Einrichten einer Firewall unter OpenBSD

Fabian Heusser

April 2002, V 0.1

Dieses Dokument wurde mit LaTeX gesetzt. Es ist als Quelltext und im PDF-Format online erhältlich:

<http://www.w3p.ch/>

Der Autor bedankt sich bei Pascal Näf für Tips, Anmerkungen und Korrekturen.

Inhaltsverzeichnis

Zusammenfassung

Dieses Dokument behandelt das Installieren einer Firewall mit OpenBSD 3.0. Zuerst werden einige Hinweise zur Installation geliefert. Danach wird die Konfiguration der Netzwerkkarten und das Einrichten des Packet Filters beschrieben. Im 2. Teil geht es darum, einem Angreifer möglichst wenig Angriffsfläche zu liefern. Wir entfernen nicht benötigte Benutzer und Dienste, passen die Datei-Berechtigungen an und restriktieren den Zugriff über SSH. Zuletzt versetzten wir unser Firewall in den 'Securelevel 2' und markieren die Schlüsseldatein so, dass sie nicht mehr verändert werden können.

1 Einleitung

1.1 Vorausetzungen

Dieses Dokument setzt mässige Kenntnisse mit UNIX-Betribssystemen und wenig Erfahrung mit OpenBSD voraus. So wird der Installationsprozess nicht im Detail erläutert.

1.2 Weitere Informationen

Viele weitere Informationen sind in der FAQ von OpenBSD [1] nachzulesen. Der englische Artikel 'Hardening OpenBSD Internet Servers' auf Geodsoft Homepage [2] diente als Grundlage für den 2. Teil. Dort findet man noch viele weitere Informationen, so auch Grundkonzepte des Härten und eine Checkliste.

1.3 Vorgaben

Hier sehen wir die Netzwerkstruktur in welcher wir unser OpenBSD Firewall konfigurieren:

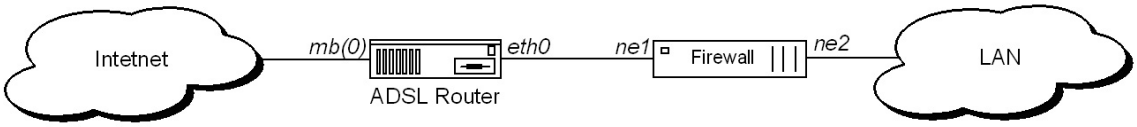

Abbildung 1: Teilnetz Topologie die wir einrichten

Der ADSL-Router übernimmt die PPPoE Kommunikation und die NAT (Network Address Translation aka Masquarading). DHCP ist nicht eingeschalten. Zugriffe aus dem Internet werden an 192.168.1.50, meinem Webserver, weitergeleitet.
Das Firewall muss die Daten filtern und transparent von ne1 nach ne2 weiterleiten.

Die Interfaces sind folgendermassen konfiguriert:

ne2 Dies ist die Netwerkkarte auf der Seite des LAN, der Vertrauenswürdigen Seite. Diese bekommt die IP 192.168.1.5

ne1 Dies ist die Netwerkkarte auf der Seite des WAN, d.h. des ADSL-Routers. Diese bekommt keine IP.

eth0 Ethernet Interface des ADSL-Routers, welches die IP 192.168.1.1 hat.

mb(0) WAN Interface des ADSL-Router, welches eine IP vom Provider bekommt.

Der Computer welchen wir als Firewall konfigurieren ist ein alter 486 'Compaq Prolinea 4/66' mit 40Mb RAM, 2 NE2000 Netzwerkkarten und 270 Mb Harddisk. Dies reicht für eine ADSL Verbindung durchaus. So ist die Maschine im Durchschnitt zu <3% ausgelastet. Auch die Harddisk ist nur zu 46% belegt, jedoch würde sie nicht reichen wenn man den Kernel neu Compilieren müsste.

2 Installation und Einrichten der Hardware

Da man ausschliessen will, dass das Firewall während der Konfiguration kompromittiert wird, ist es ratsahm das Firewall in einem Privaten LAN zu installieren, konfigurieren und testen.

2.1 Installation

Zuerst installiert man ein Minimum an OpenBSD von der CD oder von einem lokalen FTP Server. Es genügen die per Default angewählten Packete. Wenn Programme compiliert werden müssen, sollte man das Installieren des comp30.tgz Packetes noch in Betracht ziehen.

2.2 Konfiguration

Nach der Installation sollte man auf jeden Fall das root-Passwort ändern und die Systemzeit überprüfen, wie es in `afterboot(8)` beschrieben ist.

2.2.1 Kernel anpassen

Da OpenBSD bei meinen beiden Netzwerkkarten den IRQ und die Basisaddresse nicht richig erkannt hat, habe ich diese im User Kernel Mode (`USK>`)[1] angepasst. Dass diese Änderungen nun permanent vorhanden sind, erstellt man mittels `config(8)` einen neuen Kernel mit den richtigen Geräteparameter. Danach ersetzt man den alten Kernel `/bsd` mit dem neu erstellten. Wenn man zum Beispiel '`config -e -o /bsd.new /bsd`' ausführt, wird der neue Kernel in eine Datei `/bsd.new` geschrieben, welche man nun in `/bsd` (standard Bootkernel) umbenennt und die Rechte noch dem alten mit `chmod -x /bsd` angleicht.
Es ist ratsam den alten Kernel in `/bsd.old` umzubenennen, so kann man mit '`boot> /bsd.old`' mit dem alten Kernel booten.

2.2.2 Kontrolle der Netzwerk Interfaces

Nun überprüfen wir die Netzwerk-Konfiguration. Für jede Netzwerkkarte existiert eine Datei `hosname.if(5)`, wobei `if` die Bezeichnung der Netzwerkkarte ist, in unserem Fall 'ne1' oder 'ne2'.
Die Netzwerkkarte 'ne2' konfigurieren wir so, dass sie die IP 192.168.1.5 und die Netzwerkmaske 255.255.255.0 besitzt. `/etc/hostname.ne2` sieht dann so aus:

[1] In den USK kommt man wenn man beim Bootprompt '-c' eingibt.

```
inet 192.168.1.5 255.255.255.0 NONE
```

Die Netzwerkkarte auf der 'untrusted' Seite konfigurieren wir so, dass sie keine IP hat, `/etc/hostname.ne1` sieht dann so aus:

```
up
```

2.2.3 Bridge

Um das OpenBSD dazu zubringen die Packete von einer Netzwerkkarte auf die andere weiter zugeben, ist ein File `bridgename.if(5)` nötig. In unserem fall sieht die Datei `/etc/bridgename.bridge0` so aus:

```
add ne1
add ne2
up
```

2.2.4 Hostname und Kontrolle

Der Hostname kann man in der Datei `/etc/myname` dauerhaft setzen. Mehr informationen in der `|hostname(1)|`man Page. Der Default Gateway wird in der Datei `/etc/mygate/` gespeichert. Durch einen kleinen Bug in der Installations Prozedur kommt das File `/etc/hosts` nicht korrekt. Nach der IP kommt der Hostname und endet mit einem Punkt. Dieser muss noch um den Domainnamen ergänzt werden. Dann kommt noch einmal der Hostname, welcher korrekt ist. Unsere `/etc/hosts` Datei sieht nun folgender massen aus: 127.0.0.1 localhost 192.168.1.5 turgenev.w3p.net turgenev

2.2.5 Einschalten des Package forwarding

Um den Kernel überhaupt dazu zu bringen irgendwelche Packete von einem Interface zum anderen weiter zugeben, müssen wir noch die Kernel-Variable `net.inet.ip.forwarding` auf eins setzten. Dazu editiert man `/etc/sysctl.conf(5)` dem entsprechend.

Nach einem reboot oder '`route flush`' und '`sh -x /etc/netstart`' kann man, wie in `afterboot(8)` beschrieben, mit `ifconfig -a` und `netstat -rn` die gemachten Konfigurationen überprüfen. So sollten bei `ifconfig(8)` die beiden Netzwerkkarten richtig drin sein und die Bridge den Eintrag `<UP,RUNNING>` haben.

2.2.6 Test der Bridge

Wenn die obige Konfiguration erfolgreich war, sollten wir jetzt durch unseren Host hindurch Pingen können, d.h ein Host an ne1 kann ein Host an ne2 Pingen. Auch können wir unser Firewall anpingen.

2.2.7 Steahlt Firewall

Hätten wir in `/etc/hostname.ne2` nur `up` eingetragen, könnten wir unseren Rechner nur noch von der Konsole aus konfigurieren da man mit dem IP-protokoll keine Packete mehr auf diesen Rechner senden kann. Dieses erhöht die Sicherheit, mit der Einschränkung das Firewall nicht mehr Fernwarten zu können. Ein weiterer Vorteil ist, da unser Firewall sich wie ein Stück intelligentes Kabel verhält, dass sich der Integrationsaufwand verringert, da keine IP's reserviert und konfiguriert werden müssen.

2.3 Einrichten und Testen des Firewalls

Von unserer transparenten Bridge zum Stateful Firewall ist es nun nur noch ein kleiner Weg. Wir schalten den Packet Filter (pf(4) und pfctl(8)) ein indem wir die 'pf' Variable in `/etc/rc.conf(8)` auf "yes" setzen. Danach wird bei jedem Start das `/etc/pf.conf(5)` analysiert und geladen. Will man ohne Reboot das Config-File neu laden geschieht das mit `pfctl -F rules -R /etc/pf.conf`. Mit `pfctl -s rules` kann man das geladene Regel-File anzeigen; hängt man noch ein `-v` dran kommt zu jeder Regel noch eine Auswertungsstatistik. Die Log Datei des Packet Filters kann man mit `# tcpdump -netttr /var/log/pflog` anzeigen. Wenn man eine fortlaufende Anzeige, im Stil von `tail -f` möchte, erhält man mit `tcpdump -netttvi pflog0` die gewünschte Ausgabe. Einträge in die Log Datei generieren alle Regeln, welche das Schlüsselwort `log` enthalten (siehe unten).

2.3.1 Beispiel Konfiguration

Hier ein Beispiel für unsere Konfiguration:

```
### Excample Ruleset

### VARS
    lieb = "ne2"
    boese = "ne1"

    locallan="192.168.1.0/24"
    server="192.168.1.50/32"

    # all local address are spoofed
    spoofed="{10.0.0.0/12, 127.16.0.0/12, 192.168.0.0/16 }"

### GENERAL
    scrub in on $boese all min-ttl 3
    scrub in on $lieb all

    pass in  quick on $lieb all
    pass out quick on $lieb all

### LAN -> INET
    pass  out quick on $boese from $locallan to any flags S/SA keep state
```

```
### INET -> LAN
    block in log    on $boese all

    ## ANY -> SERVER
        pass in on $boese proto tcp from any  to $server port = 80 \
                                        Flags S/SA   modulate state

        pass in on $boese proto tcp from any  to $server port = 22 \
                                        Flags S/SA  modulate state

        pass in on $boese proto tcp from $hta to $server port = 2401 \
                                        Flags S/SA  modulate state

    ## SPOOFED -> ANY
        block in log on $boese from $spoofed to any
```

Eine sehr gute How-To zur Firewall Konfiguration findet man unter [3]

2.3.2 Testen der Firewall Funktionalität

Über dieses Thema sind sicher schon Bücher geschrieben worden, ich möchte das Thema hier jedoch nur anschneiden und einige Denkanstösse geben.
Eine der einfachsten Möglichkeiten eine Firewall zu testen ist sicher die Konfiguration Scanner/Sniffer. Auf der einen Seite der Firewall hat man einen Scanner, der der ganze Bereich scannt (TCP und UDP). Auf der anderen Seite plaziert man nun einen Sniffer, welcher alle Pakete anzeigt, die durch das Firewall gelangen. Dabei kann man je nach Fähikeiten des Scanners mehr oder weniger Test durchführen. So soll man das Firewall auch mit gespooften IP Addressen von privaten Netzwerkaddressräumen (z.B.: ne1 -> ne2 mit 10.0.0.1) scannen, um sicher zustellen, dass diese ihr Ziel nicht erreichen. Dies könnte dann von belangen sein, wenn man in der Konfiguration des Web Servers eine IP-Basierte Zugriffsregelung für gewisse Bereiche konfiguriert hat.
Daneben gibt es noch andere Aspekte wie:

- Was macht unsere Firewall mit IP-Options
- Was macht unsere Firewall mit Flags (DoS-, Synflood-Atacken)
- Was macht unsere Firewall mit fragmentierten Paketen
- Wie behandelt unsere Firewall Stateful Connections

Zu mindest die oberen 2 Punkte sind noch einfach zu testen, jedoch die unteren werden dann schon ein bisschen heftig.
Ich habe jetzt immer von Scanner und Sniffer gesprochen ohne konkrete Programme zu nennen, das will ich hier nachholen:

Nmap: Ein sehr leistungsfähiger Open Source Scanner für *x Betribsysteme
`http://www.insecure.org/`

Ethereal: Ein Open Source Sniffer aka Networkanalizer für *x und Win32 Betriebsysteme
`http://www.ethereal.com/`

lcrzoex: Ein allgemeines Open Source Netwerk Tool, eigentlich eine Samlung von mehr als 300 Tools mit denen sich diverse tolle Sachen anstellen lassen. So lassen sich individuell verschiedene Packete zusammenstellen, Scannen, Sniffen, Spoofen, ... Auch es ist für *x wie für Win32 Platformen geeignet.
`http://www.laurentconstantin.com/` .

Nmap hat eine Umfrage nach den besten 10 Tools für solche und ähnliche Aufgaben gestartet. Herausgekommen ist eine Liste der 50 Meistgenannten Programme, welche man unter [6] findet. So findet man HPing2 als equivalent für lcrzoex.

3 Das System widerstandsfähig machen

OpenBSD ist von Grund aus schon relativ sicher. Es lassen sich jedoch noch gewisse Optimierungen vornehmen. Dies wir das Thema in diesem Abschnitt sein. So wird das Handling mit Benutzer, Berechtigungen, Diensten und dem Securelevel 2 näher erläutert.
Die hier vorgestellten Techniken sind welche mit relativ wenig Aufwand einen grossen Nutzen bringen. Neben den hier Vorgestellten sind noch weitere Techniken auf [2] zu finden, so auch die Grundsätze des Härten.

3.1 Benutzer und Passwörter

Hier gilt es vorallem die Benutzerzahl auf einem Minimum zu halten. Dies hilft dem Administrator die Übersicht zu behalten. So bemerkt er er auch komische Veränderungen an unserem Firewall leichter. Die ausgeprägten Sicherheitsfunktionen von OpenBSD, wie der periodische Sicherheitscheck `security(8)`, überprüft täglich die `passwd(5)`- und `group(5)`-Dateien auf Syntax, leere Passwörter und andere Anomalien. Dies bietet jedoch keinen kompletten Schutz, da jemand der root Rechte hat auch das Recht hat dieses Script zu ändern. Dies wird wird ein Angreifer auch machen um seinen Aktivitäten zu tarnen.

3.1.1 Benutzer Anlegen

Für unsere Firewall erstellen wir lediglich noch einen 2. Benutzer der zum Fernwarten dient. Dieser muss in der Gruppe 'wheel' sein, so dass er `su(1)` benutzen kann.
Diesen Benutzer legen wir an, indem wir das Scrippt `adduser(8)` aufrufen, welches alle Details fragt. So kann man bei allen fragen die Vorgaben akteptieren. Ausser man hat andere Vorlieben. Danach geben wir den Benutzernamen ein. Bei der Angabe der Gruppe kann man das Erste akzeptieren und bei der Frage, ob man den Benutzer noch in eine weitere Gruppe einladen will, mit 'wheel' antworten. Deshalb weil nur Benutzer der Gruppe 'wheel' Berechtigung für `su(1)` haben.

Passwörter Danach kommt die Frage nach dem Passwort. Dies ist ein Thema für sich. Man liest viel was sichere Passwörter beinhalten sollen und was nicht. Folgt man diesen Regeln, kommt meist ein Passwort heraus, das man sich nicht mehr merken kann und irgendwo aufschreibt. Ziel verfehlt. Auf der Seiten der Geodsoft, hat es ein einfaches Perlskript, mit dem

sich gute Passwörter erzeugen lassen, die man sich auch merken kann. Dies findet man unter [4]. Es auch noch ein Password Checker unter [5], bei dem man die Kryptographische Qualität eines Passwortes bewerten lassen kann. Damit sollte man die Frage nach dem Passwort sicher ausreichend Beantworten können.
Danach hat man noch die Möglichkeit seine Eingaben noch einmal zu korrigieren. Somit hat man erfolgreich einen Benutzer angelegt, der `su(1)` darf und somit für die Wartung geeignet ist.

3.1.2 Unbenötigte Benutzer Entfernen

Unbenötigte Benutzer zu entfernen verbessert auch die Übersicht und minimiert verwundbare Stellen. So können die Benutzer named, popa3d, uucp und www ohne weiteres entfernt werden, da diese Dienste nicht in ein Firewall gehören. Dies kann mit dem `vipw(8)` relativ schmerzlos gemacht werden, einfach die Zeilen der betreffenden Benutzer löschen.
Da OpenBSDs Überwachungsfuntkionen auch Dateien dieser Benutzer überwachen müssen noch weitere Anpassungen gemacht werden. So ist das File `/etc/mtree/special` anzupassen. Einfach alle Zeilen welche 'uucp' eingetragen haben auskommentieren. Deshalb weil dieser Benutzer jetzt nicht mehr vorhanden ist, und es sonst Fehler im täglichen 'Insecure Report' anzeigt. Weiter muss in `/etc/newsyslog.conf` die Zeile '`/var/aculog uucp.dialer`' gelöscht werden.
Um Fehlermeldung in Zusammenhang mit dem fehlenden Benutzer 'named' vorzubeugen, ist es noch ratsam in `/var/named/` die Dateien zu löschen, welche dem Benutzer gehört haben (diese haben jetzt die alte User-ID '70' im Owner-Feld).

3.1.3 Datei Berechtigungen

Die default Berechtigungen von OpenBSD sind sicher genügend, jedoch kann man noch etwas optimieren. So sind viele Dateien für alle Welt lesbar. Dies sind inetd.conf, login.conf, rc, rc.conf, rc.local, rc.securelevel, rc.shutdown, netstart, syslog.conf, hosts.allow und hosts.deny. Die letzteren 2 sind noch nicht erstellt, sollten dann aber auch mit `chmod 600` versehen werden, wie die anderen. Auch mit daily, weekly, monthly, security, changelist and /etc/mtree/special sollte man nur root r/w Rechte geben. Daneben sollten die Dateien .cshrc, .login und .profile des Benutzers root auch mit '600' versehen werden.
Diese Änderungen ziehen noch eine weitere mit sich. So muss `/etc/mtree/special` noch an die neuen Berechtigungen angepasst werden. So muss man bei jedem File, das man oben geändert hat, die Berechtigungen entsprechend aktualisieren.
Weiter sollte das File 'changelist' noch in `/etc/mtree/special` aufgenommen werden, dies erreicht man mit der Zeile:

```
changelist      type=file mode=600 uname=root gname=wheel
```

Das File 'changelist' enthält eine Liste von Dateien welche auf Änderungen überwacht werden sollen.
Vielleicht haben sich nun ein paar gefragt, wieso dieser Aufwand. Da ein Einbrecher seine Aktionen bestmöglich tarnen will kommt er nicht darum herum, Änderungen an überwachten Datein zu meiden. Wenn wir nun diese Dateien offenlegen weiss ein Einbrecher genau was er meiden sollte. Es ist vergleichbar mit dem Unterschied ob wir unsere Juwelen mit rotem oder infrarot Laser schützen.

3.1.4 Zugriffs Berechtigungen

Ein wichtiges Thema ist auch wer Zugriff auf die Maschiene bekommt. So sollte man die Passwort Datei darauf überprüfen, dass nur root und reele Benuzer sich mit einer Shell einloggen können. Bei allen technischen Benutzer sollte `/sbin/nologin` stehen.
Beim Zugriff über das Netzwerk ist sicher noch eine weitere Beschrängkung sinnvoll. So sollte man das Einloggen als root über SSH verbieten. Dies geschieht, in dem man in `/etc/sshd_config/` die Varaible `PermitRootLogin` auf no setzt. Auch sollte im gleichen File das SSH Protokoll zwingend auf 2 eingestellt werden. Dies mit der Zeile `Protocol 2`.

3.2 Unbenötigte Dienste enfernen

Nach dem wir nun mit den Benutzern aufgeräumt haben, kommen die Dienste dran. Je mehr Dienste auf einem Server laufen, um so grösser ist die Möglichkeit, dass einer der Dienste ein Sicherheitsrisiko beinhaltet, oder anders: um so kürzer die Zeit bis bei einem der Dienste ein Sicherheitsloch entdeckt wird. Daraus folgt: Neben den potenziell unsicheren Diensten auch alle abschalten, die nicht benötigt werden.

3.2.1 telnet, ftp, und tcp wrappers

Telnet ist heute schon fast sträflich, und auch ftp birgt viele Risiken, so sind viele FTP Server von Bufferoverflows verwundbar. Wenn diese Dienste umbedingt benötigt werden und nicht durch SSH (auch ftp geht über SSH) ersetzt werden können, sollte man sie nur mit TCP Wrapper einsetzen. Zusätzlich sollte man den FTP Server in einem 'Rootchain' laufen zu lassen.

3.2.2 Portmap, rpc, nfs ...

Diese typische LAN Dienste sind bei OpenBSD per default an, mit der Begründung, dass sie fast immer benötigt werden. Für einen Internetserver oder ein Firewall ist dies jedoch nicht der fall. So kann man im `inetd.conf` die beiden Zeilen fast am Ende, welche mit 'rstatd/1-3' und 'rusersd/1-3' beginnen, auskommentieren. Weiter sind noch Anpassungen in `rc.conf` nötig. So muss die 'portmap' Variable auf nein gesetzt werden.
Weiter kann man im `inetd.conf` auch noch die restlichen nicht benötigten Dienste abschalten so time, daytime, finger, ident, ntalk und comsat. Wenn man nun alle Zeilen des `inetd.conf` auskommentiert hat kann man am Ende auch noch den 'Inetd' im `rc.conf` abschalten.

3.2.3 Sendmail

In OpenBSD wird 'sendmail' mit einer schrägen Konfiguration ausgeliefert. So akzeptiert es Mail nur vom localhost, Aber lokale E-Mails werden nicht ausgeliefert, wenn der Domain nicht aufgelöst werden kann. Dies erfordert Anpassungen in `/usr/share/sendmail/cf/openbsd-localhost.mc` in welcher nach '`FEATURE(‘no_default_msa’)`' die Zeile:

```
FEATURE(‘accept_unresolvable_domains’)dnl
```

Eingefügt werden muss. Danach diese mit `make openbsd-localhost.cf` kompilieren und nach `/etc/mail/localhost.conf` kopieren. Wenn von unserem Firewall auf keinen DNS zugegriffen werden kann muss noch `/etc/service.switch` erstellt werden mit dem Inhalt:

`host files`

Danach kann man mit `kill -HUP ‘sed 1q /var/run/sendmail.pid‘` sendmail neu starten.

3.2.4 Was läuft noch, was steht noch offen?

Um nun zu überprüfen ob und welche Ports noch offen sind gibt es 2 Möglichkeiten. Entweder man lässt mit `netstat -an` alle Verbindungen anzeigen, die offen sind und alle Ports auf welchen gelauscht wird. Die 2. Möglichkeit ist, dass man sich eines Port-Scanners bedient. Dieser sagt einem dann ganz genau was offen oder zu ist. Einer ist zum Beispiel Nmap von `www.insecure.org`, welchen ich oben schon einmal erwähnt habe.
Beide Möglichkeiten zeigen einem dass UDP 514, syslog, offen ist. Dies ist unbedenklich da `syslogd` in jedem fall ein Socket aufmacht, im Normalfall aber alle ankommenden Daten wegschmeisst. Startet man `syslogd(8)` jedoch mit der option ’-u’ nimmt es auch Einträge von anderen Hosts entgegen. So können Geräte wie andere Rechner oder Router in die System logs Schreiben. Lässt man diesen Service exponiert kann man mit sogenanntem Syslog Bombing angegriffen werden, welche einem die Logfiles füllt.

3.3 ntp

Da in einer Sicherheitsrelevanten Umgebung die Server alle die selbe Zeit haben sollen, installieren wir das Programm ’ntp’. Dies holt die Zeit von einem Server und richtet so die Firewall-Uhr. Installieren kann man das Packet mit `pkg_add packagelocation` wo bei ’packagelocation’ ein lokaler Pfad oder ein FTP Server sein kann. Als ich dieses Dokument schrieb war die Version 4.1.71 Aktuell, so hiess das Packet `ntp-4.1.71.tgz`. Danach muss man noch das Konfigurationsfile `ntp.conf` erzeugen. Entsprechende Beispielkofigurationen kann man von (`/usr/local/share/examples/`) kopieren. Im `rc.conf` müssen wir nichts ändern da dieser schon eingeschalten ist.

3.4 Securelevel 2

In OpenBSD gibt es sogenannte ’securelevel(7)’. Hier die wichtigsten:

0 - Während des 1. Teil des Bootens oder im Single User Mode

1 - Default Level nach der dem Hochfahren

2 - Sicherer Level nach dem Hochfahren

In den Singelusermode kommt man mit der Boot-Option `boot> -s` oder mit `kill -s TERM 1`. Securelevel 2 hat neben den Effekten von Secure Level 1 folgenden Zusätzliche restriktionen (nicht vollständig):

- settimeofday(2) kann die Zeit nicht mehr zurück stellen
- die Konfiguration von pfctl(8) (Firewall und NAT) kan nicht mehr verändert werden.

Vor allem die 2. Restriktion ist für unser Firewall nützlich. Sind wir im Securelevel 2 kann ein Angreifer, auch wenn er Root-Rechte erlangt hat, die Firewallregeln nicht ändern. Wir können diese ändern, indem wir in den Singel Usermode gehen. Im Singel User Mode werden alle Netzwerk

Verbindungen getrennt[2], somit brauchen wir Lokalen Zugriffa auf die Konsole um die Regeln zu ändern.
Um dauerhaft in den SecrueLevel 2 zu Booten ändert man die Variable 'securelevel' im File `/etc/rc.securelevel` auf '2'. Mit `sysctl -w kern.securelevel=2` kommt man nun manuell in den SecureLevel 2 (anstelle eines Reboot).
In den Singel User Mode kommt man wenn man `kill -s TERM 1` ausführt oder mit der Boot Option '-s'. Dies zeigt die ersten Sekunden keine Reaktion, dann kommen aber diverse Stautsmeldungen und man wird nach der Shell und nach dem Terminal Typ gefragt. Für alle die den Terminaltyp vergessen haben sei hier gesagt, dass 'vt220' das Standard Terminal ist. Mit 'Exit' oder CTRL-D kommt man wieder zurück in das normale Secure Level.
Da im Securelevel 2 die Uhr nicht mehr zurückgestellt werden kann muss 'ntp' mit der Option '-x' gestartet werden, so wird die Uhr nur noch gebremst. Diese Anpassung muss man im script `/etc/rc.local` vornehmen.

3.5 System Flags

Dies ist alles ein bisschen unnütz wenn man bedenkt, dass der Einbrecher in `rc.secrurelevel` das Securelevel auf 1 zurückstellen kann und dann mit einem Reboot wieder im SecureLevel 1 ist. Dem schaffen wir mit den System-Flags Abhilfe. Von diesen gibt es mehrere, wovon für uns das 'Superuser Immutable Flag' (schg) interessant ist. Wird einer Datei mit `chflags(1)` ein solches Flag zugewiesen, kann man diese nicht mehr verändern. das 'schg' Flag kann man nur noch im Securelevel 0, z.B. dem Singel User Mode, entfernen. Die gesetzten Flags kann man mit `ls -lo` anzeigen lassen.
Um unser System nun sicher im Securelevel 2 zu Parken, können wir mit

```
#chflags schg /etc/rc.securelevel
```

das Flag setzten. Es empfiehlt sich auch noch weitere Schlüsseldateien wie die `pf.conf` so zu schützen. Je mehr Dateien geschützt werden, je mehr empfiehlt es sich ein Script für die das Locken und Unlocken zu schreiben. Auch hier sind auf [2] 2 Scripts vorhanden welche einem das Locken und Unlocken von Files erleichtern.
Ein weiterer Aspekt dieser Flags ist, dass Dateien, welche mit diesem Flag geschützt sind, weder verändert, gelöscht oder überschrieben werden können, was zum Beipiel Schutz vor Trojanischen Pferden bietet.

3.6 Logon Banner

Logon Banner zeigen einem Angreifer meist nützliche Informationen. Dies ist bei OpenBSD in `/etc/motd` der Fall. Da das rc Script die ersten 2 Zeilen jeweils überschreibt, muss diese auch noch angepasst werden. Dies indem man die Zeilen von '`# patch /etc/motd`' bis nach '`rm -f $T [\n] fi`' auskommentiert. Danach hat man freie Hand beim Editieren des Logon Banners. Dieses sollte im Idealfall rechtlich Hinweise auf die Benutzung dieser Maschine haben und den Benutzer darauf Hinweisen dass seine Aktionen geloggt werden. Hier ein Beispiel:

[2] Jedoch pf und das weiterleiten von Packeten läuft weiter

Unauthorized access prohibited; all access and activities not explicitly authorized by [your name/company] are unauthorized.
All activities are monitored and logged. There is no privacy on this system.
Unauthorized access and activities or any criminal activity will be reported to appropriate authorities.

Literatur

[1] http://openbsd.hrxnet.de/faq/index.html

[2] http://geodsoft.com/howto/harden/

[3] http://www.inebriated.demon.nl/pf-howto/

[4] http://geodsoft.com/cgi-bin/password.pl

[5] http://geodsoft.com/cgi-bin/pwcheck.pl

[6] http://www.insecure.org/tools.html