/Guides/FreeBSD/Security/

Hardening FreeBSD

Updated: 06/27/2005

General Information

After a fresh install, it is important to harden the security on a server before it hits your network for use. Not only making configuration changes aid in the security of your box, but there are some practical rules to abide by. These are some hardening tips to make your FreeBSD box more secure and will apply to both the 5.x and 4.x branches, but I will assume you are running 5.x. If a 4.x change is different, I will note it.

> **Note:** Please do not apply these changes carelessly on a production server. Make sure you test, test, test on a separate box to note the effects of the changes.

Requirements

1. Clean installation of FreeBSD
2. Local root access on the box or be able to su to root.
3. A SSH client that supports ANSI colors such as puTTy or SecureCRT (if you aren't on the box).
4. Your favorite text editor (I prefer nano).

Hardening

Filesystem Structure

On a default install, you will find two places for temporary files -- /tmp and /var/tmp. Different packages and daemons use the different directories and there really isn't any need to use two different partitions for temporary files. Here we will replace /var/tmp with a link to /tmp.

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

Disable Local root Access

The first rule to securing your box is to never treat the root account as a regular user. Never conduct mundane business while you are logged in as root. As the superuser, there are no restrictions as to what root can do, so always log in with a normal user and su to root only when needed.

With that in mind, it may be tempting to log on locally as root so let's prevent root from directly logging on to your system console. Open **/etc/ttys** with your editor and change every occurence of "secure" to "insecure" as this will prevent root from logging in locally. *Note: changing the console entry will result in being prompted for the root password when booting into single-user mode; thus, making the recovery of the root password more difficult.

```
# nano -w /etc/ttys

***output omitted***
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
***output omitted***
```

Save and exit and your changes will take effect immediately.

SSH Logins

By default, FreeBSD prevents root from logging in via ssh, but it gives anyone else with a valid user account access. If you are not running a shell server, it is a good idea to restrict ssh access to only members of the wheel group -- or you can create a separate group if you want some people to log in but not be able to su to root, say a group named "sshlogins." Let's add the following to the end of the sshd configuration file:

```
# cat << EOF >> /etc/ssh/sshd_config
# PermitRootLogin=no
# AllowGroups wheel sshlogins
# Protocol 2
# X11Forwarding=no
# VersionAddendum
# EOF
```

We also want to ensure only SSHv2 connections are made as SSHv1 does not offer all the security of v2. Servers don't need to be running X11 so we might as well turn off forwarding for X11 to make sure there aren't any attempts. The final entry we made was to disable the OS display.

An optional security measure to take is add a banner for users to see before they log on. If you like the idea, follow these steps:

```
# echo "Banner /etc/welcomemsg" >> /etc/ssh/sshd_config
# cat << EOF > /etc/welcomemsg
#                           !!WARNING!!!
#              READ THIS BEFORE ATTEMPTING TO LOGON
#
# This System is for the use of authorized users only. Individuals
# using this computer without authority, or in excess of their authority,
# are subject to having all of their activities on this system monitored
# and recorded by system personnel.  In the course of monitoring individuals
# improperly using this system, or in the course of system maintenance,
# the activities of authorized users may also be monitored.  Anyone using
# this system expressly consents to such monitoring and is advised that if
```

```
# such monitoring reveals possible criminal activity, system personnel may
# provide the evidence of such monitoring to law enforcement officials.
#
# EOF
```

Password Rules

By default, FreeBSD uses md5 for password hashing and encryption. It's not bad, but blowfish is much better suited for passwords and we need to update some files to reflect blowfish. ***Note:** Passwords will not be converted to blowfish until they have been changed.

```
# echo "crypt_default=blf" >> /etc/auth.conf
```

You need to manually edit **/etc/login.conf** and change the password format in the default class to blf. We should also modify the default password policy to put a minimum password length requirement and mix upper and lower case. Let's also cause passwords to expire after 90 days and to automatically log users out if they are idle for 30 minutes. It's also a good idea to set the default umask to prevent global access. The umask is the inverse to the chmod. So in this case when new files and directories are created, they will get the permissions of 0750.

```
# nano -w /etc/login.conf

 ***output omitted**'
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/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@:\
        :minpasswordlen=8:\
      :mixpasswordcase=true:\
      :passwordtime=90d:\
      :idletime=30:\
      :umask=027:
 ***output omitted**'
```

Update the login database with:

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

After you change the user's password, you will notice the hashing for the password in /etc/master.passwd begins with $2a. This means blowfish is being used.

Restrict User Access

Scheduling jobs is a powerful feature in *nix, but at the same time, it can pose a security concern for your system if you allow users to schedule jobs -- especially if they are set up incorrectly. The potential harm

could be looping a process endlessly, running malicious code (though this wouldn't pose too big of a problem if it's not run as root), or incorrect schedules. It is recommended to restrict cron and at to only root.

```
# 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
```

The next thing we need to restrict is read and execution of certain files. Regular users should not have access to the following configuration files:

```
# 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
```

Attackers tend to clear out all log files when they are finished with your box. If they don't have access to the logs and cannot edit them or delete them, then you can go in and see what was done. First, let's disable user access to the log file directory and then we will set the permissions so the log files cannot be deleted.

Note: Applying these changes to the log files will also mean logs can no longer be rotated.

```
# chmod o= /var/log
# chflags sappnd /var/log
# chflags sappnd /var/log/*
```

You may want to restrict users from trying to execute certain programs like the following:

```
# 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
```

There are a few services that should be disabled permanently:

```
# chmod ugo= /usr/bin/rlogin
# chmod ugo= /usr/bin/rsh
```

And of course, any third-party utilities you install and don't want general users to access should be chmoded as well.

```
# chmod o= /usr/local/bin/nmap
# chmod o= /usr/local/bin/nessus
```

System Configuration for Daemon Startup

Now it is time to enable or disable certain services by editing **/etc/rc.conf**. Here we will disable sendmail as it is an insecure MTA. If you want to run a mail server, I recommend using qmail.

```
# echo 'sendmail_enable="NONE"' >> /etc/rc.conf
```

The default kernel level is -1, meaning not much gets protected. You probably only need the secure level at 2, but 3 is the most secure. If you want more information on the secure levels, read the man pages for init(8). **Note:** The securelevel can only increase once the kernel is loaded.

```
# echo 'kern_securelevel_enable="YES"' >> /etc/rc.conf
# echo 'kern_securelevel="3"' >> /etc/rc.conf
```

If you aren't running NFS, disable portmap:

```
# echo 'portmap_enable="NO"' >> /etc/rc.conf
```

inetd, or the network daemon dispatcher, is insecure so we want to make sure it is disabled.

```
# echo 'inetd_enable="NO"' >> /etc/rc.conf
```

It's a good idea to clear your /tmp directory at startup to make sure there isn't anything malicious hanging around in your temp files.

```
# echo 'clear_tmp_enable="YES"' >> /etc/rc.conf
```

If you are not logging to a remote machine, it is a good idea to make sure syslogd does not bind to a network socket.

```
# echo 'syslogd_flags="-ss"' >> /etc/rc.conf
```

ICMP Redirect messages can be used by attackers to lead you to their router or some other router, which would be bad. Let's ignore those packets and log them.

```
# echo 'icmp_drop_redirect="YES"' >> /etc/rc.conf
# echo 'icmp_log_redirect="YES"' >> /etc/rc.conf
```

The following option is a good choice as it will log all attempts to closed ports. This is good to know if people are trying to access your box through a specific port.

```
# echo 'log_in_vain="YES"' >> /etc/rc.conf
```

Set Kernel States

There are some kernel states we need to change and we'll add them to **/etc/sysctl.conf** to make them permanent. The first one is to prevent users from seeing information about processes that are being run under another UID.

```
# echo "security.bsd.see_other_uids=0" >> /etc/sysctl.conf
```

> **Note:** 4.x users use the following instead:

```
# echo "kern.ps_showallprocs=0" >> /etc/sysctl.conf
```

The second change to make is to enable the concept of blackholing. This is so RST packets don't get sent back in response to closed ports. This helps to block port scans.

```
# echo "net.inet.tcp.blackhole=2" >> /etc/sysctl.conf
# echo "net.inet.udp.blackhole=1" >> /etc/sysctl.conf
```

We want to generate a random ID for the IP packets as opposed to incrementing them by one. This will prevent remote observers from determining the rate packets are being generated by watching the counter.

Note: This setting is only for 5.3 and beyond. If you are running any older version of FreeBSD, you will need to compile your kernel with this option.

```
# echo "net.inet.ip.random_id=1" >> /etc/sysctl.conf
```

Kernel Entries

There are a couple of security settings we can fix at the kernel level. One security hole we need to plug is disabling ctrl+alt+del so somebody can't walk up to your box and reboot your server with the three-finger solute. Add the following lines to the options:

Note: The RANDOM_IP_ID option is only for versions of FreeBSD that are older than 5.3.

```
# nano -w /usr/srs/sys/i386/conf/MYKERNEL

 ***output omitted**'
options         SC_DISABLE_REBOOT       # Disable Ctrl+Alt+Del
options         RANDOM_IP_ID            # Enables random IP ID generation
 ***output omitted**'
```

If you haven't already compiled a custom kernel for your hardware, make the necessary kernel config changes at this time. If you have never done that before, use Derrick's kernel config guide as a guideline.

Once you finish customizing your kernel, install it and then reboot. Once it comes back up, log in and update your ports tree so you can upgrade your ports.

Optional Settings For a Stealthier System

The following options may be used, but are only recommended for system that are gateways, log servers, or dedicated firewalls. You may apply these to normal servers if you would like, but they may decrease performance -- especially on web servers.

We can configure FreeBSD to drop SYN/FIN packets:

```
# echo 'tcp_drop_synfin="YES"' >> /etc/rc.conf
```

Add the following to your kernel configuration to enable the ability to drop SYN/FIN packets and to enable stealth forwarding. Stealth forwarding passes packets without touching the TTL, so this is useful for hiding firewalls from traceroutes.

```
# nano -w /usr/src/sys/i386/conf/MYKERNEL

 ***output omitted**'
options         TCP_DROP_SYNFIN         # Enables the ability to drop SYN/FIN
packets
options         IPSTEALTH               # Enable stealth forwarding
 ***output omitted**'
```

Now your FreeBSD server has been hardened and ready for your production use. You can also use the lockdown utility (/usr/ports/security/lockdown) and it will automate a lot of this, but not everything.

Author: Jon LaBass
jon at bsdguides dot org

12 Comments

Posted by **infovein420** on August 27, 2005 at 8:40:38 pm EEST

This sshd_config line should have the comma removed. Only a space is required for more than 1 parameter.

AllowGroups wheel, sshlogins
to
AllowGroups wheel sshlogins

Leaving the comma will cause authentication problems for a client.

Scott

Posted by **Jon** on August 27, 2005 at 8:40:38 pm EEST

Thanks Scott. But, leaving the comma doesn't cause authentication problems as I just tested it on FreeBSD 5.3. However, I read the manpages and it does say to separate using spaces. I've updated the guide to reflect that.

Posted by **rickster** on August 27, 2005 at 8:40:38 pm EEST

how to undo this change
chmod o= /var/log
chflags sappnd /var/log
chflags sappnd /var/log/*

Posted by **Jon** on August 27, 2005 at 8:40:38 pm EEST

You can undo the chflags by lowering your kernel securelevel and rebooting.

To do this, first edit /etc/rc.conf and change kern_securelevel equal to -1. Then

reboot
chflags nosappend /var/log
chflags nosappend /var/log/*

The default permissions on /var/log is 755 so the following will restore it.

chmod o=rx /var/log

Just reboot again after changing the kernel securelevel to whatever you want.

Posted by **z0rmus** on August 27, 2005 at 8:40:38 pm EEST

About the section "Password Rules":

"The minpasswordlen and minpasswordcase facilities for enforcing restrictions on password quality, which used to be supported by login.conf, have been superseded by the pam_passwdqc(8) PAM module."

"PAM configuration for the 'passwd' service passwd(1) does not use the auth, account or session services."

So you must to change in /etc/pam.d/passwd the statement password to:

```
# password
password        requisite       pam_passwdqc.so         enforce=users
```

Sandro Herman

Posted by **X-Istence** on August 27, 2005 at 8:40:38 pm EEST

To do this, first edit /etc/rc.conf and change kern_securelevel equal to -1

That would defeat the purpose of kern_securelevel being set in the first place, as any attacked could remove that, reboot the server, edit the log files, set it again, and reboot again. That is also one of the reasons why if you do set securelevel in either rc.conf or sysctl.conf you also chflags them to not be able to touch them in secure level.

Posted by **shadowbq** on August 10, 2006 at 6:28:39 pm EEST

When running X11, by default it wants to rotates the X11 logs
you have to manually unset the chflags on

/var/log/Xorg.0.log
/var/log/Xorg.0.log.old

If you dont the Xorg/X11 server will not start.

Posted by **staffan** on February 15, 2007 at 4:27:50 pm EET

I have followed the instructions in this guide and now I can't start X11 (startx or GDM gnome). I removed the restrictions on the /var/log, but still no luck...

Does anyone has any tip on how to make it work again? or why it may not work.

//Staffan Öhrberg

Posted by **staffan** on February 15, 2007 at 5:24:59 pm EET

I can run X11 when kern_securelevel=-1, but this is not secure, right?

//Staffan

Posted by **Jon** on May 24, 2007 at 8:00:41 pm EEST

Setting the kern_securelevel is really more effective for production servers that are Internet-facing. Therefore, there really isn't a reason to set the securelevel when running X11.

Posted by **FissionChips** on June 03, 2008 at 2:43:53 pm EEST

Trying to get x11 running again, I can't remove restrictions OR edit rc.conf

(root):
echo 'kern_securelevel_enable="YES"' >> /etc/rc.conf
cannot create /etc/rc.conf: Read-only file system

Any hints?

Posted by **FissionChips** on June 03, 2008 at 2:58:50 pm EEST

I'm not usually one to post his problems on discussion forums, but I'd been toying around with this one a while and ran out of ideas. Untill I tried one more thing, and it worked.

Excuse my newbieness.. Thanks for all the great guides!

Login to comment.