Konfiguration eines Servers mit FreeBSD 5.4

Von Beat Gätzi

beat@chruetertee.ch

Aktuelle Version auf: www.chruetertee.ch

Version: 13.11.05 15:48:29

Inhaltsverzeichnis

1. Einführung

Diese Konfigurationsanleitung bezieht sich auf die Konfiguration eines FreeBSD 5.4 auf einem Server ohne graphische Oberfläche. Dies ist keine Kochbuchanleitung, sondern mehr eine Ideensammlung. Also mach nur das was Du verstehst und brauchst und lass die Finger von allem anderen! Der Autor übernimmt keine Verantwortung für das was Du machst!!!

2. RAID

Von https://www.bsdforen.de/showthread.php?t=1281

2.1. SoftRaid erstellen mit atacontrol

Wir haben 2 Platten, jeweils als Masterdevice an ATA Channel 0 ad0 und an 1 ad2. Zur Installation haben wir ein CD-ROM als Slave an den 2ten IDE Kanal angeschlossen.

Entweder System auf Platte ad0 installieren. Es darf ruhig minimal installiert werden, da wir sowieso später die Installation erneut durchführen. Neustart. Von der Installation auf ad0 booten.

Oder System mit FreeSBIE booten.

```
# atacontrol create RAID1 ad0 ad2
```

Dabei wird die Raidsignatur auf den beiden Platten geschrieben. Allerdings werden die vorhandenen Daten nicht gespiegelt. FreeBSD Installation's CD wieder in das CDROM einlegen, neustarten und neu installieren.

Jetzt zeigt der Installer als mögliche Ziele der Installation :

ad0, ad2, UND unser Array ar0

Wir wollen natürlich auf ar0 installieren.

So werden jetzt alle Daten identisch auf ad0 und ad2 geschrieben

2.2. SoftRaid funktionen testen

```
# atacontrol detach 1
```

Wir schalten den ATA Channel 1 um einen Ausfall einer Platte zu simulieren. System meldet "UAHHH Array degraded. Wir schreiben jetzt mal auf dem "degraded Array", z.B.

```
# dmesg >beispiel.txt
# atacontrol attach 1
```

Wir schalten Channel 1 wieder zu

```
# atacontrol addspare ar0 ad2
```

und geben dem Array ar0 wieder eine "spare" Disk, damit wir mit dieser wieder das Array herstellen können.

```
# atacontrol rebuild ar0
# atacontrol status ar0
```

gibt jetzt zurück wie weit die Wiederherstellung fortgeschritten ist.

3. Partitionen

Mountpoint	*Minimal*	*Empfohlen*
/	100 MB	300 MB
/tmp	50 MB	300 MB
swap	0 MB	1 – 2 mal RAM Grösse
/var	100 MB	300 MB
/usr	2 GB	Rest

Dies ist der "Standardfall", bedenke das je nach Anwendung des Servers diese Werte nicht stimmen müssen und einige Partitionen massiv vergrössert werden müssen, z.B. /var bei einem Datenbankserver.

Alternativ können noch Partitionen für /home, /root gemacht werden. Auch können die Jails in eine eigene Partition z.B. /jails oder /usr/jail erstellt werden. Eine Jail ohne installierte Programme braucht ca. 150 MB.

4. Propolice for FreeBSD 5.4

Von http://www.paranoid.nl/~eilander/freebsd/propolice/

Diese Anleitung wird FreeBSD mit einem Stack-Smashing Protector ausrüsten. Sei Dir sicher ob Du das auch wirklich willst!!! Wenn Du nicht weisst, um was es hier geht, lass die Finger davon!

Lade Dir folgende Patches hinunter:

http://www.paranoid.nl/~eilander/freebsd/sysctlarnd/sysctl_arnd.diff

http://www.paranoid.nl/~eilander/freebsd/propolice/propolice-fbsd54-2.diff

http://www.paranoid.nl/~eilander/freebsd/propolice/propolice-test.c

MD5 (sysctl_arnd.diff) = e530ba63456fb87b056c784b672cc64f

MD5 (propolice-fbsd54-2.diff)= ff500c6c4d1ae9d693f4c962068e3d5d

MD5 (propolice-test.c) = 35418722dcb28f856e2bf12979d1d326

Die FreeBSD 5.4 Quelldateien müssen auf dem System verfügbar sein.

```
# cd /usr/src
# patch -p0 < sysctl_arnd.diff
# cp /usr/src/sys/sys/sysctl.h /usr/include/sys/sysctl.h
```

Kernel neu bilden und neuen Kernel booten. Als nächstes werden libc und gcc mit SSP neu gebaut:

```
# cd /usr/src
# patch -p0 < propolice-fbsd54-2.diff
# cd /usr/src/lib/libc
# make obj && make depend && make all install
# cd /usr/src/gnu/usr.bin/cc
```

```
# make obj && make depend && make all install
```

Testen ob dies geklappt hat

```
# gcc -v
Using built-in specs.
Configured with: FreeBSD/i386 system compiler
Thread model: posix
gcc version 3.4.2 [FreeBSD] 20040728 (propolice)
```

Der Stackprotector kann mit propolice-test.c getestet werden:

```
# gcc -o ptest propolice-test.c
# ./ptest
Abort (core dumped)
# tail -2 /var/log/messages
Apr 23 15:55:34 devel ptest: stack overflow in function main
Apr 23 15:55:34 devel kernel: pid 17722 (ptest), uid 0: exited on signal 6
(core dumped)
```

SSP sendet dem Programm ein SIGABORT (signal 6).

Ein ausführbares C Programm ist geschützt, wenn man __stack_smash_handler mit strings sieht. Diese Binärdateien werden auf Systemen ohne SSP in der LIBC nicht laufen!

```
$ strings a.out | grep smash
__stack_smash_handler
```

Kompiliere alles neu, was Du mir SSP geschützt haben willst:

- Kernel
- Welt
- Applikationen

Programme sind nur geschützt, wenn sie auch mit einem propolice-gcc übersetzt wurden. Also verwende keine vorkompilierten Pakete, sondern baue alles selber aus den Ports.

5. Passwortverschlüsselungsalgorithmus ändern

> Von http://wiki.bsdforen.de/index.php?title=FreeBSD_-_Server_absichern
> Ursprünglich geschrieben von Highfish unter http://www.bsdforen.de/showthread.php?t=2174
> Lizenzbestimmung des BSDForen.de Wikis:
> http://wiki.bsdforen.de/index.php/BSDForen.de_Wiki:Lizenzbestimmungen

Du kannst den Passwortverschlüsselungsalgorithmus wechseln. Der bereits genügend sichere MD5-Algorithmus kann durch den vermutlich noch besseren Blowfish-Verschlüsselungsalgorithmus ersetzt werden. Folgende Änderungen sind notwendig:

/etc/login.conf

```
 :passwd_format=md5:\    =>    :passwd_format=blf:\ (Erstes Zeichen ist ein
Leerschlag!)
```

Damit die Änderungen in der /etc/login.conf auch wirklich übernommen werden, muss folgendes Kommando ausgeführt werden:

```
# cap_mkdb /etc/login.conf
```

/etc/auth.conf

```
# crypt_default = md5 des   =>   crypt_default=blf
```

Jetzt müssen die Passwörter der Benutzer einzeln auf den neuen Verschlüsselungsalgorithmus umgestellt werden:

```
# passwd <Benutzername>
```

Alle mit dem Blowfish-Algorithmus verschlüsselten Passwörter beginnen in der Passwörter-Datei /etc/master.passwd mit "$2".

6. GENERIC Kernel sichern

Vor dem backen eines eigenen Kernels wir der GENERIC Kernel gesichert

```
# mkdir /boot/kernel.GENERIC
# cp -R /boot/kernel/* /boot/kernel.GENERIC
# cp /boot/device.hints /boot/device.hints.default
```

Im Falle des Falles kann der GENERIC Kernel im Bootmenu über folgenden Befehl gebooted werden

```
# boot /kernel.GENERIC
```

7. Kernelkonfiguration

7.1. Grundsätzliches

```
# more /usr/src/sys/i386/conf/NOTES
```

Zeigt Informationen zu einigen Kerneloptionen an.

```
# cd /usr/src/sys/i386/conf/ && make LINT
```

Erstellt die LINT Datei, mit allen verfügbaren Kerneloptionen und devices.

7.2. Kernel anpassen

Führe ein

```
# dmesg | grep CPU
```

aus und entferne die Unterstützung für andere CPU Klassen. Im Falle eines

```
# dmesg | grep CPU
CPU: Intel(R) Pentium(R) M processor 1.80GHz (599.49-MHz 686-class CPU)
```

kommentiere die anderen beiden aus:

```
#cpu            I486_CPU
#cpu            I586_CPU
```

Falls Du kein IPv6 haben möchtest, kommentiere diese Option aus

```
#options        INET6                   # IPv6 communications protocols
device          gif                     # IPv6 and IPv4 tunneling
device          faith                   # for IPv6 and IPv4 translation
```

Falls dieser Server nicht ein NFS-Server wird, kommentiere die Option aus

```
#options        NFSSERVER               # Network Filesystem Server
```

Wird vom Prozessor SSE/MMX2 und/oder FPU unterstützt, so kann dieses in der Konfiguration aktiviert werden.

```
options         CPU_ENABLE_SSE
options         CPU_FASTER_5X86_FPU
```

Ob diese Unterstützt werden, zeigt ein

```
# dmesg | grep Features
```

Für IPFW Unterstützung, die standardmässig alles akzeptiert und Weiterleitung unterstützt.

```
options         IPFIREWALL
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPFIREWALL_VERBOSE
options         IPFIREWALL_VERBOSE_LIMIT=100
options         IPFIREWALL_FORWARD
options         IPFIREWALL_FORWARD_EXTENDED
```

Server also IP Gateway nutzen:

```
options         IPDIVERT
```

TTL wird nicht hinuntergezählt, um Server vor traceroute zu verstecken;

```
options         IPSTEALTH
```

Für device polling Unterstützung folgende Optionen hinzufügen. Dies ist nur auf schnellen Einprozessormaschinen sinnvoll die viel Netzwerk verkehr zu bewältigen haben. Mehr Informationen unter polling(4) und http://info.iet.unipi.it/~luigi/polling/ .

```
options         DEVICE_POLLING
options         HZ=1000         # oder 2000
```

Verhindert durch das Drücken von Alt-Esc oder Ctl-Print Screen den Eintritt in den Kerneldebugger

```
options         SC_DISABLE_KDBKEY
```

Verhindert das neu starten des Systems durch Ctl-Alt-Del

```
options         SC_DISABLE_REBOOT
```

Falls keine Maus am Server angeschlossen ist

```
options         SC_NO_FONT_LOADING
options         SC_NO_CUTPASTE
options         SC_NO_SYSMOUSE
```

Verwirft TCP Pakete die das SYN und das FIN Flag gesetzt haben. Dies erschwert OS fingerprinting, kann aber zu Problemen mit einigen Applikationen führen. Diese Option ist auf Webserver nicht zu

empfehlen!

```
options         TCP_DROP_SYNFIN
```

Die muss in der /etc/rc.conf mit tcp_drop_synfin="YES" eingeschaltet werden.

Bei einem System auf dem mehrere Personen arbeiten, können Dienste wie ps, sockstat und w durch das MAC Framework ein bisschen weniger auskunftsfreudig gemacht werden. Dieses Framework ist immer noch experimentell, also entscheide selber ob Du es brauchst.

```
options         MAC
options         MAC_SEEOTHERUIDS
```

Nicht gebrauchte Geräteunterstützungen können auch noch aus dem Kernel entfernt werden, um ihn ein bisschen schlanker zu machen.

7.3. Kernel backen

Neu modisch:

```
# cd /usr/src && make buildkernel KERNCONF=<KERNELNAME> && make
installkernel KERNCONF=<KERNELNAME>
```

Altmodisch:

```
# cd /usr/src/sys/i386/conf && config <KERNELNAME> && cd
../compile/<KERNELNAME> && make depend && make && make install
```

8. Konfigurationsdateien

8.1. /etc/make.conf

Entferne nicht gebrauchte Teile des Basissystems in der Datei, durch entfernen der Raute beim jeweiligen Eintrag. Mein Vorschlag sieht so aus:

```
NO_CVS= true     # do not build CVS
NO_BLUETOOTH=  true     # do not build Bluetooth related stuff
NO_I4B= true     # do not build isdn4bsd package
NO_PF=           true     # do not build PF firewall package
NO_LPR= true     # do not build lpr and related programs
NO_USB=         true     # do not build usbd(8) and related programs
NO_VINUM=        true     # do not build Vinum utilities
NOATM=           true     # do not build ATM related programs and libraries
NOGAMES=         true     # do not build games (games/ subdir)
NOINET6=         true     # do not build IPv6 related programs and libraries
NOINFO= true     # do not make or install info files
NOPROFILE=       true     # Avoid compiling profiled libraries
```

Falls nur ein Betriebssystem auf dem Server zum Einsatz kommt, kann die Wartezeit des Bootloaders verkürzt werden. Die Zeit ist in Millisekunden.

```
BOOTWAIT=3000
```

Falls Du IPv6, CUPS und X11 Unterstützung beim Bauen vom Paketen nicht haben willst, so wende die folgenden 3 Zeilen an:

```
WITHOUT_IPV6=yes
WITHOUT_CUPS=yes
WITHOUT_X11=yes
```

Es sind zwar nur 4 Ports die ein WITHOUT_DEBUG unterstützen, aber schaden tut es sicher nicht, also fügen wir noch folgendes an:

```
WITHOUT_DEBUG=true
```

Und diesen noch um die Last auf dem Apache FTP's ein bisschen zu verteilen.

```
MASTER_SITE_APACHE_HTTPD?= http://mirror.switch.ch/mirror/apache/dist/httpd/
http://mirror.switch.ch/mirror/apache/dist/httpd/
ftp://mirror.switch.ch/mirror/apache/dist/httpd/
http://www.apache.org/dist/httpd/ http://www.eu.apache.org/dist/httpd/
```

Lass die Finger von Compilerflags wie -O2 -fast-math -funroll-loops! Dies bringt nur Probleme und definitiv keine Performance. Auf einem FreeBSD 4.9 lief scp nicht mehr, nachdem ich das Userland mit -fast-math -funroll-loops gebaut habe. Auf einem FreeBSD 5.4 beendeten Programme plötzlich mit einem Segmentation fault (core dumped), nachdem ich sie mit CPUTYPE?=pentium4m aus den Ports gebaut habe. Falls es wirklich Optimierung sein muss, verwende CFLAGS= -O -pipe und COPTFLAGS= -O -pipe, da dies durch den FreeBSD Kernel und Userland unterstützt wird.

8.2. /boot/loader.conf

Verkürzt der Countdown auf 3 Sekunden und macht den Daemon schön farbig:

```
autoboot_delay="3"
loader_color="YES"
```

8.3. /etc/rc.conf

Mein Vorschlag, picke Dir raus, was Dir gefällt. rc.conf(5) gibt Auskunft über die Variablen.

```
keymap="swissgerman.iso"
sshd_enable="YES"
sendmail_enable="NO"
nfs_server_enable="NO"
nfs_client_enable="NO"
portmap_enable="NO"
syslogd_enable="YES"
syslogd_flags="-ss"
clear_tmp_enable="YES"
firewall_enable="YES"
firewall_script="/etc/ipfwrules"
firewall_logging="YES"
```

```
kern_securelevel_enable="YES"
kern_securelevel="3"
devd_enable="YES"
ipv6_enable="NO"
inetd_enable="NO"
tcp_drop_synfin="YES" # drop TCP packets with SYN+FIN
#tcp_extensions=”NO” # turn off RFC1323 extensions
#tcp_keepalive=”NO” # disable stale TCP connection timeout
icmp_drop_redirect="YES" # ignore ICMP REDIRECT packets
icmp_log_redirect="YES" # log ICMP REDIRECT packets
icmp_bmcastecho=”NO” # respond to broadcast ping packets
```

8.4. /etc/sysctl.conf

Folgende Zeilen werden hinzugefügt, um das Ganze ein bisschen sicherer zu machen:

```
security.bsd.see_other_uids=0
net.inet.ip.random_id=1
net.inet.tcp.blackhole=2
net.inet.udp.blackhole=1
security.jail.set_hostname_allowed=0
kern.randompid=2
```

Wird seeotheruids aus dem MAC Framework verwendet, so kann diese Beschränkung mir folgender Variable für Benutzer der Gruppe wheel aufgehoben werden.

```
security.mac.seeotheruids.specificgid_enabled=1
```

8.5. /etc/syslog.conf

Bei folgende Zeilen wird die Raute entfernt, da wir alle Konsolen ausgaben geloggt haben wollen:

```
console.info                                    /var/log/console.log
```

Danach ein “touch /var/log/console.log“ als root ausführen, um die Logdatei zu erstellen

8.6. /etc/ssh/sshd_config

Folgendes sollte in der sshd_config stehen

```
Port 22
Protocol 2
LogLevel INFO
PermitRootLogin no
AllowTcpForwarding no
X11Forwarding no
```

```
Subsystem       sftp    /usr/libexec/sftp-server
AllowGroups <Benutzergruppe die Zugriff per SSH haben soll>
```

8.7. /etc/ttys

Datei muss wie folgt geändert werden, dass sich root nicht mehr direkt anmelden kann und auch beim starten in den Singleusermode nach dem root-Passwort gefragt wird. Alle Einträge von secure auf insecure ändern.

```
console none                            unknown off insecure
#
ttyv0   "/usr/libexec/getty Pc"         cons25  on  insecure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv2   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv3   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv4   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv5   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv6   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv7   "/usr/libexec/getty Pc"         cons25  on  insecure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off  insecure
# Serial terminals
# The 'dialup' keyword identifies dialin lines to login, fingerd etc.
ttyd0   "/usr/libexec/getty std.9600"   dialup  off insecure
ttyd1   "/usr/libexec/getty std.9600"   dialup  off insecure
ttyd2   "/usr/libexec/getty std.9600"   dialup  off insecure
ttyd3   "/usr/libexec/getty std.9600"   dialup  off insecure
# Dumb console
dcons   "/usr/libexec/getty std.9600"   vt100   off insecure
```

8.8. /etc/fstab

/tmp und /var können mit den Optionen "nosuid,noexec,nodev" eingebunden werden. /usr kann mit der Option "nodev" eingebunden werden. Eventuell kann auch / und /usr Schreibgeschützt einhängen werden (Option: "ro"). Falls /tmp mit “noexec“ eingegängt wird, so muss diese Partition vor einem make installworld ohne diese Option neu gemounted werden, da dieses sonst fehlschlägt.

```
# Device     Mountpoint  FStype  Options                    Dump Pass #
/dev/ad0s1a   /           ufs      ro                         1    1
/dev/ad0s1b   none        swap     sw                         0    0
/dev/ad0s1e   /var        ufs      rw,nosuid,noexec,nodev     1    2
/dev/ad0s1f   /tmp        ufs      rw,nosuid,noexec,nodev     0    2
/dev/ad0s1g   /usr        ufs      ro,nodev                   1    2
```

8.9. /etc/login.conf

Um die User in den Ressourcen zu beschränken, Datei wie folgt anpassen. Die Werte können nach belieben angepasst werden und sind dem Einsatz des Servers anzupassen. Das Einschränken der Benutzer muss nicht immer eine gute Sache sein, verwende es nur, wenn es auch nötig ist.

```
default:\
        :passwd_format=blf:\
        :copyright=/etc/COPYRIGHT:\
        :welcome=/etc/motd:\
        :setenv=MAIL=/var/mail/$,BLOCKSIZE=K,FTP_PASSIVE_MODE=YES:\
        :path=/sbin /bin /usr/sbin /usr/bin /usr/local/sbin /usr/local/bin
~/bin:\
        :nologin=/var/run/nologin:\
        :cputime=1h30m:\
        :datasize=8M:\
        :stacksize=2M:\
        :memorylocked=4M:\
        :memoryuse=8M:\
        :filesize=8M:\
        :coredumpsize=8M:\
        :openfiles=24:\
        :maxproc=32:\
        :sbsize=unlimited:\
        :vmemoryuse=100M:\
        :priority=0:\
        :ignoretime@:\
        :idletime=30:\
        :umask=022:
root:\
        :ignorenologin:\
        :passwd_format=blf:\
        :copyright=/etc/COPYRIGHT:\
        :welcome=/etc/motd:\
        :setenv=MAIL=/var/mail/$,BLOCKSIZE=K,FTP_PASSIVE_MODE=YES:\
        :path=/sbin /bin /usr/sbin /usr/bin /usr/games /usr/local/sbin
/usr/local/bin /usr/X11R6/bin ~/bin:\
        :nologin=/var/run/nologin:\
        :cputime=unlimited:\
        :datasize=unlimited:\
        :stacksize=unlimited:\
```

```
        :memorylocked=unlimited:\
        :memoryuse=unlimited:\
        :filesize=unlimited:\
        :coredumpsize=unlimited:\
        :openfiles=unlimited:\
        :maxproc=unlimited:\
        :sbsize=unlimited:\
        :vmemoryuse=unlimited:\
        :priority=0:\
        :ignoretime@:\
        :idletime=30:\
        :umask=022:
```

Nach dem ändern folgenden Befehl als root ausführen

```
# cap_mkdb /etc/login.conf
```

8.10. ~/.exrc

Um den vi(1) ein bisschen umgänglicher zu machen, verwende ich folgende Optionen:

```
set verbose
set showmode
set ruler
set windowname
set autoindent
set ignorecase
set showmatch
set number
```

9. IPFW Firewall

Von http://www.bsdforen.de/showthread.php?p=30110

Das Firewallscript wird unter /etc/ipfwrules erstellt. Unter den Variablem open_tcpports und open_udpports können die offenen Ports per Komma getrennt eingetragen werden.

```
#!/bin/sh
#
# Erstmal alles saubermachen bevor wir anfangen

# Also die Regeln auf "Null" stellen
/sbin/ipfw -q -f flush
```

```
# IPFW-Kommando "Quiet"
fwcmd="/sbin/ipfw -q add"

# Das setzen unserer eigenen Variabeln
open_tcpports="22,80" # ${open_tcpports} Offene Ports
open_udpports=”7777” # ${open_udpports} Offene Ports

# Erlaubt Loopbackverbindungen
${fwcmd} 00100 allow ip from any to any via lo0

# Stateful Packet Inspection
${fwcmd} 00200 check-state

# Ack Packete ohne vorheriges Req werden geblockt
${fwcmd} 00250 deny log tcp from any to any established in

#Erlaubt alle Verbindungen welche von hier initiiert wurden
${fwcmd} 00300 allow tcp from any to any out setup keep-state
${fwcmd} 00310 allow udp from any to any out keep-state

# Erlaubt bereits bestehenden hergestellten Verbindungen offen zu bleiben
${fwcmd} 00320 allow tcp from any to any  established
${fwcmd} 00330 allow udp from any to any established

# Erlaubte Dienste die ausm Internet erreicht werden dürfen
${fwcmd} 00400 allow tcp from any to any ${open_tcpports} setup keep-state
${fwcmd} 00410 allow udp from any to any ${open_udpports} keep-state

# Sendet RESET an alle Ident Pakete, welche auf Port 113 tcp eingehen
${fwcmd} 00500 reset log tcp from any to me 113 in

# Loggt ICMP Anfragen (echo und dest. unreachable)
${fwcmd} 00700 allow log icmp from any to any in icmptype 3
${fwcmd} 00710 allow log icmp from any to any in icmptype 8

# ICMP erlauben
${fwcmd} 00750 allow icmp from any to any
```

```
# Alles andere verbieten
${fwcmd} deny log ip from any to any
```

10. Dateisystem

10.1. Temporäres Verzeichnis

Um /var/tmp auf der gleichen Partition wie /tmp zu verwenden, führe folgende Befehle aus:

```
# mv /var/tmp/* /tmp
# rm -rf /var/tmp
# ln -s /tmp /var/tmp
```

10.2. Zugriffsrechte

Um die Zugriffsrechte der Benutzer einzuschränken, kann folgendes Skript verwendet werden. Auf einem System mit nur vertrauenswürdigen Benutzern ist dies nicht notwendig.

```
#!/bin/sh
chmod 600 /var/log/*
chmod 700 /root
chmod 700 /home/*
echo "root" > /var/cron/allow
echo "root" > /var/at/at.allow
chmod o= /etc/crontab
chmod o= /usr/bin/crontab
chmod o= /usr/bin/at
chmod o= /usr/bin/atq
chmod o= /usr/bin/atrm
chmod o= /usr/bin/batch
chmod o= /etc/fstab
chmod o= /etc/ftpusers
chmod o= /etc/group
chmod o= /etc/hosts
chmod o= /etc/hosts.allow
chmod o= /etc/hosts.equiv
chmod o= /etc/hosts.lpd
chmod o= /etc/inetd.conf
chmod o= /etc/login.access
chmod o= /etc/login.conf
chmod o= /etc/newsyslog.conf
chmod o= /etc/rc.conf
```

```
chmod o= /etc/ssh/sshd_config
chmod o= /etc/sysctl.conf
chmod o= /etc/syslog.conf
chmod o= /etc/ttys
chmod o= /usr/bin/users
chmod o= /usr/bin/w
chmod o= /usr/bin/who
chmod o= /usr/bin/lastcomm
chmod o= /usr/sbin/jls
chmod o= /usr/bin/last
chmod o= /usr/sbin/lastlogin
chmod ugo= /usr/bin/rlogin
chmod ugo= /usr/bin/rsh
```

10.3. Dateiflags

Beachte Dateiflags bringen evtl. mehr Sicherheit, erschweren den Unterhalt eines Servers aber massiv. Also verwende die Dateiflags nur mit Bedacht!

```
# chflags schg /bin/*
# chflags schg /sbin/*
# chflags schg /usr/sbin/*
# chflags schg /boot/kernel
# touch /boot.config
# chflags schg /boot.config
```

11. Software

Folgende Software lohnt sich schon standardmässig zu installieren:

/usr/ports/security/chkrootkit	A tool to locally check for signs of a rootkit
/usr/ports/sysutils/cpdup	A comprehensive filesystem mirroring program
/usr/ports/net/cvsup-without-gui	General network file distribution system optimized for CVS (non-GUI version)
/usr/ports/sysutils/pkg_cutleaves	Interactive script for deinstalling 'leaf' packages
/usr/ports/security/portaudit	Checks installed ports against a list of security vulnerabilities
/usr/ports/sysutils/portupgrade	FreeBSD ports/packages administration and management tool suite

12. Jails

12.1. Erstellen

```
# mkdir -p /usr/jail/myjail
```

```
# cd /usr/src && make buildworld && make installworld
DESTDIR=/usr/jail/myjail
# cd /usr/src/etc && make distribution DESTDIR=/usr/jail/myjail
# mount_devfs devfs /usr/jail/myjail/dev
# cd /usr/jail/myjail
# ln -sf /dev/null kernel
# mkdir /usr/jail/myjail/stand
# cp /stand/sysinstall /usr/jail/myjail/stand
# echo 'sshd_enable="YES"' >> /usr/jail/myjail/etc/rc.conf
# echo 'sshd_flags="-p <alternativer Port>"' >> /usr/jail/myjail/etc/rc.conf
# echo 'sendmail_enable="NO"' >> /usr/jail/myjail/etc/rc.conf
# echo 'rpcbind_enable="NO"' >> /usr/jail/myjail/etc/rc.conf
# echo 'network_interface=""' >> /usr/jail/myjail/etc/rc.conf
# echo ''hostname="<hostname>" >> /usr/jail/myjail/etc/rc.conf
# echo 'syslogd_enable="YES"' >> /usr/jail/myjail/etc/rc.conf
# echo 'syslogd_flags="-ss"' >> /usr/jail/myjail/etc/rc.conf
# echo 'kern_securelevel_enable="YES"' >> /usr/jail/myjail/etc/rc.conf
# echo 'kern_securelevel="3"' >> /usr/jail/myjail/etc/rc.conf
# echo "nameserver <IP des Nameservers>" > /usr/jail/myjail/etc/resolv.conf
```

usr/jail/public/etc/ssh/sshd_config 'PermitRootLogin' auf 'no' setzen und # davor entfernen

```
# ifconfig <interface> alias <IP>/32
# jail /usr/jail/myjail  <jailname> <IP> /bin/sh
# touch /etc/fstab
# cd /etc && newaliases
```

Wurde System ohne IPv6 Unterstützung gebaut, so wird newaliases eine Fehlermeldung ausgeben. Dieses Problem wird wie folgt behoben:

```
# cd /etc/mail
# cp freebsd.mc <Vollständiger Servername mit Domain>.mc
Löschen von folgender Zeile:
DAEMON_OPTIONS(`Name=IPv6, Family=inet6, Modifiers=O')
# make
# make install
# make restart-mta
```

root Passwort, Zeitzone , Tastaturbelegung, Benutzer und Gruppen mit Hilfe von /stand/sysinstall erstellen

In /etc/ssh/sshd_config ListenAddress <IP> setzen

Zeile mit “adjkerntz -a“ in der /etc/crontab mit Raute auskommentieren, da dies sonst Fehlermeldungen zur Folge hat.

```
# cp /etc/defaults/periodic.conf /etc/
```

Folgende Zeilen in der /etc/periodic.conf ändern, da sonst Fehlermeldungen in den periodic Ausgaben sind.

daily_status_network_enable="NO"

daily_status_security_ipfwdenied_enable="NO"

daily_status_security_ipfdenied_enable="NO"

daily_status_security_pfdenied_enable="NO"

daily_status_security_ipfwlimit_enable="NO"

daily_status_security_ip6fwdenied_enable="NO"

daily_status_security_ipf6denied_enable="NO"

daily_status_security_ip6fwlimit_enable="NO"

```
# exit
```

12.2. in '/etc/rc.conf' einfügen

```
Ifconfig_<interface>_alias0="inet <eine_ip_adresse> netmask 0xffffffff"
jail_enable="YES"
jail_list="public"
jail_set_hostname_allow="NO"
jail_sysvipc_allow="NO"
jail_public_rootdir="/usr/jail/public"
jail_public_hostname="<hostname>"
jail_public_ip="<die_ip_adresse_von_alias0>"
jail_public_exec="/bin/sh /etc/rc"
jail_public_devfs_enable="YES"
jail_public_mount_enable="NO"
```

12.3. für weitere jails

```
fconfig_<interface>_alias0="inet <eine_ip_adresse> netmask 0xffffffff"
ifconfig_<interface>_alias1="inet <eine_2te_ip_adresse> netmask 0xffffffff"
jail_enable="YES"
jail_list="public1 public2"
jail_set_hostname_allow="NO"
jail_sysvipc_allow="NO"
jail_public1_rootdir="/usr/jail/public1"
```

```
jail_public1_hostname="<hostname>"
jail_public1_ip="<die_ip_adresse_von_alias0>"
jail_public1_exec="/bin/sh /etc/rc"
jail_public1_devfs_enable="YES"
jail_public1_mount_enable="NO"
jail_myjail2_rootdir="/usr/jail/myjail2"
jail_myjail2_hostname="<hostname>"
jail_myjail2_ip="<die_ip_adresse_von_alias1>"
jail_myjail2_exec="/bin/sh /etc/rc"
jail_myjail2_devfs_enable="YES"
jail_myjail2_mount_enable="NO"
```

12.4. Jail starten

Falls Jails in der /etc/rc.conf konfiguriert sind

```
# /etc/rc.d/jail start
```

sonst

```
# jail /usr/jail/myjail <Jailname> <IP> /bin/sh /etc/rc
```

12.5. Anzeigen der aktiven Jails

```
# jls
```

12.6. Jails kopieren

```
# cpdup /usr/jail/myjail1 /usr/jail/myjail2
```

12.7. Verzeichnisse des Hostsystemes in Jail mounten

```
# mount_nullfs /usr/ports/ /usr/jails/myjail/usr/ports/
```

ACHTUNG: Dies kann erstens eine Sicherheitslücke sein, da man Files aus dem Hostsystem in der Jail beschreibbar macht und zweitens bitte “man 8 mount_nullfs“ lesen!!!

Aus “man 8 mount_nullfs“ Abschnitt Bug: THIS FILE SYSTEM TYPE IS NOT YET FULLY SUPPORTED (READ: IT DOESN'T WORK) AND USING IT MAY, IN FACT, DESTROY DATA ON YOUR SYSTEM. USE AT YOUR OWN RISK. BEWARE OF DOG. SLIPPERY WHEN WET

Bei mir tuts, aber ich hab Dich gewarnt!

12.8. Jail beenden

Falls Jails in der /etc/rc.conf konfiguriert sind

```
# /etc/rc.d/jail stop
```

Sonst mit “jls“ die JID in Erfahrung bringen und dann mit “killall -j <JID>“ die Jail beenden.

13. Schöne TCSH

Von http://www.bsdbox.de/?page_id=6

Folgende Datei bearbeiten: /etc/csh.cshrc

Dort einfügen:

```
alias ls ls -GF
if ($?promt) then
set promt = "[%B%n@%m%b] %B%~%b/> "
endif
```

Danach in /root wechseln und die Datei .cshrc editieren. Dort den originalen "set prompt"-eintrag durch diesen hier ersetzen:

```
set prompt = "[%{^[[41m%}%B%n@%m%b] %B%~%b/>%{^[[39m%} "
```

14. Aktualisierung

14.1. Kernel und Userland

Zuerst die Quellcodes mittels CVSup auf den aktuellen Stand bringen:

```
#  cvsup -g -L 2 /usr/sup/stable-supfile
```

Mein /usr/sup/stable-supfile:

```
*default host=cvsup.ch.FreeBSD.org
*default base=/var/db
*default prefix=/usr
*default release=cvs tag=RELENG_5_4
*default delete use-rel-suffix
*default compress
src-all
```

Danach unbedingt /usr/src/UPDATING lesen und ggf. den Anweisungen folgen. Ein komplettes Update wird wie folgt gemacht.

```
# cd /usr/src && make clean && rm -rf /usr/obj/*
# make buildkernel KERNCONF=<KERNELNAME>
# make buildworld
# make installkernel KERNCONF=<KERNELNAME>
```

Vor dem Neustart überprüfen, dass in der /etc/fstab die Partition /tmp nicht mit "noexec" eingebunden ist, da sonst ein make installworld fehlschlägt.

```
# reboot
```

In den Single-Usermode wechseln (boot -s)

```
# /sbin/fsck -p
```

```
# /sbin/mount -u /
# /sbin/mount -a -t ufs
# /sbin/swapon -a
# /etc/rc.d/preseedrandom
# /usr/sbin/mergemaster -p
# cd /usr/src
# make installworld
# /usr/sbin/mergemaster
# /sbin/reboot
# cd /usr/src && make clean && rm -rf /usr/obj
```

14.2. Ports

Da ich kein Freund von Aktualisierungen bin, werden nur Ports aktualisiert, die Sicherheitslücken aufweisen.

Folgender Befehl zeigt dir installierte Ports mit Sicherheitsproblemen an:

```
# portaudit -Fda
```

Portaudit befindet sich im security/portaudit Port.

Zuerst die Quellcodes mittels CVSup auf den aktuellen Stand bringen:

```
#  cvsup -g -L 2 /usr/sup/ports-supfile
```

Mein /usr/sup/ports-supfile:

```
*default host=cvsup.ch.FreeBSD.org
*default base=/var/db
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix
*default compress
ports-all
```

```
# cd /usr/ports && make fetchindex
```

Vor dem Update unbedingt /usr/ports/UPDATING lesen und den Anweisungen folgen!

Ist das Problem ohne Upgrade auf eine neue höhere Version des Ports zu beheben(z.B. Open-motif-2.2.3_1 auf 2.2.3_2)), so kann nur der Port mit portupgrade neu gebaut und installiert werden.

```
# portupgrade <Portname>
```

Hat die neue Version eine höhere Minor- oder Majorreleasenummer, so müssen alle vom Port abhängigen Programme neu gebaut werden.

```
# portupgrade -rf <Portname>
```

Danach ggf. die Konfigurationsdateien anpassen und ggf. das Programm oder der Daemon neu starten. Nach einem Update kann der Portbaum noch aufgeräumt werden.

```
# portsclean -CDP
```

15. Programm löschen

Um ein installiertes Programm zu löschen, verwende pkg_deinstall und kein make deinstall, da ein make deinstall die Abhängigkeiten nicht überprüft und das Programm deinstalliert, obwohl andere Ports dieses noch bräuchten.

Um ein Programm mit nur von ihm genutzten Abhängigkeiten zu löschen, verwende

```
# pkg_deinstall -R -d -v <Portname>
```