.

Aber Du hast mich erwischt: Meine *CVS*- und *Subversion*-Repositorys sichere ich zugegebenermaßen zusätzlich periodisch auf optische Datenträger. Ich unterscheide in meiner Backup-Strategie also zwischen für mich wichtigen und weniger wichtigen Daten.

17.1 Datensicherung selbstgebacken

Der naive Ansatz für ein Backup ist ein selbstgebackenes kleines Skript, das Du – je nach Wichtigkeit der Daten – zweimal die Woche (z. B. Sonntag und Mittwoch jeweils um 23 Uhr) ausführen lässt. Es packt all Deine wichtigen Dateien zusammen und sichert sie in ein Verzeichnis, das *nicht* auf der Festplatte liegt, deren Daten Du sicherst, sondern auf einer *zusätzlichen* Platte. Ab und an solltest Du so ein Backup auch auf einen weiteren Rechner legen oder am besten auf ein optisches Medium brennen.

Mein Skript sieht folgendermaßen aus:

/root/backup.sh

```
1 #!/bin/sh
2 myDate=‘date +%Y_%m_%d_-_%H_%M‘
3 tar cvjf /export2/Backup/backup_khazad-dum_${myDate}.tar.bz2 \
4          /root/.ssh /root/backup /root/bin /usr/lib/AntiVir/hbedv.key \
5          /etc /var/spool/uucp /var/backups /var/cache/debconf \
6          /var/spool/sieve /var/spool/cyrus /home /var/lib/dpkg
7 # vi:tw=0
```

Solltest Du weitere wichtige Dateien oder Verzeichnisse haben, die Du sichern willst, musst Du das Skript entsprechend ergänzen. Was auch immer Du tust: Vergiss nicht, es mittels

```
khazad-dum:~ # chmod u+x /root/backup.sh
```

ausführbar zu machen und *cron* zur periodischen Abarbeitung anzuvertrauen. Zu diesem Zweck könnte die `crontab` Deines Benutzers `root` die folgende Zeile enthalten:

```
0 23 * * Sun,Wed /root/backup.sh
```

Großer Nachteil dieses Skripts ist es, dass auch Dateien, die sich so gut wie nie ändern, jedes Mal im Backup enthalten sind, obwohl eine einmalige Sicherung irgendwann am Anfang völlig gereicht hätte.

17.2 Datensicherung mit *Backup2l*

Eine raffiniertere Datensicherungsmethode bietet das Paket *backup2l*. Es handelt sich dabei um ein Skript, das Backups in Verzeichnissen erstellt und die gezielte Wiederherstellung einzelner Dateien oder kompletter Sicherungen ermöglicht. Es leistet die Erstellung von Vollbackups sowie darauf aufbauender inkrementeller* Sicherungen. Es ist also kein Problem, die übliche Backup-Strategie von Monats-, Wochen- und Tagesbändern nachzuahmen.

Meine Backup-Strategie sieht so aus: Zuerst wird eine Vollsicherung (Level 0) angefertigt. Ab dann wird, darauf aufbauend, inkrementell gesichert. Jede inkrementelle Sicherung enthält die Änderungen gegenüber der letzten Sicherung auf demselben oder dem nächstniedrigeren Level.

Meine Konfigurationsdatei sieht folgendermaßen aus:

/etc/backup2l.conf

```
FOR_VERSION=1.3
VOLNAME="all"
# zu sichernde Verzeichnishierarchien:
SRCLIST=(/root \
         /usr/lib/AntiVir \
         /etc \
         /var/spool \
         /var/backups \
         /var/cache/debconf \
         /home \
         /var/lib)
# dabei auslassen:
SKIPCOND=(-path "*.nobackup*" \
          -o -name "*.o" \
          -o -path "/var/spool/cups*" \
          -o -path "/var/spool/exim4*" \
          -o -path "/var/spool/squid*" \
          -o -path "/var/spool/uucp/chico*" \
          -o -path "/var/lib/apt/lists*")
# wohin sichern
BACKUP_DIR="/export2/Backup/khazad-dum"
MAX_LEVEL=3
MAX_PER_LEVEL=8
MAX_FULL=8
GENERATIONS=5
CREATE_CHECK_FILE=1
AUTORUN=0
```

* Die Man Page spricht von *differential backups*. Das kann ich aber nicht so recht glauben, denn dann müssten aufeinanderfolgende Backups derselben Ebene stetig größer werden. Das ist aber nicht der Fall. Insofern dürften sie stattdessen *inkrementell* sein, was auch die Inhalte der Archive nahe legen.

```
28  SIZE_UNITS=""
29  # extrem schnelle Extraktion einzelner Dateien möglich
30  CREATE_DRIVER="DRIVER_AFIOZ"
31
32  PRE_BACKUP ()
33  {
34      echo "  pre-backup running"
35      echo "  writing dpkg selections to /root/dpkg-selections.log..."
36      dpkg --get-selections | diff - /root/dpkg-selections.log \
37         > /dev/null || dpkg --get-selections > /root/dpkg-selections.log
38      echo "  backing up SVN repositories..."
39      /root/bin/backupSVN.sh /foo/bar/SVN/Repository1 \
40                             /foo/bar/SVN/Repository2
41      echo "  stopping Cyrus IMAPd..."
42      /etc/init.d/cyrus21 stop
43  }
44
45  POST_BACKUP ()
46  {
47      echo "  post-backup running"
48      echo "  starting Cyrus IMAPd..."
49      /etc/init.d/cyrus21 start
50  }
```

Was bedeutet das konkret? Ich habe `MAX_LEVEL=3` und `MAX_PER_LEVEL=8` gesetzt. Das bedeutet, dass nach acht Level-3-Backups ein Level-2-Backup angefertigt wird. Nach acht Level-2-Backups (und somit $8 \cdot 8$ Level-3-Backups) wird ein Level-1-Backup erzeugt. Und somit wird nach acht Level-1-Backups, also $8 \cdot 8$ Level-2-Backups oder $8 \cdot 8 \cdot 8$ Level-3-Backups, ein weiteres Level-0-Backup (eine Vollsicherung also) geschrieben.

Da sich – von meinen Mails abgesehen – die Datenbestände auf meinem Server nicht so furchtbar ändern, ist das eine gute Strategie. Denn ich erzeuge nun nur alle $8^3 + 8^2 + 8 = 584$ Tage eine Vollsicherung. Um meinen Datenbestand vollständig wiederherzustellen, brauche ich nur maximal $8 \cdot 3 + 1 = 25$ Archive heranzuziehen. Tipp: Wenn Du als „Treiber“ *AFIO* einsetzt und demzufolge das Paket *afio* installierst, dauert die Wiederherstellung einzelner Dateien nur wenige Sekunden.

Ich habe mich dazu entschieden, maximal acht Level-0-Backups (`MAX_FULL=8`) auf meinem System liegen zu lassen. Bei der derzeitigen Datenmenge sind das gerade 24 GB zzgl. der inkrementellen Backups, die sich aber meist im Bereich weniger zehn MB bewegen. Und den Platz habe ich locker. Sollte ich merken, das das knapp wird, kann ich die Variablen jederzeit abändern – das Skript ist robust genug, das korrekt zu handhaben (behauptet zumindest dessen Autor – ich habe es nicht nachgeprüft).

Interessant sind schließlich die beiden Funktionen `PRE_BACKUP` und `POST_BACKUP`, die es Dir ermöglichen, vor Beginn und nach Ende des Backups Aktionen auszuführen, beispielsweise die Beendigung bzw. den Start von Daemon-Prozessen.

KAPITEL 18

Bauen eines eigenen Kernels

Einen eigenen LINUX-Kernel kann man unter Debian GNU/Linux mit der üblichen Prozedur

```
khazad-dum:~ # cd /usr/src/linux
khazad-dum:/usr/src/linux # make menuconfig
...
khazad-dum:/usr/src/linux # make bzImage modules
...
```

bauen. Das geht jedoch völlig am Paketmanagement vorbei. Ich persönlich ziehe es daher vor, mir mein eigenes Kernel-Paket zu schnüren – vor allem deshalb, weil es so herrlich unkompliziert zu machen ist.

Gründe dafür, einen eigenen Kernel zu bauen, gibt es viele. Man möchte vielleicht die ganzen Module, die die Standardkernel mitbringen, nicht alle haben, möchte weitere Optimierungen einbringen, Patches nutzen, immer die aktuellste Kernel-Version laufen haben oder – wie in meinem Fall – prinzipiellen Problemen aus dem Weg gehen.

Mein Server lief problemlos mit den Standard-Kernels ab 2.6.8.1. Mein Desktop-System (`imladris`, mit Asus P4PE-Mainboard) dagegen hatte mit sowohl *kernel-image-2.6.8-1-686* als auch *kernel-image-2.6.8-2-686* Schwierigkeiten mit dem Auslesen von CDs mittels *cdparanoia*. Erst *kernel-image-2.6.9-1-686* brachte hier Besserung. Aber ein anderes Problem bestand bei allen drei Kernel-Versionen: Ich konnte DMA für mein ATAPI-DVD-ROM-Laufwerk nicht aktivieren. Der Versuch sah so aus:

```
khazad-dum:~ # hdparm -d 1 /dev/hdc
/dev/hdc:
     setting using_dma to 1 (on)
     HDIO_SET_DMA failed: Operation not permitted
     using_dma    =  0 (off)
khazad-dum:~ #
```

Nach einigem Herumprobieren kam ich darauf, dass der standardmäßig als Modul vorliegende IDE-Treiber schuld daran war. Mit meinem eigenen Kernel mit fest eincompiliertem `piix`-Treiber hatte ich keine derartigen Probleme mehr damit.

Voraussetzung dafür, selbst einen Kernel übersetzen zu können, ist erst einmal eine Infrastruktur aus Entwicklungspaketen. Du brauchst grundsätzlich die Pakete *gcc*, *libc6-dev* und auf x86-Plattformen *bin86*. Für ein `make menuconfig`, das die Kernel-Konfiguration über eine semigraphische Oberfläche ermöglicht, ist zusätzlich *ncursesX.X-dev* erforderlich. Ziehst Du ein klickbares Interface vor, kannst Du die Targets `xconfig` oder `gconfig` nutzen. Für ersteres benötigst Du die Pakete *g++* und *libqt3-mt-dev*, für letzteres *libglade2-dev*.

Als nächstes lädst Du Dir entweder einen Vanilla-Kernel* herunter oder aber ein Debian-Kernel-Source-Paket. Ich ziehe ersteres vor. Egal, welchen Weg Du gehen willst – das erhaltene Ding packst Du unter `/usr/src` aus:

```
khazad-dum:~ # cd /usr/src
khazad-dum:/usr/src # tar xvjf /tmp/linux-2.6.11.tar.bz2
linux-2.6.11/
linux-2.6.11/arch/
linux-2.6.11/arch/i386/
linux-2.6.11/arch/i386/kernel/
linux-2.6.11/arch/i386/kernel/process.c
...
khazad-dum:/usr/src #
```

Dann konfigurierst Du den Kernel. Als Basis kannst Du die Kernel-Konfiguration eines derzeit auf der eigenen Maschine laufenden älteren Kernels derselben Serie (z. B. Kernel 2.6.9) verwenden (typischerweise unter `/boot/config-*` abgelegt). Du kopierst die Konfigurationsdatei unter dem Namen `.config` ins Kernel-Source-Verzeichnis und rufst `make oldconfig` auf. Anschließend kannst Du mittels `make menuconfig` (oder, falls Du das *X Window System* laufen hast, `make xconfig` bzw. `make gconfig`) Deinen eigenen Kernel zurechtzimmern.

```
khazad-dum:/usr/src # cd linux-2.6.11
khazad-dum:/usr/src/linux-2.6.11 # cp /boot/config-2.6.9-1-686 \
                                     .config
khazad-dum:/usr/src/linux-2.6.11 # make oldconfig
...
khazad-dum:/usr/src/linux-2.6.11 # make menuconfig
...
khazad-dum:/usr/src/linux-2.6.11 #
```

Als nächstes erstellst Du Dir die passenden Debian-Pakete (Plural deshalb, weil zumindest ich auf meinem Desktop-System nicht nur den Kernel, sondern auch das dazu passende NVIDIA-Kernelmodul baue). Dazu bearbeitest Du zunächst `/etc/kernel-pkg.conf` mit Deinem Lieblingseditor und trägst Dich dort als Maintainer ein. Dann führst Du folgende Kommandos aus (beachte, dass Du die `--added-modules=nvidia-kernel`-Option auf Deinem Server wahrscheinlich nicht brauchst):

```
khazad-dum:/usr/src/linux-2.6.11 # make-kpkg clean
```

* zu finden unter `ftp://ftp.de.kernel.org/pub/linux/kernel/v2.6/`

```
/usr/bin/make -f /usr/share/kernel-package/rules real_stamp_clean
make[1]: Entering directory '/usr/src/linux-2.6.11'
...
khazad-dum:/usr/src/linux-2.6.11 # make-kpkg --revision=custom.1.0 \
                                     --append-to-version=-bla \
                                     --initrd \
                                     --added-modules=nvidia-kernel \
                                     kernel_image modules_image
khazad-dum:/usr/src/linux-2.6.11 #
```

Die `--revision` wird lediglich an das Paket angehängt; es findet keine Entsprechung im Kernel- bzw. Modulnamen. Allerdings verwendet *dpkg* die Nummer zur Sortierung, so dass es weiß, ob ein Up- oder Downgrade stattfindet. Pass also ein bisschen auf damit. Für den Zweck, verschiedene Flavors* zu kennzeichnen, ist `--append-to-version=-bla` gedacht. Es setzt die Variable `EXTRAVERSION` im Kernel-Makefile.

Hast Du externe Module, die Du immer automatisch integriert wissen möchtest, so kannst Du die Module unterhalb von `/usr/src/modules` ablegen. Dann reicht es aus, den Modulnamen mit der Option `--added-modules=...` auf der Kommandozeile zu spezifizieren. Willst Du mehrere Module gebaut wissen, so gib deren Namen durch Kommata getrennt an. Vergiss aber nicht das zusätzliche Target `modules_image`, denn sonst werden die externen Module nicht erstellt.

Ein Beispiel: Ich für meinen Teil benötige auf meinem Desktop-System das NVIDIA-Kernelmodul. Für nVidia-Grafikkarten gibt es praktischerweise ein Debian-Paket, das die nötigen Sourcen enthält. Wenn Du das Paket *nvidia-kernel-source* einspielst, installiert es Dir die Datei `/usr/src/nvidia-kernel-source.tar.gz`. Diese packst Du im Verzeichnis `/usr/src` aus, was Dir das Verzeichnis `modules/nvidia-kernel` beschert. Das bedeutet für Dich, dass Du als Option `--added-modules=nvidia-kernel` verwenden musst.

Willst Du einen neuen Kernel aus denselben Sourcen bauen, so vergiss nicht, zuerst ein

```
khazad-dum:/usr/src/linux-2.6.11 # make-kpkg clean
```

durchzuführen. Und als `--revision` nimmst Du nun natürlich `custom.2.0`.

Die Pakete werden im Verzeichnis `/usr/src` abgelegt, von wo aus Du sie nun mittels

```
khazad-dum:/usr/src # dpkg -i /usr/src/kernel-image-2.6.11-...deb \
                        /usr/src/nvidia-kernel-2.6.11-...deb
```

installieren kannst. Anschließend musst Du einen Reboot durchführen und den neuen Kernel testen.

Ab dieser Stelle lasse ich Dich mit Deinen Problemen alleine. Aber da Du immer wieder zu den bereits installierten *funktionierenden* Kernels zurück kannst, habe ich da auch keinerlei Skrupel oder Hemmungen ...

* Du willst also verschiedene 2.6.11-Kernel bauen, die sich z. B. in ihren Modulen unterscheiden.

KAPITEL 19

Zertifikatsverwaltung mit OpenSSL

Zertifikate sind ein wunderbarer Mechanismus, wenn es darum geht, die Authentizität einer Person oder eines Dienstes sicher zu stellen. Der Zertifikatsmechanismus basiert auf asymmetrischer Verschlüsselung, arbeitet also mit öffentlichen und privaten Schlüsseln. Die öffentlichen Schlüssel kannst Du ruhig weltweit verteilen. Auf die privaten dagegen musst Du aufpassen.

Die Zertifikate werden von einer *Certification Authority* (CA) vergeben. Genauer: Zertifikatsanfragen (*Certification Requests*) werden von ihr signiert. Das kostet meistens Geld. Da Du kein offizielles Zertifikat brauchst (oder doch?), zeige ich Dir, wie Du einfach Deine eigene CA aufsetzen kannst, um Dienste mit Serverschlüsseln zu versehen. Den öffentlichen Schlüssel Deiner CA kannst Du dann als „vertrauenswürdig“ in Deinen E-Mail-Client, Browser oder sonst was integrieren. Serverschlüssel gelten damit auch als vertrauenswürdig, so sie von Deiner Certification Authority signiert wurden. Achte aber darauf, dass die Gültigkeitsdauer Deiner Zertifikate vollständig innerhalb der Deiner signierenden CA liegt.

Im Folgenden wirst Du Deine eigene CA aufsetzen und Deine Schlüssel am Beispiel des *Cyrus IMAPd* generieren sowie in den Daemon einbinden. Als ich mir angesehen habe, wie das geht, waren mir vor allem [2] sowie [3] sehr hilfreich.

Voraussetzung für das Weitere ist, dass Du *OpenSSL* installiert hast.

Wenn Du gar nicht so genau wissen willst, wie diese ganze Generiererei überhaupt abläuft, sondern nur schnell ein paar Zertifikate zurechtklicken willst, solltest Du Dir mal *TinyCA* ansehen. Das ist ein graphisches Front-End zu *OpenSSL*. Ich mache so was allerdings lieber von Hand. Bei Dir mag das anders aussehen; dann installierst Du eben *tinyca* und tobst Dich damit aus.

Deine Architektur wird wie in Abb. 19.1 auf der nächsten Seite dargestellt aussehen: Du hast eine Root-CA, der eine Server- und eine User-CA untergeordnet sind. Diese wiederum verteilen Server- bzw. User-Zertifikate.

In dieser Anleitung gehen wir die Hierarchie auch komplett durch: Du erstellst zunächst die Root-CA, dann die Server-CA und abschließend ein Zertifikat für den *Cyrus IMAPd*. Die Erstellung der User-CA bleibt Dir dann als (sehr einfache, weil analoger Verlauf) Übung überlassen.

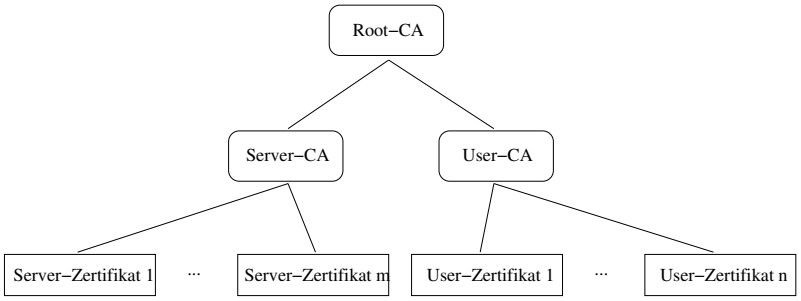

Abbildung 19.1: *Zertifikats-Hierarchie*

1. **Konfigurationsdatei erstellen.**

   Du benötigst eine Konfigurationsdatei, um die Zertifikate zu erstellen. Der Default in `/etc/ssl/openssl.cnf` ist nicht ausreichend; Du solltest ihn daher durch Deine eigene Version ersetzen. Meine gebe ich Dir hier als Beispiel an (am einfachsten ist es sicherlich, wenn Du die aus dem Tar-Ball auf meiner Web-Seite als Basis heranziehst):

/etc/ssl/openssl.cnf

```
# Obi-Wans OpenSSL-Konfigurationsdatei
#

HOME        = .
RANDFILE    = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file    = $ENV::HOME/.oid
oid_section = new_oids

path = /etc/ssl

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1 = 1.2.3.4
# Or use config file substitution like this:
# testoid2 = ${testoid1}.5.6

####################################################################

[ ca ]

default_ca    = Server_CA        # The default ca section

####################################################################

[ Root_CA ]

dir             = $path/RootCA             # Where everything is kept
certs           = $dir/certs               # Where the issued certs are kept
crl_dir         = $dir/crl                 # Where the issued crl are kept
database        = $dir/index.txt           # database index file.
new_certs_dir   = $dir/newcerts            # default place for new certs.

certificate     = $dir/private/RCAcert.pem # The CA certificate
serial          = $dir/serial              # The current serial number
```

```
39  crl             = $dir/crl.pem             # The current CRL
40  private_key     = $dir/private/RCAkey.pem  # The private key
41  RANDFILE        = $dir/private/.rand       # private random number file
42
43  x509_extensions  = RCA_ext                 # The extentions to add to the cert
44
45  default_days     = 1825                    # fuenf Jahre
46  default_crl_days = 365                     # ein Jahr
47  default_md       = md5                     # which md to use.
48  preserve         = no                      # keep passed DN ordering
49
50  # A few difference way of specifying how similar the request should look
51  # For type CA, the listed attributes must be the same, and the optional
52  # and supplied fields are just that :-)
53  policy            = policy_match
54
55  ####################################################################
56
57  [ Server_CA ]
58
59  dir             = $path/ServerCA           # Where everything is kept
60  certs           = $dir/certs               # Where the issued certs are kept
61  crl_dir         = $dir/crl                 # Where the issued crl are kept
62  database        = $dir/index.txt           # database index file.
63  new_certs_dir   = $dir/newcerts            # default place for new certs.
64
65  certificate     = $dir/private/SCAcert.pem  # The CA certificate
66  serial          = $dir/serial              # The current serial number
67  crl             = $dir/crl.pem             # The current CRL
68  private_key     = $dir/private/SCAkey.pem   # The private key
69  RANDFILE        = $dir/private/.rand       # private random number file
70
71  x509_extensions  = SCA_ext                 # The extentions to add to the cert
72
73  default_days     = 1825                    # fuenf Jahre
74  default_crl_days = 365                     # ein Jahr
75  default_md       = md5                     # which md to use.
76  preserve         = no                      # keep passed DN ordering
77
78  policy          = policy_anything
79
80  ####################################################################
81
82  [ User_CA ]
83
84  dir             = $path/UserCA             # Where everything is kept
85  certs           = $dir/certs               # Where the issued certs are kept
86  crl_dir         = $dir/crl                 # Where the issued crl are kept
87  database        = $dir/index.txt           # database index file.
88  new_certs_dir   = $dir/newcerts            # default place for new certs.
89
90  certificate     = $dir/private/UCAcert.pem  # The CA certificate
91  serial          = $dir/serial              # The current serial number
92  crl             = $dir/crl.pem             # The current CRL
93  private_key     = $dir/private/UCAkey.pem   # The private key
94  RANDFILE        = $dir/private/.rand       # private random number file
95
96  x509_extensions  = UCA_ext                 # The extentions to add to the cert
97
98  default_days     = 1825                    # fuenf Jahre
99  default_crl_days = 365                     # ein Jahr
100 default_md       = md5                     # which md to use.
101 preserve         = no                      # keep passed DN ordering
102
103 policy          = policy_anything
104
```

```
105 ##################################################################
106
107 # For the CA policy
108 [ policy_match ]
109
110 countryName            = match
111 stateOrProvinceName    = supplied
112 localityName           = optional
113 organizationName       = supplied
114 organizationalUnitName = optional
115 commonName             = supplied
116 emailAddress           = optional
117
118 # For the 'anything' policy
119 # At this point in time, you must list all acceptable 'object'
120 # types.
121
122 [ policy_anything ]
123 countryName            = match
124 stateOrProvinceName    = optional
125 localityName           = optional
126 organizationName       = optional
127 organizationalUnitName = optional
128 commonName             = supplied
129 emailAddress           = optional
130
131 ##################################################################
132
133 [ req ]
134
135 default_bits       = 2048
136 default_keyfile    = privkey.pem
137 distinguished_name = req_distinguished_name
138 attributes         = req_attributes
139 x509_extensions    = v3_ca    # The extentions to add to the self signed cert
140
141 string_mask        = nombstr
142
143 ##################################################################
144
145 [ req_distinguished_name ]
146
147 countryName                    = Country Name (2 letter code)
148 countryName_default            = DE
149 countryName_min                = 2
150 countryName_max                = 2
151
152 stateOrProvinceName            = State or Province Name (full name)
153 stateOrProvinceName_default    = Bavaria
154
155 localityName                   = Locality Name (eg, city)
156 localityName_default           = Nuremberg
157
158 organizationName               = Organization Name (eg, company)
159 organizationName_default       = Kenobi Laboratories
160
161 # we can do this but it is not needed normally :-)
162 #1.organizationName            = Second Organization Name (eg, company)
163 #1.organizationName_default    = World Wide Web Pty Ltd
164
165 organizationalUnitName         = Organizational Unit Name (eg, section)
166 organizationalUnitName_default = KenobiLab CA
167
168 commonName                     = Common Name (eg, YOUR name)
169 commonName_default             = Obi-Wan Kenobi
170 commonName_max                 = 64
```

```
171
172     emailAddress                    = Email Address
173     emailAddress_default            = obiwan@meins.de
174     emailAddress_max                = 64
175
176     # SET-ex3                       = SET extension number 3
177
178     ####################################################################
179
180     [ req_attributes ]
181
182     # Das Challenge Password dient dazu, sich bei Verlust des geheimen
183     # Schluessels gegenueber der Herausgeber-CA fuer einen
184     # Zertifikatswiderruf auszuweisen. Wird bei der Erstellung der
185     # Zeritifikatsanforderung erfragt.
186
187     challengePassword     = A challenge password
188     challengePassword_min = 4
189     challengePassword_max = 20
190
191     unstructuredName      = An optional company name
192
193     ####################################################################
194
195     [ RCA_ext ]
196
197     basicConstraints        = critical, CA:TRUE
198     keyUsage                = cRLSign, keyCertSign
199     subjectKeyIdentifier    = hash
200     authorityKeyIdentifier  = keyid,issuer:always
201     subjectAltName          = email:copy
202     issuerAltName           = issuer:copy
203     # crlDistributionPoints = URI:http://khazad-dum.arda.lan/RCA.crl
204     nsCertType              = sslCA, emailCA, objCA
205     # nsBaseUrl             = https://khazad-dum.arda.lan/
206     # This will be displayed in Netscape's comment listbox.
207     nsComment               = "ausgegeben von KenobiCA"
208
209     ####################################################################
210
211     [ SCA_ext ]
212
213     # basicConstraints      = critical, CA:FALSE
214     keyUsage                = digitalSignature, keyEncipherment
215     subjectKeyIdentifier    = hash
216     authorityKeyIdentifier  = keyid,issuer:always
217     subjectAltName          = email:copy
218     issuerAltName           = issuer:copy
219     # crlDistributionPoints = URI:http://khazad-dum.arda.lan/SCA.crl
220     nsCertType              = server
221     # nsBaseUrl             = https://khazad-dum.arda.lan/
222     nsComment               = "ausgegeben von einer Kenobi-Server-CA"
223
224     ####################################################################
225
226     [ UCA_ext ]
227
228     # basicConstraints      = critical, CA:FALSE
229     keyUsage                = digitalSignature, keyEncipherment, keyAgreement
230     subjectKeyIdentifier    = hash
231     authorityKeyIdentifier  = keyid,issuer:always
232     subjectAltName          = email:copy
233     issuerAltName           = issuer:copy
234     # crlDistributionPoints = URI:http://khazad-dum.arda.lan/SCA.crl
235     nsCertType              = client, email
236     # nsBaseUrl             = https://khazad-dum.arda.lan/
```

```
237 nsComment               = "ausgegeben von einer Kenobi-User-CA"
238
239 ####################################################################
240
241 [ v3_ca ]
242
243 basicConstraints        = critical, CA:TRUE
244 keyUsage                = cRLSign, keyCertSign
245 subjectKeyIdentifier    = hash
246 authorityKeyIdentifier  = keyid,issuer:always
247 subjectAltName          = email:copy
248 issuerAltName           = issuer:copy
249 # crlDistributionPoints = URI:http://khazad-dum.arda.lan/RCA.crl
250 nsCertType              = sslCA, emailCA, objCA
251 # nsBaseUrl             = https://khazad-dum.arda.lan/
252 # This will be displayed in Netscape's comment listbox.
253 nsComment               = "ausgegeben von einer RootCA"
254
255 ####################################################################
256
257 [ crl_ext ]
258
259 # CRL extensions.
260 # Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.
261
262 issuerAltName          = issuer:copy
263 authorityKeyIdentifier = keyid:always,issuer:always
```

2. **Eigene Root-CA aufsetzen.**

   Hier erzeugst Du Dein *Root-Zertifikat*. Dazu legst Du zunächst die grundlegende Verzeichnisstruktur samt einiger Dateien unter `/etc/ssl` an. Dazu wiederholst Du den im Folgenden abgedruckten Vorgang für die Verzeichnisse `RootCA`, `ServerCA` und `UserCA`. Nicht wundern: Ich habe hier nur das Beispiel `RootCA` abgedruckt.

```
khazad-dum:~ # cd /etc/ssl
khazad-dum:/etc/ssl # mkdir RootCA
khazad-dum:/etc/ssl # cd RootCA
khazad-dum:/etc/ssl/RootCA # mkdir certs newcerts private
khazad-dum:/etc/ssl/RootCA # chmod go-rwx private
khazad-dum:/etc/ssl/RootCA # echo "01" > serial
khazad-dum:/etc/ssl/RootCA # touch index.txt
khazad-dum:/etc/ssl/RootCA # ls -l
drwxr-xr-x  2 root root 6 Dec 14 20:50 certs
-rw-r--r--  1 root root 0 Dec 14 20:50 index.txt
drwxr-xr-x  2 root root 6 Dec 14 20:50 newcerts
drwx------  2 root root 6 Dec 14 20:50 private
-rw-r--r--  1 root root 3 Dec 14 20:50 serial
khazad-dum:/etc/ssl/RootCA # cd ..
khazad-dum:/etc/ssl #
```

   Jetzt generierst Du ein selbstsigniertes Zertifikat für Deine CA. Du wirst feststellen, dass aufgrund unserer Vorarbeit aus Schritt 1 bereits einige Eingabefelder korrekt

vorbesetzt sind, so dass Du jeweils nur die Return-Taste zu drücken brauchst. Das Zertifikat ist 1825 Tage gültig, also fünf Jahre. Die Passphrase, nach der Du gefragt wirst, solltest Du Dir gut merken. Du brauchst sie immer dann, wenn Du ein neues Zertifikat signieren möchtest.

Zertifikat für die CA erzeugen

```
khazad-dum:/etc/ssl# openssl req -newkey rsa:2048 -x509 -days 1825 \
                           -out RootCA/private/RCAcert.pem -outform PEM \
                           -keyout RootCA/private/RCAkey.pem
Generating a 2048 bit RSA private key
.....................+++
.............................................+++
writing new private key to 'RootCA/private/RCAkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [Bavaria]:
Locality Name (eg, city) [Nuremberg]:
Organization Name (eg, company) [Kenobi Laboratories]:
Organizational Unit Name (eg, section) [KenobiLab CA]:KenobiLab Root-CA
Common Name (eg, YOUR name) [Obi-Wan Kenobi]:
Email Address [obiwan@meins.de]:

khazad-dum:/etc/ssl# find RootCA -type f
RootCA/private/RCAkey.pem
RootCA/private/RCAcert.pem
RootCA/serial
RootCA/index.txt
khazad-dum:/etc/ssl#
```

Ergebnis dieses Vorgangs ist ein selbstsigniertes Zertifikat (dafür ist `-x509` verantwortlich) im PEM-Format mit RSA-Schlüsselpaar. Die Schlüssellänge beträgt 2 048 Bit (wenn Du mehr möchtest, musst Du das entsprechend ändern). Das Zertifikat ist `/etc/ssl/RootCA/private/RCAcert.pem`, der private Schlüssel `/etc/ssl/RootCA/private/RCAkey.pem`.

Du musst das Zertifikat jetzt noch nach `/etc/ssl/certs` kopieren und den Namen dabei in `00.pem` abändern (das entspricht der Seriennummer des Zertifikats). Anschließend verlinken wir das Zertifikat über seinen Hash-Wert. Also:

```
khazad-dum:/etc/ssl # cp RootCA/private/RCAcert.pem certs/00.pem
khazad-dum:/etc/ssl # cd certs
khazad-dum:/etc/ssl/certs # c_rehash .
Doing .
00.pem => 7d7f13c8.0
khazad-dum:/etc/ssl/certs # cd ..
khazad-dum:/etc/ssl #
```

Es ist wichtig, alle Zertifikate über ihren Hash-Wert zu verlinken, da die Suche nach Zertifikaten *immer* über diesen erfolgt.

3. **Server-CA erstellen**

   Deine Root-CA steht damit. Jetzt brauchst Du die Server-CA, damit Du Server-Zertifikate erstellen kannst. Dazu erstellst Du wieder eine Zertifikatsanfrage und signierst diese mit Deinem Root-Zertifikat. Die verwendete Schlüssellänge beträgt wieder 2 048 Bit.

Zertifikatsanfrage für die Server-CA erzeugen

```
khazad-dum:/etc/ssl# openssl req -newkey rsa:2048 -days 1825 \
                            -out ServerCA/private/SCAreq.pem -outform PEM \
                            -keyout ServerCA/private/SCAkey.pem
Generating a 2048 bit RSA private key
.....................................................................+++
.....................................................+++
writing new private key to 'ServerCA/private/SCAkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [Bavaria]:
Locality Name (eg, city) [Nuremberg]:
Organization Name (eg, company) [Kenobi Laboratories]:
Organizational Unit Name (eg, section) [KenobiLab CA]:KenobiLab Server-CA
Common Name (eg, YOUR name) [Obi-Wan Kenobi]:
Email Address [obiwan@meins.de]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:EinChallengePasswort
An optional company name []:

khazad-dum:/etc/ssl# find ServerCA -type f
ServerCA/private/SCAkey.pem
ServerCA/private/SCAreq.pem
ServerCA/serial
ServerCA/index.txt
khazad-dum:/etc/ssl#
```

Zertifikatsanfrage für die Server-CA signieren

```
khazad-dum:/etc/ssl# openssl ca -name Root_CA -in ServerCA/private/SCAreq.pem \
                                -out ServerCA/private/SCAcert.pem
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for /etc/ssl/RootCA/private/RCAkey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
```

```
localityName          :PRINTABLE:'Nuremberg'
organizationName      :PRINTABLE:'Kenobi Laboratories'
organizationalUnitName:PRINTABLE:'KenobiLab Server-CA'
commonName            :PRINTABLE:'Obi-Wan Kenobi'
emailAddress          :IA5STRING:'obiwan@meins.de'
Certificate is to be certified until Dec 16 09:05:59 2009 GMT (1825 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

khazad-dum:/etc/ssl# find ServerCA -type f
ServerCA/private/SCAkey.pem
ServerCA/private/SCAreq.pem
ServerCA/private/SCAcert.pem
ServerCA/serial
ServerCA/index.txt

khazad-dum:/etc/ssl#
```

Auch dieses Zertifikat behandelst Du wieder wie vorhin das Root-Zertifikat. Tipp: Die Seriennummer bekommst Du heraus, indem Du das `index.txt` der signierenden CA anschaust.

4. **IMAP-Server-Zertifikatsanfrage und -Schlüssel erstellen.**

   Jetzt erstellst Du eine Zertifikatsanfrage für Deinen IMAP-Server, die durch die Signatur der Server-CA zum Zertifikat wird. Beachte bitte, als Common Name *unbedingt* den Namen Deines Servers einzutragen, und zwar so, wie Du (und Deine Anwender) ihn zukünftig zu verwenden gedenkst. Da ich meinen IMAP-Server unter dem Namen `imap.arda.lan` anspreche, nicht jedoch unter dem „richtigen“ Namen des Rechners `khazad-dum.arda.lan`, verwende ich auch `imap.arda.lan` für das Zertifikat.

Zertifikatsanfrage für den IMAP-Server erzeugen

```
khazad-dum:/etc/ssl# openssl req -newkey rsa:1024 -keyout CyrusKey.pem -keyform PEM \
                                 -out CyrusReq.pem -outform PEM
Generating a 1024 bit RSA private key
..............++++++
....++++++
writing new private key to 'CyrusKey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [Bavaria]:
Locality Name (eg, city) [Nuremberg]:
Organization Name (eg, company) [Kenobi Laboratories]:
Organizational Unit Name (eg, section) [KenobiLab CA]:KenobiLab R&D
```

```
Common Name (eg, YOUR name) [Obi-Wan Kenobi]:imap.arda.lan
Email Address [obiwan@meins.de]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:EinChallengePasswort
An optional company name []:

khazad-dum:/etc/ssl# ls -l Cyrus*
-rw-r--r--  1 root root 963 2004-12-17 10:23 CyrusKey.pem
-rw-r--r--  1 root root 777 2004-12-17 10:23 CyrusReq.pem
khazad-dum:/etc/ssl#
```

Diese Zertifikatsanfrage liegt in `CyrusReq.pem` und als 1 024-Bit-RSA-Schlüssel in `CyrusKey.pem`, beide im PEM-Format, vor. Vergiss auch hier Dein Passwort nicht, das Du als "PEM pass phrase" eingeben musst. Du brauchst es für den privaten Schlüssel des Servers und dann, wenn Du ihn ungültig machen oder das Zertifikat erneuern willst. Das "challenge password" kannst Du auch leer lassen. Sinn dessen (es taucht übrigens im *Klartext* im Attributsbereich des Requests auf) ist es, sich auch bei Verlust des privaten Schlüssels durch Angabe dieses Challenge Password als der korrekte Inhaber ausweisen zu können.

5. **Passwortschutz aus dem temporären Schlüssel entfernen.**

   Da der Cyrus den privaten Schlüssel braucht, aber Du beim Start des Servers natürlich nicht zehn Passwörter für die einzelnen Dienste einhacken willst (und kannst, wenn Du nicht vor Ort bist), *entfernst* Du jetzt den Passwortschutz aus dem privaten Schlüssel (hoffentlich hast Du Dir die "PEM pass phrase" vorhin gemerkt, die brauchst Du dazu nämlich):

   ```
   khazad-dum:/etc/ssl # openssl rsa < CyrusKey.pem > cyrus-key.pem
   ```

   Jetzt hast Du den ersten Teilerfolg: Du hältst einen schimmernden, ungesicherten Schlüssel in Deinen Patschhändchen. Pass gut darauf auf und sorge mittels

   ```
   khazad-dum:/etc/ssl # chmod go-rwx cyrus-key.pem
   ```

   dafür, dass der Schlüssel nicht für die ganze Welt lesbar ist. Ich betone nochmals: Er ist jetzt nicht mehr durch ein Passwort geschützt und kann von jedem verwendet werden, der ihn jetzt noch in seine gierigen Pfoten bekommt.

   Ach so: Den `CyrusKey.pem` hebst Du Dir in `/etc/ssl/private` auf. Du brauchst ihn zwar nicht mehr, aber was soll der Geiz.

6. **Server-Zertifikatsanfrage mit dem CA-Zertifikat signieren.**

Das ist der letzte Schritt. Danach hast Du auch Dein Server-Zertifikat in der Hand und kannst es mit Deinem Server zusammen einsetzen.

Server-Zertifikatsanfrage signieren

```
khazad-dum:/etc/ssl# openssl ca -name Server_CA -in CyrusReq.pem -out CyrusCert.pem
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for /etc/ssl/ServerCA/private/SCAkey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
localityName          :PRINTABLE:'Nuremberg'
organizationName      :PRINTABLE:'Kenobi Laboratories'
organizationalUnitName:T61STRING:'KenobiLab R&D'
commonName            :PRINTABLE:'imap.arda.lan'
emailAddress          :IA5STRING:'obiwan@meins.de'
Certificate is to be certified until Dec 16 09:36:46 2009 GMT (1825 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

khazad-dum:/etc/ssl# ls -l CyrusCert.pem
-rw-r--r--  1 root root 4920 2004-12-17 10:36 CyrusCert.pem
khazad-dum:/etc/ssl#
```

Dein Zertifikat liegt nun in `CyrusCert.pem`. Schütze auch dies vor neugierigen Blicken (und vor allem fremden Zugriff) durch

```
khazad-dum:/etc/ssl # chmod og-rwx CyrusCert.pem
```

Den `CyrusReq.pem` hebst Du Dir in `/etc/ssl/private` auf.

7. **Optional: Serverschlüssel und -Zertifikat in dieselbe Datei schreiben.**

   Manche Server – der Cyrus IMAPd gehört *nicht* dazu – wollen Schlüssel und Zertifikat aus *derselben* Datei lesen. In dem Fall reicht es aus, beide Dateien mittels

   ```
   khazad-dum:/etc/ssl # cat key.pem certificate.pem > server.pem
   ```

   zu einer zusammenzuführen. Vergiss nicht das abschließende

   ```
   khazad-dum:/etc/ssl # chmod og-rwx server.pem
   ```

Das CA-Zertifikat kopierst Du nach `/etc/ssl/certs`, den nicht per Passwort gesicherten Schlüssel nach `/etc/ssl/private`.

Die Pfade des Zertifikats und des Schlüssels kannst Du nun in die Konfigurationsdatei Deines Servers einfügen.

Am Beispiel des Cyrus IMAPd: Ich habe das Cyrus-Zertifikat nach `/etc/ssl/certs` und den zugehörigen Schlüssel nach `/etc/ssl/private` kopiert. Lesen darf die beiden

Dateien ausschließlich Benutzer `cyrus`. Auf die Verzeichnisse haben alle Benutzer Lese- und Ausführungszugriff. Dann habe ich beide in `/etc/imapd.conf` eingetragen, den Cyrus IMAPd neu gestartet und mich darüber gefreut, dass es so prima tut.

Überprüfen lässt sich das am Beispiel des Cyrus IMAPd folgendermaßen, wobei man schön die Hierarchie Root-CA – Server-CA – Service sehen kann (ich habe die Zeilen geeignet umgebrochen):

```
khazad-dum:~ # openssl s_client -CApath /etc/ssl/certs -port 993 \
                                -host khazad-dum > /tmp/foo
depth=2 /C=DE/ST=Bavaria/L=Nuremberg/O=Kenobi Laboratories \
        /OU=KenobiLab Root-CA/CN=Obi-Wan Kenobi \
        /emailAddress=obiwan@meins.de
verify return:1
depth=1 /C=DE/ST=Bavaria/L=Nuremberg/O=Kenobi Laboratories \
        /OU=KenobiLab Server-CA/CN=Obi-Wan Kenobi \
        /emailAddress=obiwan@meins.de
verify return:1
depth=0 /C=DE/ST=Bavaria/L=Nuremberg/O=Kenobi Laboratories \
        /OU=KenobiLab R&D/CN=imap.arda.lan \
        /emailAddress=obiwan@meins.de
verify return:1
^C
khazad-dum:~ #
```

Sauberer wäre es, alle Server, die auf die Zertifikate zugreifen dürfen, in eine zusätzliche Gruppe zu packen und dieser – neben `root` natürlich – Zugriff darauf einzuräumen.

Halt, da war ja noch was. Du erinnerst Dich, dass ich Dir noch verraten wollte, wie man die Zertifikate in Deinen Mail-Client importiert? Sinn des Ganzen ist es, dass Dein Mail-Client (im Beispiel verwende ich mal MOZILLA MAIL) keine Meldung *„Website certified by an unknown authority“* bringt. Dazu musst Du nicht nur das Server-, sondern auch die CA-Zertifikate in Deinen Mail-Client schaffen – denn wie soll er dem Server-Zertifikat trauen, wenn er die ausgebende CA nicht kennt?

Dazu gibst Du im Verzeichnis `/etc/ssl` das folgende Kommando ein:

```
khazad-dum:~ # openssl pkcs12 -export -chain -nokeys \
                -name "'imap.arda.lan"' \
                -in certs/CyrusCert.pem \
                -inkey private/CyrusKey.pem \
                -out /tmp/imap.p12
```

Die Option `-chain` exportiert nicht nur das Zertifikat für `imap.arda.lan`, sondern auch die gesamte Zertifikatshierarchie „darüber“. Die Datei `/tmp/imap.p12` kopierst Du in ein Verzeichnis, das Dein Mail-Client erreichen kann (mail es Dir, falls Du es nicht mit *cp* oder *scp* auf die Zielmaschine schaffen kannst). Dann importierst Du es in Deinen Mail-Client.

Wie man das macht, zeige ich hier am Beispiel des Mail-Clients aus der Mozilla-Suite, und zwar in der englischsprachigen Fassung: Ruf die Dialogbox auf, die Du mittels

`Edit` ⇒ `Preferences` ⇒ `Privacy & Security` ⇒ `Certificates` erreichst. Darin findest Du einen Button `Manage Certificates...`. Wenn Du diesen drückst, erscheint der Zertifikatsmanager von Mozilla Mail. In diesem findest Du einen Button `Import`, über die Du die Datei `imap.p12` importieren kannst. Wenn Du beim Export ein Passwort vorgegeben hast, musst Du dieses beim Import angeben. Wenn das geklappt hat, taucht gleich auf der ersten Karteikarte das Zertifikat für `imap.arda.lan` auf. Dann solltest Du noch die Karteikarte „`Authorities`“ aufrufen und dort nach „`Kenobi Laboratories`“ suchen. Dort findest Du zwei Zertifikate vor: Je eines für unsere Root- und unsere Server-CA. Mit dem Button „`Edit`“ solltest Du beide noch mit Vertrauen versehen, nachdem Du beide selektiert hast.

Literaturverzeichnis

[1] Marcel Hilzinger, „Qual der Wahl – Das optimale Dateisystem: Performance, Features, Optionen“, in: *Linux-Magazin*, Linux New Media AG, Ausgabe 11, 2004, S. 28ff.

[2] Ingmar Camphausen, Stefan Kelm, Britta Liedke, Lars Weber, „Aufbau und Betrieb einer Zertifizierungsinstanz. DFN-PCA Handbuch“, Berlin, DFN 2000, http://www.pca.dfn.de/certify/ssl/handbuch/

[3] OpenSSL-HowTo des Dept. of Genetics, Washington University School of Medicine, St. Louis, http://sapiens.wustl.edu/~sysmain/info/openssl/

Index

W

X

Z