A basic guide to securing FreeBSD 4.x-STABLE

Written by Marc Silver <marcs@draenor.org>
http://draenor.org/securebsd

$Id: secure,v 1.11 2002/05/29 08:50:32 marcs Exp $;

Table of Contents:

Overview

========

The word security means different things to different people. While this document covers various aspects and suggests things that can be done to secure default installations of FreeBSD, it is is by no means an authoritive guide to securing FreeBSD. It merely discusses a model that I use on my own machines and one that I have had great success with. I'd also like to point out that I am by no means a security 'expert'... I am merely a _very_ paranoid sysadmin who takes great pride in securing my servers.

For a broader look at security on FreeBSD and as a primer to this document, I would suggest that everyone read the man page for security(7) on their FreeBSD system.

This is a work in progress. As such, this document will change, grow and develop over time. If you have something to add, wish to suggest a change, make a comment or say anything for that matter please email me (marcs@draenor.org). The authoritive home of this document is http://draenor.org/securebsd

It should also be noted that this document isn't by any means going to stop remote or local DoS attacks. It can merely help you to better secure default FreeBSD instalations. This document is also NOT an 'advanced' paper on Securing FreeBSD. It covers basic ideas that you can use for securing your machine.

With that out of the way, let's begin.

The Foundation for a secure system
==================================

A system should be set up to be secure from the very beginning. There are a number of things that can be done during the FreeBSD installation that can save you serious headaches later. In my opinion, file system setup can make a big difference in cases where you can (and must) assume that the attacker already has a local login on the machine.

o File System Layout

   The file system layout below may be used as a guideline for any system. Obviously, disk layout can/will differ from machine to machine based on the function of the machine but this should serve as a basic guide.

   You may adapt this to suit your own needs.

```
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad0s1a    128990    31390    87282    26%    /
/dev/ad0s1f     49583    27879    17738    61%    /tmp
/dev/ad0s1d  12348393  2563101  8797421    23%    /usr
/dev/ad0s1h   4065262    97983  3642059     3%    /home
/dev/ad0s1g   2032623     6026  1863988     0%    /var
procfs              4        4        0   100%    /proc
```

   Now, let's look at the output from the mount(8) command:

```
/dev/ad0s1a on / (ufs, local)
/dev/ad0s1f on /tmp (ufs, local, nodev, nosuid, soft-updates)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
/dev/ad0s1h on /home (ufs, local, nosuid, with quotas, soft-updates)
/dev/ad0s1g on /var (ufs, local, soft-updates)
procfs on /proc (procfs, local)
```

Now, let's discuss why I've set things up this way.

The root partition (/) is a reasonable 128MB, (as recommended in the tuning(7) man page) and is home to the kernel as well as KLD's and various other fairly important directories which are linked directly off it (/sbin is just one that comes to mind). With this in mind, it's possible at a later stage to mount the root partition as read-only by editing the flags for this partition in the fstab(5) file.

Temporary files are stored in /tmp, and since this directory is usually world writeable, it's important to not allow certain files to be used from this directory. Using the fstab(5) file (also see mount(8)) you should add the NOSUID and NODEV flags for /tmp which disables suid programs and stops character or block special devices on the filesystem. You may also want to add the NOEXEC flag for /tmp, but this is severely restrictive and may begin to make things difficult for your users. NOEXEC will also cause problems when you 'make installworld', since a fairly normal /tmp is required for this. Enabling NOEXEC may also limit your ability to find an intruder. It's important to note that you should symlink /usr/tmp and /var/tmp to this /tmp partition, else you're still giving users a tmp directory with no restrictions.

User specific directories are kept in /home and on this partition it's a good idea to add the NOSUID flag, as well as adding QUOTA support to limit the amount of disk space that your users may use.

Both /usr and /var are standard partitions with soft-updates enabled.

You may choose to also disable procfs. See 'Disabling procfs' for more information.

This model can obviously be changed to suit your needs, and you can be even more anal if you wish. This however, is intended to strike a happy medium between security and usability.

Post Installation

Once your FreeBSD system has been installed there are a number of things that can be done to help harden the machine.

o System Secure Levels

Security levels are at the core of FreeBSD security. They are extremely powerful and are essential in securing FreeBSD.

For most machines there is absolutely no reason to run in securelevel -1, unless you wish to run X-Windows on the machine. If you're not running X-Windows, then I would suggest switching to securelevel 1 using the sysctl(8) variable kern.securelevel. Changing this to 1 will mean that you may no longer replace the kernel without being in single user mode (system immutable and system append-only flags are also enforced), KLD's may not be loaded/unloaded and /dev/mem and /dev/kmem may not be opened for writing. To change this without rebooting you should issue the following command:

```
sysctl kern.securelevel=1
```

To make this change more permanent, add the following to /etc/rc.conf:

```
kern_securelevel_enable="YES"
```

kern_securelevel="1"

On more critical machines, you may wish to increase the securelevel to 2 or 3. I will not discuss these higher secure levels in this document, but hopefully you're interested enough to want to find out more. For more information on the various secure levels and what they do, please read the man page for init(8).

It has been noted that system secure levels > 0 can cause problems with applications that monitor things such as CPU temperature. This is because of the way secure levels > 1 protect certain devices and certain memory allocations.

o Removal of the toor user

By default, FreeBSD ships with an additional user that has a UID of 0. This user is known as toor (root backwards), and is intended as a backup user, so that if you mistakenly broke (for eg) root's shell, you could log in using this user and fix things. The account is disabled (passwordless) by default, and hence of no use UNLESS you change it's password. You may either choose to set a password for it, or remove it. I prefer to remove it, but the choice to do so is entirely up to you.

It should be noted that the rmuser(8) command will not allow the deletion of an account with a UID of 0, so you will need to use vipw(8) to remove this account.

o Shut down services that you dont need/want

It's important to not have any non-essential services running on the machine, or any services that you dont recognise. The best thing to do is kill all the services running on your machine and then explicitly enable those that you want running. This way you know for sure what's running on your machine. You can tell what TCP ports are open on your machine by using the netstat(1) command. eg:

```
secure-me (1) :  netstat -na | grep LIST
tcp4       0      0  *.80                   *.* LISTEN
tcp4       0      0  *.25                   *.* LISTEN
tcp4       0      0  *.22                   *.* LISTEN
```

This shows that ports TCP ports 22 (ssh), 25 (smtp), and 80 (http) are listening on this machine and are bound to all IP's. If you have a process listening and you're unsure of what process is keeping that port open you may use sockstat(1) to list open sockets and provide you with the relevant information.

Use rc.conf(5) to easily configure which services start up by default, as well as local package init scripts which can be found in /usr/local/etc/rc.d

Similarly you may wish to see if you have anything listening via UDP. You can also get this information via netstat(1):

```
secure-me (2) :  netstat -nap udp
udp4       0      0  *.514                  *.*
```

Here, you see that syslogd is listening on port 514 (UDP).

I will now discuss some common services and what you can do to better secure them.

- syslogd ::

syslogd will by default bind itself to UDP 514, but you can prevent this from happening by adding a second '-s' flag to syslogd's command line on startup. This prevents syslogd from using network sockets and can be done by simply adding the following line to /etc/rc.conf :

syslogd_flags="-ss"

See the syslogd(8) man page for more information.

- portmap ::
portmap is used for remote procedure calls and its most common application in FreeBSD is for use with NFS. To disable portmap add the following line in your /etc/rc.conf:

portmap_enable="NO"

- telnetd ::
This service should be avoided at all costs. While telnet is useful, there is just no excuse to use it anymore. All data transferred across a telnet session is transmitted in clear text (including usernames and passwords). This should be disabled, either by killing inetd(8) completely, or by removing the telnetd line from /etc/inetd.conf. If you MUST run this service, then look at using something like login.access(5) or ipfw(8) to limit where connections to this service may come from.

FreeBSD comes standard with sshd(8), a drop-in replacement for telnetd with far superior security built in.

- sshd ::
FreeBSD (since 4.1.1) now comes with OpenSSH as part of the base system, and sshd(8) is a perfect drop in replacement for telnetd, while remaining more secure by using encryption to protect your session. The protocol also allows for stronger encryption with the use of RSA/DSA keys.

It should be noted that the most current versions of OpenSSH now use SSH protocol version 2, but for those systems that use a slightly older version, it is advised to only allow version 2 of the protocol. This can be done by making sure the following line exists in /etc/ssh/sshd_config:

Protocol 2

This will tell the sshd that it should only allow incoming SSH2 connections - and it will not fallback to version 1. Please note that you may need to restart the sshd in order for this change to take effect.

Similarly, you should also make sure that all outbound SSH connections are using SSH2 by default, and then failing back to SSH1. This can be done by editing /etc/ssh/ssh_config

It's also preferable to use DSA keys wherever possible to enhance security even more. This document will not cover the use of DSA keys since this information is available (and probably will be kept up to date) on http://www.openssh.com/

For more information on OpenSSH, RSA/DSA keys and how to use them all effectively, please visit http://www.openssh.com/

- inetd ::
inetd is designed to listen for connections on certain sockets. It

is used for popular applications like telnetd(8), qpopper and ftpd(8). At the very least, you should comment out (by adding a # to the beginning of the lines you do not want) the services that you do not need/intend to run. If you do not require any of the daemons that inetd runs, you should disable it completely by adding the following line to /etc/rc.conf :

inetd_enable="NO"

If you do use inetd, take a look at /etc/hosts.allow to limit where you wish to allow inbound connections from.

ftpd ::
FreeBSD comes with a fairly robust ftpd by default. FTP is by default an insecure protocol since usernames and passwords are transitted in clear text. A good replacement for FTP is sftp(1) which offers all the benefits of FTP with the encryption of SSH. Alternatively, you may also use scp(1).

If however, you decide to run an ftpd there are a few things that you can do to help harden the server. By default, ftpd is run with one "-l" flag, but this may be increased to provide more information about what commands users are issuing the server. By adding a "-r" flag, you can also put the ftpd into read-only mode, which will disable all commands that modify the filesystem. I also add the "-A" flag to only allow anonymous logins. That way, there is no chance of user accounts being compromised by the tranmission of clear text passwords.

The ftpd line should now look something like this in /etc/inetd.conf:

ftp stream tcp nowait root /usr/libexec/ftpd ftpd -l -l -r -A

It should be noted that LOG_FTP messages are not logged by syslogd(8) by default and may have to be enabled. Please see the man page for ftpd(8) for more information.

For obvious reasons it is not recommended to allow user logins. If you want user logins, I would suggest using ncftpd (http://www.ncftpd.com) which allows you to use a seperate database of usernames/passwords that are independent of the usernames on the system, and which can (and should) have different passwords.

o Log in vain

Even though you've now disabled many services, you should log connection attempts to ports without listeners/daemons. To do this simply add the following line to /etc/rc.conf:

log_in_vain="YES"

To change this without rebooting your server issue the following commands:

sysctl net.inet.tcp.log_in_vain=1
sysctl net.inet.udp.log_in_vain=1

Now, failed connection attempts to ports without listeners will be recorded to /var/log/messages.

o Blackhole

FreeBSD also allows you the option to blackhole any TCP/UDP traffic

that is bound for ports without daemons/listeners. Instead of logging the connection like 'log in vain' does, it ignores the packet, thereby creating a blackhole into which packets dissapear. The man page for this feature also states that this could potentially slow down DoS attacks aimed at your system. The Blackhole MIB can be easily enabled by issuing the following commands:

```
sysctl net.inet.tcp.blackhole=1
sysctl net.inet.udp.blackhole=1
```

It should be noted that enabling blackhole for UDP will prevent people from being able to traceroute(8) to your system. The TCP value may also be increased to 2. For more information, please see the blackhole(4) manpage.

To make these changes permanent, add the following lines to /etc/sysctl.conf:

```
net.inet.tcp.blackhole=1
net.inet.udp.blackhole=1
```

It should also be noted that this is NOT a replacement for ipfw/ipf.

NOTE: According to James Lawrence <jl@imaginengine.com>, this breaks Konqueror.

o Crontabs

Firstly, there are certain files which you may generally not want users looking at. The crontab of the root user is a perfect example. You can safely chmod /etc/crontab to 0640 so that only root and users in the wheel group can see it. Your users do not need to know what jobs are started by cron.

At the same time, you may not want to allow users to use crontab(1) at all. You can easily stop them by creating /var/cron/deny and adding a list of users to that file. Those users will then be told:

```
crontab: you (marcs) are not allowed to use this program
```

Similarly, you may also create a /var/cron/allow and only add users that should be allowed to use crontab to that file. For more information, please see the crontab(1) man page.

o Secure the console

Many people are concerned that a malicious user with physical access could simply reboot into single user mode and change the root password. While it's quite clear that if an attacker has physical access to your machine, NOTHING you do can keep it safe, you can prevent people from simply changing the root password in single user mode by performing one simple step. This can be done by editing /etc/ttys and changing the the word 'secure' on the 'console' line to 'insecure'. This will require you to enter the root password when dropping into single user mode. Your line will then look like this:

```
console none                            unknown off insecure
```

You should also be aware that if you do this, and you somehow lose the root password, you will have to use a fixit floppy to reset the password, because dropping into single user mode will NOT allow you to change the password.

o Process accounting

It's nice to know exactly what's happening on your machine and to this end I would suggest enabling process accounting on any machine that you run. This enables you to see what commands users are executing, and it can also be useful when debugging certain problems. It does add some slight overhead, but generally you shouldn't notice degraded performance. To enable, merely execute the following commands:

```
secure-me (1) :  touch /var/account/acct
secure-me (2) :  accton /var/account/acct
```

To make this change more permanent, add the following line to /etc/rc.conf:

```
accounting_enable="YES"
```

Once accounting is enabled, you can then use the lastcomm(1) and sa(8) commands to get meaningful statistics from the process accounting database.

o ipfw

While ipfw is well beyond the scope of this document, you may wish to secure the machine further as well as gain information on attack patterns on your machine using ipfw. This can sometimes provide information that someone is more interested in your machine than they should be. See the ipfw(8) page for more information.

o Mail aliases

It's important to make sure that you're being notified of anything that's happening on your system. FreeBSD has many scripts that are triggered on a daily/weekly/monthly basis (see the man page for periodic(8) and take a look at /etc/periodic for more information) that contain information about your system such as SUID program changes, kernel messages and other useful information. For this reason it is important to get these mails. The output of these scripts are sent to the root account, but you may choose to send the output to multiple addresses. To do this, edit /etc/mail/aliases and add a line similar to this:

```
root:   localuser, remoteuser@yourdomain.com
```

This means that not only is the local administrator getting a copy, but you're also mailing the output to a (hopefully anyway) seperate mail server where it is theoretically out of harms way.

Kernel changes

o Disable bpf if you dont need it

One of the first things I do when I install a FreeBSD machine is recompile the kernel. One of the options that I like to disable in the kernel is the bpf device, since this would stop an attacker from putting the network card of the machine into promiscious mode. This is useful should the machine itself is compromised. Simply comment out the following line in your kernel file:

```
#pseudo-device   bpf             #Berkeley packet filter
```

Users who are reliant on DHCP should not disable bpf. Disabling BPF can also affect applications such as snort, which rely on being able

to drop a network card into promiscuous mode.

You may also want to add in the options for ipfw, ipfilter, and add quota support at the same time.

o Disable Ctrl-Alt-Del

You can stop users with physical access from using the Ctrl-Alt-Del combination to reboot your machine. Simple add the following line to your kernel to disable this:

```
options         SC_DISABLE_REBOOT       # disable reboot key sequence
```

You should be aware that all this will really prevent is users from rebooting your machine. If you've also marked your console as insecure it'll stop people from rebooting to change the root password. That said, if someone has physical access and they want to do something malicious, you're already in more serious trouble...

o Quota Support

In order to enable Quota support for your filesystems, you will need to enable this option in your kernel. This can be done with the following option:

```
options         QUOTA                   #enable disk quotas
```

o ipfw/ipf support

Firewall support can also be added into the kernel. What you choose to add (or not) is up to you. FreeBSD comes standard with ipfw and ipf in the base system. See LINT for more information.

Managing user accounts

o User quotas

By enforcing user quotas on certain filesystems you can limit the damage that an attacker who wants to consume disk space can do. Enforce quotas wherever possible to prevent users from filling your disks. This also gives you the added advantage of being able to manage your disk usage more effectively.

Quotas can only be used if you have compiled support for them in the kernel. Once you've done this, you will need to add the following lines in /etc/rc.conf:

```
enable_quotas="YES"
check_quotas="YES"
```

You may then use edquota(8), quotacheck(8), quotaon(8), quotaoff(8) and repquota(8) to manage quota filesystems.

o Home directory permissions

You should be aware of what it is your users can see. Just as you dont want user's to be able to see what is in root's crontab you may also not want them to view what is in root's directory. A quick 'chmod 0750 /root' will make sure that they can't see the contents unless they're in the wheel group.

To that end, you may also want to restrict user home directories by setting their permissions to 0700 by default. This way users will

have to explicitly change their directory permissions in order for other users to view their directory contents.

o Hiding processes

You can also limit what processes a user can see when using the ps(1) command. By default, FreeBSD will allow users to see all processes on the system, including those that do not belong to them. You may wish to only allow the user to see processes owned by them. To do this, you may use the kern.ps_showallprocs sysctl variable. You can change this while the system is running by issuing the following command:

sysctl kern.ps_showallprocs=0

To make this change permanent, insert the following line into /etc/sysctl.conf:

kern.ps_showallprocs=0

The root user is not affected by kern.ps_showallprocs and can always see all processes.

While this method is effective for limiting what output ps(1) gives, it will not stop an attacker from traversing /proc to find out what processes are running. See 'Disabling procfs' for more information.

o Disabling procfs

procfs can be used to gather information on running processes. It is required for the complete operation of programs such as ps(1), w(1) and truss(1). Due to the amount of information that procfs may yield many administrators feel that it is advantageous to disable this filesystem.

This step is ENTIRELY voluntary. You do not need to disable this if you do not want to.

To disable procfs, add the NOAUTO option to /etc/fstab for this filesystem. You may then mount it manually if needed.

o login.conf(5)

FreeBSD allows you to add users to 'login classes', where you can control (for example) how much CPU/memory each user can use. This can be very effective in limiting local DoS attacks, whether intentional or by accident. Since the use of login.conf would most likely require a document of it's own, you are encouraged to read the man page - login.conf(5) - for more information.

Stay up to date

o Keep your packages current

When you're running daemons that are worldly visible and accessible it's important to make sure (and it's common sense) that your packages are always up to date. If you see a new version of a package you have installed, then update it via the ports tree to make sure that you've always got the latest version. It only takes a few minutes in most cases, but it's worth the effort if you're saving the machine from being compromised. It'll help to watch lists like bugtraq for security advisories.

For more information on keeping your packages current, please see the FreeBSD HandBook:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/

o Keep your OS current

Similarly, it's important to keep FreeBSD itself up to date. Keep your source tree up to date, and 'make world' if/when new security patches are made available. It'll help to watch lists like bugtraq for security advisories.

For more information on keeping your OS current, please see the FreeBSD HandBook:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/

You should also subscribe to the FreeBSD Security Advisories mailing list. See:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html

for more information on subscribing.

Be Vigilant

You should always be on the look out for strange behaviour as it is very often the first sign that something is wrong. You should especially be on the lookout for:

o A machine that has recently rebooted.

If your machine has rebooted (and you didn't do it) check to see if anything serious has been changed. You should be especially watchful of system securelevel's, since these will need to be changed for the attacker to change things like the kernel, KLM's etc and afford an opportunity to bypass system immutable and system append-only flags.

o Changes in SUID files.

The daily reports that get run contain information about any changes in SUID files. Pay important attention to these.

o Changes in critical system files.

Watch out for changes in critical system files, like /kernel. When you install FreeBSD, you should take MD5 values of these files and store them somewhere safe (not on the machine). Compare these values from time to time. If they dont match (for reasons you cant explain) then investigate. You can use something like /usr/ports/security/tripwire for this, though there are also other alternatives.

Topics of Interest

Other topics that may be of interest to you, but which are outside the scope of this document:

o Jail

The jail environment is similar to chroot(8), but with several enhancements. Explanations of what it is and how to use it may be

found in the jail(8) man page and at:

http://docs.freebsd.org/44doc/papers/jail/jail.html

Other documents about FreeBSD Security

o The following sites also contain information on securing FreeBSD:

- http://www.freebsd.org/security/
- http://www.freebsd.org/~jkb/howto.html
- http://www.subterrain.net/presentations

o Tools that help secure FreeBSD

- http://www.openssh.com/ - Now comes in the FreeBSD base distribution.

o Other useful applications

- http://www.ncftpd.com/ - A secure FTP daemon.

Thanks

o I would also like to thank the following people for making suggestions to improve this document or for pointing out problems, suggesting additions etc.

Ivan Bruce Morrisby (Jim), without whom I would NEVER have bothered to sit down and write this. Thank you for your inspiration and great friendship.

Hiten Pandya <hitmaster2k@yahoo.com> (who is converting this document to DocBook for addition as an article on the FreeBSD website)
Gary W. Swearingen <swear@blarg.net>
Tom Rhodes <darklogik@zoominternet.net>
Nick Cleaton <nick@cleaton.net>
Sean Lewis <sml@subterrain.net>
Jean-Michel Amblat <jmamb@videotron.ca>
Dominic Marks <dominic_marks@btinternet.com>
Seth Bromberger <email removed by request>
Chris Phillips <chris@bchosting.com>
Sebastian Benner <sebastian.benner@fernuni-hagen.de>
Trevor Johnson <trevor@jpj.net>
James Lawrence <jl@imaginengine.com>