3 Erste Schritte im neuen System

3.1 Arbeiten mit dem Server

Nachdem wir die Installation von FreeBSD gemeistert haben, können wir uns nun daranmachen, das System zu konfigurieren, und ihm die Aufgaben zuweisen, die es in Zukunft für uns erledigen soll.

Wie ich Dir während der Installation bereits verraten habe, erlaubt die Standardkonfiguration von `sshd` unter FreeBSD die Anmeldung mit dem Superuser `root` nicht. Dafür haben wir einen normalen Systembenutzer angelegt, der der Gruppe `wheel` angehört, sodass wir mit dem Befehl `su` in den Superuser-Modus wechseln können. Eine Alternative zu `su`, `sudo`, lernen wir in diesem Kapitel ebenfalls kennen.

FreeBSD wird – wie Linux-Systeme auch – über die Konsole (auch Shell genannt) gesteuert. Grafische Oberflächen wie Gnome oder KDE haben auf einem Server nichts verloren, da sie unnötig Ressourcen verbrauchen bzw. ein Sicherheitsrisiko darstellen können. Um uns die Arbeit mit der Konsole so angenehm wie möglich zu gestalten, werden wir später unsere Shell etwas anpassen.

Ein erster Schritt wird nun sein, uns mithilfe eines SSH-Clients mit dem Server zu verbinden. Für Microsoft Windows gibt es den sehr guten Client PuTTY, unter Mac OS und Linux ist ein SSH-Client bereits installiert.

Mit folgendem Befehl stellen wir unter Mac OS oder Linux eine SSH-Verbindung zu unserem frisch installierten Server her:

```
# ssh -p <PORT> <USER>@<HOST>
```

Der Teil `-p <PORT>` ist dabei optional. Wenn Du den Parameter `-p` weglässt, wird automatisch der Standard-SSH-Port `22` verwendet. Im Laufe dieses Buchs werden wir diesen Port ändern. Dann wird dieser Parameter für Dich an Bedeutung gewinnen.

`<USER>` musst Du durch den Namen des neu angelegten Benutzers und `<HOST>` durch die IP-Adresse bzw. Domain des Servers ersetzen. Falls Du PuTTY verwendest, musst Du diese Angaben in die erforderlichen Felder der Eingabemaske eintragen.

War der Verbindungsaufbau schließlich erfolgreich, wirst Du aufgefordert, Dein Passwort einzugeben. Während der Eingabe ändert sich die Eingabezeile nicht, was viele beim ersten Mal verwundert. Tatsächlich handelt es sich aber um ein Sicherheitsfeature, da ein Zuschauer auf diese Weise nicht sehen kann, wie lang Dein Passwort ist.

PuTTY: http://www.putty.org/

3.2 Das richtige Passwort

Die Wahl des richtigen Passworts gehört zu den Kernelementen eines Sicherheitskonzepts. Oft ist das Passwort einerseits zwar leicht zu merken, andererseits aber dadurch auch leicht zu erraten.

Es gibt aber auch Passwörter, die zwar nicht leicht zu erraten, oft aber so kompliziert sind, dass sie aufgeschrieben werden müssen, um nicht vergessen zu werden: Das stellt allerdings wieder ein Sicherheitsrisiko dar.

Absolut empfehlenswert ist daher der Einsatz einer Passwort-Verwaltungssoftware wie beispielsweise KeePass. So musst Du Dir nur ein Passwort merken (und das muss entsprechend komplex sein).

Um ein Passwort zu erstellen, das gut zu merken, gleichzeitig aber auch ausreichend komplex ist, gibt es einen einfachen Trick. Überlege Dir einen einfachen Satz, einen Vers oder ein Zitat, beispielsweise »Ein Hut, ein Stock, ein Regenschirm.«.

Als Passwort verwendest Du nun einfach die Anfangsbuchstaben der einzelnen Wörter sowie die Satzzeichen. Damit ergibt sich in diesem Beispiel folgendes Passwort: EH,eS,eR.

Dabei gilt: je länger der Satz, desto sicherer das Passwort.

Hinweis: KeePass bietet Dir die Möglichkeit, komplexe Passwörter zu generieren.

KeePass: http://keepass.info/

3.3 Die Verzeichnisstruktur von FreeBSD

Im Einleitungskapitel habe ich bereits auf die Konsistenz von FreeBSD hingewiesen. Diese wird deutlicher, wenn wir uns die Verzeichnisstruktur auszugsweise ansehen.

/boot

Hier liegen der Kernel und die Dateien, die für den Systemstart erforderlich sind. Später werden wir uns hier speziell die Datei `loader.conf` näher ansehen, mit deren Hilfe wir den Bootprozess beeinflussen können. Verwechsle dieses Verzeichnis nicht mit der Partition vom Typ *freebsd-boot*.

/root

Dies ist das Verzeichnis des Benutzers `root`, der auf allen UNIX-artigen Systemen den Superuser darstellt.

/bin

In diesem Verzeichnis liegen Systemprogramme, die von allen Benutzern und natürlich von `root` ausgeführt werden können, beispielsweise das Programm zum Wechseln von Verzeichnissen: `cd`.

/sbin

Dieses Verzeichnis beherbergt sämtliche Systemprogramme, die nur vom Benutzer `root` ausgeführt werden können.

/lib

In diesem Verzeichnis sind gemeinsame Systembibliotheken abgelegt.

/var

Dies ist der Ort, an dem Prozesse und Programme Log-Dateien (`/var/log`) und sogenannte »Prozesssperren« (.pid-Dateien) ablegen (`/var/run`).

/etc

Systemkonfigurationsdateien werden in diesem Verzeichnis abgelegt. Besonders wichtig ist unter FreeBSD die zentrale Steuerungsdatei `/etc/rc.conf`, in der unter anderem die zu startenden Dienste oder auch der Hostname definiert werden.

/rescue

Sollte eine FreeBSD-Installation reparaturbedürftig sein, findest Du in diesem Verzeichnis Programme, die Dir bei der Fehlerbeseitigung helfen. Diese sind so kompiliert, dass sie auch auf beschädigten Installationen ausgeführt werden können.

/tmp

In diesem Verzeichnis werden temporäre Dateien abgelegt. Verwechsle dieses Verzeichnis nicht mit der Partition vom Typ *freebsd-swap*, die lediglich für die Auslagerung von Daten aus dem Arbeitsspeicher verwendet wird.

/usr

Unterhalb dieses Verzeichnisses werden Benutzerdaten abgelegt, die unabhängig vom Basissystem sind (Applikationen, Homeverzeichnisse etc.).

/usr/home

Hier liegen die Verzeichnisse der Systembenutzer (außer das von `root`). Ein System-Link zeigt von `/home/` auf `/usr/home/`.

/usr/lib, /usr/bin, /usr/sbin

Dies sind die Äquivalente zu ihren Namensvettern des Basissystems, allerdings sind diese nicht für das Basissystem relevant.

/usr/local

In diesem Verzeichnis wird Zusatzsoftware abgelegt, beispielsweise der Webserver oder Datenbankserver etc.

/usr/local/lib, /usr/local/bin, /usr/local/sbin

Auch diese Verzeichnisse sind gleichbedeutend mit ihren Namensvettern, allerdings sind diese für installierte Anwendungen und Dienste relevant (z. B. Webserver).

/usr/local/etc

Hier liegen die Konfigurationsdateien der installierten Applikationen (vgl. `/etc/`).

Handbuch: http://freebsd.org/doc/de/books/handbook/dirstructure.html

3.4 Die Editoren »vi« und »ee«

FreeBSD wird mit zwei Texteditoren ausgeliefert: `vi` und `ee`. Da wir sehr viele Dateien bearbeiten werden, sollten wir uns mit diesen Werkzeugen vertraut machen. Deren Bedienung schauen wir uns daher nachfolgend kurz an.

3.4.1 vi

`vi` ist der wohl bekannteste Editor für die Konsole. Zwar ist er sehr leistungsfähig, allerdings ist die Bedienung etwas gewöhnungsbedürftig. Daher gebe ich Dir hier einen kurzen Überblick über die wichtigsten Befehle.

Um eine Datei mit vi zu öffnen, reicht der folgende Befehl:

```
# vi <DATEINAME>
```

`vi` öffnet Dateien standardmäßig im Lesemodus. Um den Änderungsmodus zu aktivieren, drückst Du einfach die Taste `a`. Um diesen Modus wieder zu verlassen, musst Du die `ESC`-Taste drücken.

> **Hinweis:** <ESC> bzw. <ENTER> sind hier keine Eingaben, sondern Tastenbezeichnungen Deiner Tastatur.

Befehle in `vi` beginnen immer mit einem Doppelpunkt. Folgende Befehle sind fürs Erste von Bedeutung:

vi beenden

```
:q <ENTER>
```

vi beenden und Änderungen speichern

```
:wq <ENTER>
```

vi beenden und Änderungen verwerfen

```
:q! <ENTER>
```

In diesem Buch werde ich – so weit wie möglich – den Editor `ee` verwenden, da mir die Bedienung besser gefällt. Das ist allerdings Geschmacksache und nicht wirklich relevant.

---

vi-Cheatsheet: http://www.lagmonster.org/docs/vi.html

3.4.2 ee

Um eine Datei in ee zu öffnen, verwenden wir die gleiche Syntax wie auch bei vi:

```
# ee <DATEINAME>
```

ee öffnet Dateien dabei automatisch im Schreibmodus. Der Vorteil von ee ist, dass im oberen Bereich permanent eine Befehlsübersicht eingeblendet ist. Das Dach (^) vor dem Kürzel steht dabei für die Taste STRG.

ee beenden

```
<ESC> <ENTER>
```

Wurden Änderungen vorgenommen, fragt ee beim Beenden nach, ob diese gespeichert werden sollen oder nicht.

ee beenden und Änderungen speichern

```
<ESC> <ENTER> <ENTER>
```

ee beenden und Änderungen verwerfen

```
<ESC> <ENTER> b <ENTER>
```

Alternativ kannst Du auch einfach den Aufforderungen auf dem Bildschirm folgen, denn ee hat ein Menü-System, das sich über Tastenbefehle gut bedienen lässt.

3.5 »sudo« – weil es nicht immer root sein muss

Hinweis: In diesem Kapitel greife ich bereits etwas voraus. Es kommen Dinge zur Sprache, die Dir im Moment vielleicht noch nicht viel sagen. Gegebenenfalls kannst Du dieses Kapitel daher überspringen und zu einem späteren Zeitpunkt durcharbeiten.

Systemweite Einstellungen kann aufgrund des Sicherheitskonzepts von FreeBSD nur der Benutzer root vornehmen. Regelmäßig als Superuser zu arbeiten ist allerdings nicht ratsam. Es kann aber natürlich Situationen geben, in denen die Nutzung von mehr als einem Benutzer mit root-Rechten sinnvoll oder gar erforderlich ist, beispielsweise dann, wenn ein Benutzer die Berechtigung haben soll, den Webserver neu zu starten, nicht aber das gesamte System.

Um mit einem normalen Systembenutzer Änderungen an Konfigurationsdateien vornehmen zu können, was ihm grundsätzlich untersagt ist, gibt es das Tool sudo. Der Name dieses Programms ist die Kurzfassung von »substitute user do«, was so viel wie »Tu es als jemand anderes« – in der Regel als Superuser – bedeutet.

Dieses Tool ermöglicht es einem normalen Systembenutzer, Befehle mit root-Rechten auszuführen, ohne dabei root sein oder dessen Passwort kennen zu müssen. Mit sudo ist es möglich, diese Berechtigungen auf bestimmte Befehle einzuschränken.

sudo ist in einem Standard-FreeBSD-System nicht installiert, daher müssen wir das jetzt nachholen.

Hinweis: Bis zu diesem Zeitpunkt haben wir uns noch nicht angesehen, wie Software auf einem FreeBSD-System installiert wird. Falls Du es nicht schon weißt, lies Dir bitte Kapitel 6, »Software installieren«, durch.

Der Portname von sudo lautet security/sudo. Dieser kann mithilfe des folgenden Befehls installiert werden:

```
# cd /usr/ports/security/sudo && make install clean
```

sudo konfigurieren

sudo wird mithilfe des sudo-Editors visudo konfiguriert, der – wie der Name bereits verrät – genauso bedient wird wie vi selbst. Mit folgendem Befehl öffnen wir die Konfigurationsdatei von sudo und passen die Einstellungen so an, dass unser normaler Systembenutzer <USER>-Befehle mit root-Rechten ausführen kann:

```
# visudo -f /usr/local/etc/sudoers
```

Die folgende Zeile bewirkt, dass der Benutzer <USER> sämtliche Befehle ohne Eingabe seines Passworts mit root-Rechten ausführen darf:

```
<USER> ALL = (ALL) NOPASSWD: ALL
```

Falls Du einem Benutzer nur gestatten willst, einen ganz bestimmten Befehl mit root-Rechten auszuführen, dann kannst Du folgende Zeile als Vorlage verwenden:

```
<USER> ALL = NOPASSWD: <PFAD/ZUM/PROGRAMM>
```

Den absoluten Pfad zu dem Programm kannst Du mithilfe des folgenden Befehls ermitteln:

```
# whereis <PROGRAMM>
```

Was aber tun, wenn Du nicht nur einem Benutzer entsprechende Rechte einräumen willst, sondern eine ganze Benutzergruppe mit root-Rechten per sudo ausstatten willst? In diesem Fall folgst Du der gleichen Syntax, stellst dem Gruppennamen allerdings ein Prozentzeichen voran, um den Namen als Gruppenname zu identifizieren.

```
%<GRUPPE> ALL = (ALL) NOPASSWD: ALL
```

Mitglieder der Gruppe <GRUPPE> können nun jeden Befehl mit den Rechten des Superusers ausführen, indem sie dem Aufruf einfach sudo voranstellen.

```
# sudo <BEFEHL>
```

Falls Dich das ewige Voranstellen von sudo nervt, kannst Du auch einfach in eine temporäre root-Shell ähnlich su wechseln, indem Du

```
# sudo -s
```

ausführst.

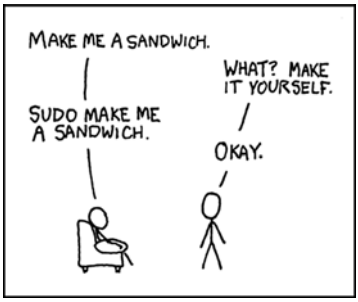

Abb. 3–1 *Quelle: http://xkcd.com/149/*

Hinweis: Sofern nicht anders angegeben, werden in diesem Buch sämtliche Arbeiten am Server mit root-Rechten durchgeführt. Ob Du dabei mit sudo, su oder sudo -s arbeitest, ist egal.

3.6 Die Shell anpassen

Die Shell oder auch Konsole ist die Arbeitsumgebung eines Benutzers auf dem Server. Sie unterstützt beim Navigieren durch das System und kann viele Informationen bereithalten, die Dir die Arbeit erleichtern. Da jeder Benutzer anders arbeitet, lässt sich auch eine Shell an individuelle Bedürfnisse anpassen.

Auf einem FreeBSD-System sind bereits mehrere Shells installiert und stehen uns zur freien Verfügung. Jeder Benutzer kann selbst wählen, welche er verwenden möchte. Mein Favorit ist die tcsh.

Im ersten Schritt passen wir unseren Prompt an. Das ist der Teil, mit dem die Eingabezeile beginnt. Ich werde mir meinen tcsh-Shell-Prompt wie folgt konfigurieren.

Für den normalen Benutzer:

```
<HOST> <UHRZEIT> <PFAD> >
```

Für root:

```
[R] <HOST>:<PFAD> #
```

<HOST> zeigt mir den Hostnamen des Systems an, <UHRZEIT> steht logischerweise für die aktuelle Zeit, und <PFAD> wird durch den Pfad ersetzt, in dem ich mich gerade befinde. Das [R] deutet darauf hin, dass ich gerade als root arbeite.

Die Shell wird über eine Konfigurationsdatei mit dem Namen .tcshrc im Homeverzeichnis des jeweiligen Benutzers gesteuert. Diese Datei existiert in Deinem Homeverzeichnis noch nicht, Du kannst aber die Datei .cshrc aus dem Wurzelverzeichnis (nicht /root) als Vorlage verwenden, in das Homeverzeichnis Deines Benutzers kopieren und in .tcshrc umbenennen.

Hinweis: Achte darauf, dass die Datei von dem betreffenden Benutzer gelesen werden kann. Falls Du nicht weißt, wie man das macht, lies Dir bitte jetzt Kapitel 4.2, »Das Berechtigungsmodel«, durch.

Um den Prompt (engl. »Eingabeaufforderung«) entsprechend anzupassen, suchen wir in der Datei .tcshrc Deines Homeverzeichnisses die Zeile

```
if ($?prompt) then
```

und schreiben eine der folgenden Zeilen in eine neue Zeile darunter.

Für den normalen Benutzer:

```
set prompt = "%B%m%b %t %~ %> "
```

Für root:

```
set prompt = "%B[R] %m:%~ %# "
```

Zusätzlich solltest Du die Tastenbelegung für die ENTF-Taste ändern. Standardmäßig erzeugt diese Taste eine Tilde (~). Um dies in das erwartete Verhalten zu ändern – sodass ENTF das Zeichen löscht, das vor dem Cursor steht – suche folgende Zeile in der Datei .tcshrc:

```
if ( $?tcsh ) then
```

Trage in diesen if-Block die folgende Anweisung ein:

```
bindkey ^[[3~ delete-char
```

Um die Übersichtlichkeit der Konsole zu verbessern und die Navigation in der Verzeichnisstruktur zu erleichtern, kannst Du folgende Parameter setzen, die die Ausgabe einfärben und Pfadangaben mithilfe der Tabulator-Taste automatisch vervollständigen.

```
set autolist
set color
set colorcat
```

Eine weitere sehr nützliche Funktion von Shells sind die sogenannten Aliase. Mit deren Hilfe lassen sich lange, teils komplexe Anweisungen verkürzen.

Ein Beispiel: Der Befehl `ls` listet unter anderem den Inhalt eines Verzeichnisses auf. `ls` hat aber einige Parameter, die die Ausgabe erheblich anschaulicher gestalten können. Der Befehl

```
# ls -G
```

listet dabei den Inhalt des aktuellen Verzeichnisses auf und bereitet ihn farblich auf. Über einen Alias können wir nun erreichen, dass `ls` immer `ls -G` ausführt.

Einen Alias anzulegen, ist sehr einfach. Auch hierfür bearbeiten wir die Datei `.tcshrc` im Homeverzeichnis des betreffenden Benutzers und tragen folgende Zeile ein, wobei die Syntax für andere Aliase übernommen werden kann.

```
alias ls ls –G
```

Aus Sicherheitsgründen empfiehlt es sich, einen weiteren Parameter in die `tcsh`-Konfiguration einzutragen, nämlich `savehist`. Dieser Parameter gibt an, ob die Historie Deiner Eingaben bis zum nächsten Anmelden gespeichert werden soll oder nicht. Diesen Parameter setzt Du wie folgt unter die `set prompt`-Anweisung:

```
set savehist = 0
```

Anschließend müssen wir unserem Benutzer die `tcsh`-Shell als neue Standard-Shell zuweisen. Das geht am einfachsten mit folgendem Befehl, den Du als der Benutzer ausführen musst, für den die Änderung vorgenommen werden soll:

```
# chsh -s tcsh
```

Bei der nächsten Anmeldung wird die neue Konfiguration geladen und Du kannst Deine neue Shell begutachten.

3.7 SSH absichern

SSH ist der Dienst, über den wir uns von jedem Ort aus mit dem Server verbinden und ihm Befehle erteilen können. Aus diesem Grund ist er immer wieder Ziel von Angriffen, um die Zugangsdaten von Systembenutzern oder gar von `root` zu erraten, womit erheblicher Schaden angerichtet werden könnte.

Daher müssen wir ein besonderes Augenmerk auf die Sicherheit dieses Dienstes richten. Wir müssen aber zwischen gefühlter und tatsächlicher Sicherheit unterscheiden.

3.7.1 Gefühlte Sicherheit erhöhen

Um den SSH-Daemon gefühlt sicherer zu machen, ändern wir den Port, auf dem er lauscht. Standardmäßig ist das der Port 22. Da das aber nun einmal bekannt ist, wissen Bösewichte, welchen Port sie zu attackieren haben.

Viele Attacken sind keine gezielten Angriffe und werden in Fachkreisen daher als »Netzrauschen« bezeichnet. Dabei handelt es sich primär um einfache Programme, die mithilfe von Wörterbüchern ganze IP-Bereiche auf verwundbare SSH-Instanzen untersuchen.

Eine einfache Möglichkeit diesen Scannern aus dem Weg zu gehen, ist das Ändern des sshd-Ports. Gegen gezielte Angriffe ist dies allerdings kein Schutz, da es ein Leichtes ist, den korrekten Port herauszufinden.

Hinweis: Falls Du den Port nicht ändern willst, solltest Du Dir in Kapitel 7, »Die Firewall konfigurieren – Spezielle pf-Konfigurationen«, das Tool ssh-guard anschauen. Es hilft Dir dabei, Brute-Force-Attacken zu erkennen und abzuwehren.

Um den Port nun zu ändern, bearbeiten wir die Konfigurationsdatei von sshd und ändern den Parameter Port von 22 auf einen Wert unserer Wahl. Wir sollten aber darauf achten, dass wir keinen Port verwenden, den eine später von uns eingesetzte Software belegt.

```
# ee /etc/ssh/sshd_config
```

Vorher:

```
Port 22
```

Nachher:

```
Port 4711
```

Es kann sein, dass der Parameter Port mit einer vorangestellten Raute (#) kommentiert ist. Diese musst Du natürlich entfernen, um den Standardport zu überschreiben. Ich habe hier als Beispiel den Port 4711 gewählt.

3.7.2 Tatsächliche Sicherheit erhöhen

Da wir nun schon die Konfigurationsdatei im Editor geöffnet haben, ändern wir noch weitere Parameter, die die Sicherheit von sshd erhöhen.

Im ersten Schritt sollten wir noch die IP-Adresse angeben, auf der sshd lauschen soll. Das erhöht zwar nicht direkt die Sicherheit, wird uns aber im weiteren Verlauf die eine oder andere Fehlermeldung ersparen, wenn wir weitere IP-Adressen hinzufügen.

Dafür ändern wir den Parameter ListenAddress auf

```
ListenAddress <IP-ADRESSE>
```

wobei Du <IP-ADRESSE> durch die IP-Adresse ersetzen musst, unter der Dein Server erreichbar ist.

Der Parameter PermitRootLogin sollte bereits auf no stehen. Das sollte auch so bleiben. Damit verhinderst Du, dass sich root per SSH anmelden kann.

Ein Parameter, den Du unbedingt anpassen solltest, ist AllowUsers. Diesem kannst Du einen Benutzernamen oder eine durch Leerzeichen getrennte Liste von Benutzernamen übergeben. Mithilfe von AllowUsers kannst Du festlegen, welche Benutzer sich per SSH auf Deinem System anmelden dürfen.

```
AllowUsers <BENUTZER1> <BENUTZER2>
```

Hinweis: Es gibt auch einen Parameter AllowGroups. Dieser hat genau die gleiche Funktion wie AllowUsers; hier gibst Du logischerweise statt einzelner Benutzer ganze Benutzergruppen an.

Nachdem Du die Änderungen gespeichert hast, musst Du sshd neu starten, was der folgende Befehl bewirkt. Da sshd unter FreeBSD als Systembestandteil betrachtet wird, befinden sich die Konfigurationsdateien im Systembereich unter /etc/ und nicht unter /usr/local/etc/.

```
# /etc/rc.d/sshd restart
```

Solltest Du Änderungen an der Konfiguration dieses Dienstes vornehmen, empfehle ich Dir, die aktuelle Verbindung aufrechtzuerhalten und in einer neuen Sitzung die geänderte Konfiguration zu testen. Die Änderungen sind nur für neue Sitzungen wirksam, sodass Du im Fehlerfall die Möglichkeit hast, Korrekturen vorzunehmen.

3.8 Zeitsynchronisation per NTP

NTP ist die Abkürzung für »Network Time Protocol«, ein Protokol, das zur Synchronisierung von Uhren – beispielsweise der Systemuhr unseres Servers – dient.
Auf einem Server ist die korrekte Systemzeit äußerst hilfreich und wichtig. Daher konfigurieren wir auf unserem Server einen NTP-Client, der die Systemzeit von einem externen Zeitserver bezieht.

Hinweis: Du kannst auch selbst einen Zeitserver betreiben und für andere Systeme im Netzwerk bereitstellen. Dies ist allerdings nicht Teil dieses Buchs.

Wir werden unseren Server später mit mehreren IP-Adressen ausstatten und in mehrere »Untersysteme« (siehe Kapitel 8, »Arbeiten mit Jails«) aufteilen. Wir wollen dabei allerdings vermeiden, dass Dienste (sog. »Daemons«) sich an alle Adressen binden, die sie finden können.

Aus diesem Grunde installieren wir OpenNTPD, das nicht Bestandteil von FreeBSD ist. Es kann sich an eine einzige IP-Adresse – nämlich die unseres Loopback-Interface – binden.

Hinweis: An dieser Stelle möchte ich Dich noch einmal auf Kapitel 6, »Software installieren«, verweisen.

Den OpenNTPD-Port `net/openntpd` kannst Du mithilfe des folgenden Befehls installieren:

```
# cd /usr/ports/net/openntpd && make install clean
```

Die Konfigurationsdatei für OpenNTPD, `ntpd.conf`, befindet sich im Verzeichnis `/usr/local/etc/` und muss von uns noch angepasst werden.

Zunächst binden wir diesen Dienst an unser Loopback-Interface `lo0`, dem standardmäßig die Adresse `127.0.0.1` zugewiesen ist.

```
listen on 127.0.0.1
```

Anschließend teilen wir OpenNTPD mit, von welchen Serverpools es regelmäßig die aktuelle Uhrzeit beziehen soll:

```
server0.de.pool.ntp.org
server1.de.pool.ntp.org
server2.de.pool.ntp.org
server3.de.pool.ntp.org
```

Die Konfigurationsdatei hat im Endeffekt folgenden Inhalt:

```
listen on 127.0.0.1
server0.de.pool.ntp.org
server1.de.pool.ntp.org
server2.de.pool.ntp.org
server3.de.pool.ntp.org
```

OpenNTPD korrigiert die Uhrzeit nicht, wenn die Abweichung zum Serverpool größer als 1000 Sekunden ist. Daher korrigieren wir die Uhrzeit einmalig manuell mit folgendem Befehl:

```
# /usr/sbin/ntpd -g -q
```

Nachdem die Uhrzeit nun korrekt ist, aktivieren wir OpenNTPD, sodass es auch beim Systemstart automatisch mit gestartet wird.

Wie ich bereits bei der Vorstellung der Verzeichnisstruktur erwähnt habe, ist die Datei `/etc/rc.conf` die zentrale Konfigurationsdatei von FreeBSD. Hier müssen wir nun auch OpenNTPD aktivieren, indem wir die folgende Zeile einfügen:

```
openntpd_enable="YES"
```

Hinweis: Achte darauf, dass die Zeile keine Leerzeichen enthält, auch nicht zwischen »enable« und »=«, sonst wird das System den Parameter nicht interpretieren können.

Jetzt können wir OpenNTPD mit folgendem Befehl starten:

```
# /usr/local/etc/rc.d/openntpd.sh start
```

Wie bei fast allen Diensten kannst Du dessen aktuellen Status abfragen, indem Du den Parameter `start` durch `status` ersetzt:

```
# /usr/local/etc/rc.d/openntpd.sh status
```

OpenNTPD: http://www.openntpd.org/

3.9 E-Mails für root an ein Postfach weiterleiten

In regelmäßigen Abständen erstellt FreeBSD verschiedenste Berichte und schickt diese an den Benutzer `root`. Da wir über wichtige Ereignisse aber möglichst umgehend informiert werden möchten und Nachrichten nicht selbstständig vom Server abholen wollen, können wir alle Nachrichten an Systembenutzer an externe E-Mail-Adressen weiterleiten.

Diese Einstellungen nehmen wir in der Datei `/etc/mail/aliases` vor. In ihr wird definiert, wohin Nachrichten an den jeweiligen Alias umgeleitet werden sollen. Dort finden sich beispielsweise folgende Einträge:

```
(...)
# root: me@my.domain
MAILER-DAEMON: postmaster
postmaster: root
(...)
```

Diese besagen, dass Nachrichten an den Alias `postmaster` an `root` umgeleitet werden. Für `root` ist kein Alias definiert, weshalb E-Mails in das lokale Postfach zugestellt werden.

Wenn wir nun die Raute (#) vor `root` entfernen und `me@my.domain` durch unsere E-Mail-Adresse ersetzen, werden Nachrichten umgeleitet. Das sähe dann beispielsweise so aus:

```
(...)
root: benutzer@example.com
(...)
```

Bevor diese Weiterleitung allerdings aktiv wird, muss die Datei `/etc/mail/aliases` übersetzt werden. Das erreichen wir mithilfe des folgenden Befehls:

```
# newaliases
```

Anschließend sind diese Regeln aktiv. Von nun an werden E-Mails an `root` an die angegebene Adresse weitergeleitet.

3.10 Zusammenfassung

In diesem Kapitel hast Du Dich hoffentlich schon etwas mit FreeBSD und ein paar der mitgelieferten Werkzeuge vertraut gemacht. Den Aufbau eines Systems zu verinnerlichen ist essenziell. Im Störungsfall solltest Du wissen, wo Du suchen bzw. eingreifen musst.

Den Grundstein für ein sicheres System haben wir ebenfalls gelegt. Ein zentrales Einfallstor für Angreifer – den SSH-Daemon – haben wir bereits grundlegend abgesichert, und wir haben den Systembenutzer mit einem sicheren Passwort versehen.

Im nächsten Abschnitt werden wir uns weiter mit dem Thema Sicherheit, nämlich dem Berechtigungsmodell von FreeBSD, beschäftigen. Dieses kennst Du vielleicht bereits von anderen UNIX-artigen Systemen, die das gleiche Modell verwenden.