Global Information Assurance Certification Paper

SANS GIAC Certification

GCUX Practical Assignment
Version 1.9

Securing FreeBSD step by step

Dave Patterson
05/30/03

Contents

1 - Introduction

FreeBSD is one of the earlier UNIX distributions; however it enjoys a very broad based support community. It runs a variety of platforms, x86 compatible, UltraSPARC, DEC Alpha, and IA-64 architectures. FreeBSD supports a large number of applications easily available from its port collection".

> FreeBSD makes an ideal Internet or Intranet server. It provides robust network services under the heaviest loads and uses memory efficiently to maintain good response times for thousands of simultaneous user processes.[1]

I will be setting up a log host running syslog-ng based on the FreeBSD operating system. This system will act as the primary log host for a small to medium sized internal corporate network. I will outline security enhancements to the basic system and the additional applications required to enhance performance. The various settings will be proposed with the "security over functionality" principal; but I will also attempt to incorporate changes with the smallest impact on functionality.

This document will define log host security configuration standards for the FreeBSD operating platform, in addition to utilizing several other pieces of security software.

Current stable versions of software referenced in this document:

- FreeBSD 4.8
- OpenSSH 3.5p1 {system default}
- Syslog-ng 1.6.0.rc3
- AIDE 0.9

All of the information contained in this paper relates to the FreeBSD operating system, specifically 4.8 {version 5.0 is somewhat different} it will not be discussing system V type OS's.

2 - Description of the system

Our base system starts out as follows:
An IBM clone running an Intel Celeron 1.7 GHz processor, with 256 MB of DDR system memory. The system will have a 80GB IDE hard disk and an NE2000 compatible 10/100 Ethernet card. The following system is suggested because it is an adequately powered, affordable system.

3- Risk analysis of the system

Physical access to the box will be extremely limited. The server will be kept in a locked cabinet.

User access to this machine will be very limited – 2 accounts will be created for the administrators that may need to admin this box. ssh root logins will be disabled.

The server only needs to run two internet services: the syslog {running tcp on port 514 and ssh on port 22. Port 514 will receive logs from other hosts. Port 22 is the ssh port and will be used for remote administration, but only from allowed networks. All other ports can be closed by disabling services, shutting down daemons, or by the reconfiguration of certain applications.

So the key security objectives/concerns are:

- patching and hardening the operating system to its fullest;
- syslog-ng exploits that can result in either denial of service or bufferoverflow conditions – so updating syslog-NG regularly.
- similar exploits that effect OpenSSH, or the applications used to build OpenSSH, which means fully patching and hardening OpenSSH, OpenSSL, and zlib.

During the installation and configuration of all hosts, the text editor “vi” is used extensively. It is typically the default text editor that is native on most UNIX platforms and certainly with FreeBSD. If you are unfamiliar with “vi”, I strongly recommend the following web sites for detailed instruction on how to use this editor:

http://www.cs.wustl.edu/~jxh/vi.html
https://engineering.purdue.edu/ECN/Resources/KnowledgeBase/Categories/editors/vi

The following legend will be used throughout this paper

All commands that you will enter will be in this color {light blue}

Console output will be in this color {green}

Important points will be noted in this color {orange}

Section II - Step by Step Guide for Creating the Operating Environment

1 – Making the floppies

Creating Floppies from DOS

The first thing we will also need to do is get the floppy install tools to build the boot floppies. Then you create the floppies to install from; there is a link at the beginning of appendix I containing the location of the files listed below. To create the kernel floppy image to boot from we will need to copy the files fdimage.exe, kern.flp, and mfsroot.flp to the "c" drive. Run the "fdimage.exe" command from a DOS prompt, you'd do something like this:

```
C:\> fdimage kern.flp a:
```

Creating Floppies from UNIX

If you're creating the boot floppy from a UNIX machine, you will need to key in the following:

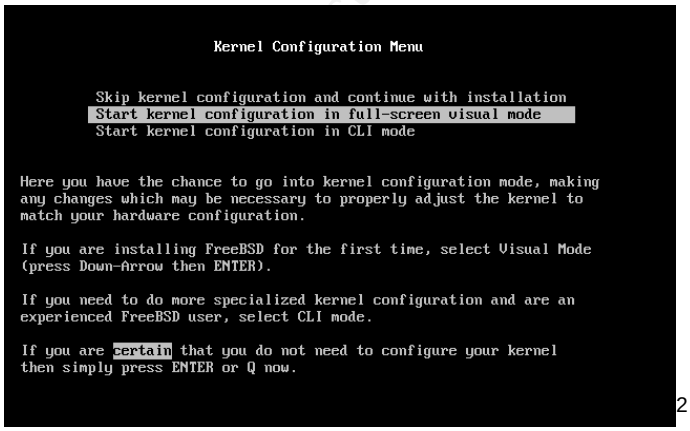
```
# dd if=floppies/kern.flp of=/dev/rfd0
```

Or

```
# dd if=floppies/kern.flp of=/dev/floppy
```

2 – Installing from floppies

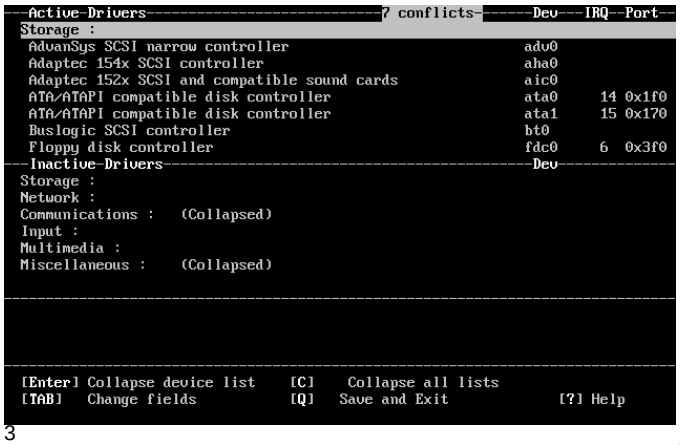
Insert the kern.flp disk and boot the system from the floppy drive. Once the boot image comes up you will be prompted to insert the mfsroot.flp disk. After the system completes the configure phase of the boot process, you will see this screen:



2

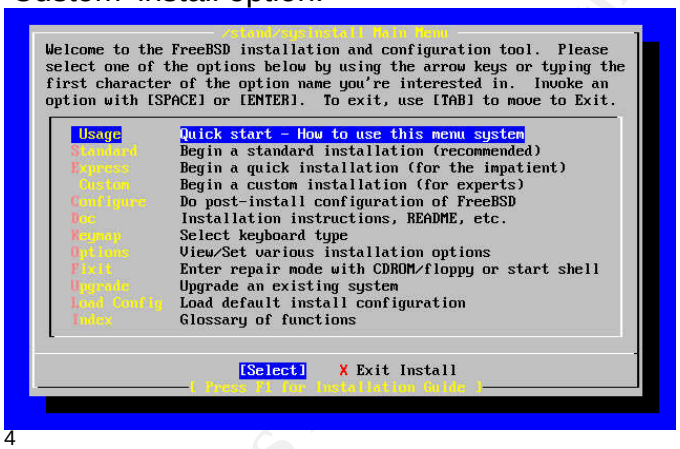
Choose 'Start kernel configuration in full-screen visual mode'. Starting the kernel in this way allows you to resolve any conflicts that may have arose during the automatic configuration process.

2a Now move the highlighted bar down to the conflicts, and key “del” to remove conflict. Once all conflicts are removed, go ahead finish the boot process. Notice the SCSI and ATA conflicts; this box has no SCSI so we need to remove those conflicts



3

2b Now we are presented with the main configuration menu. Select the ‘Custom’ install option.

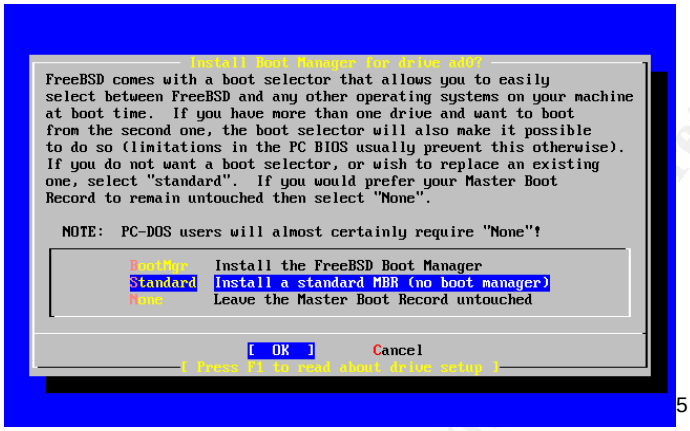


4

2c – Select “a” to use the entire disk as one partition. Also, select “s” to set the disk to be bootable.

```
Disk name:      ad0                              FDISK Partition Editor
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset       Size(ST)        End     Name  PType       Desc  Subtype    Flags

     0             63         62        -      6     unused        0
    63       16514001   16514063    ad0s1      3     freebsd     165    CA

The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice   F = `DD' mode
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable   I = Wizard m.
T = Change Type        U = Undo All Changes     Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

2d Select the “standard” boot manager – otherwise the system will not boot.



5

2e Now you will see the file system configuration menu. The menu will be empty. First key down into the center of the screen. Now create the new file system by pressing “c”. You will need to select the size, type {file system or swap}, and the mount point.

```
                        FreeBSD Disklabel Editor

Disk: ad0       Partition name: ad0s1   Free: 0 blocks (0MB)

Part      Mount          Size Newfs   Part      Mount          Size Newfs
----      -----          ---- -----   ----      -----          ---- -----
ad0s1a    /            128MB UFS   Y
ad0s1b    swap         503MB SWAP
ad0s1e    /var         256MB UFS+S Y
ad0s1f    /tmp         256MB UFS+S Y
ad0s1g    /usr        6919MB UFS+S Y

The following commands are valid here (upper or lower case):
C = Create         D = Delete    M = Mount pt.    W = Write
N = Newfs Opts     Q = Finish    S = Toggle SoftUpdates
T = Toggle Newfs   U = Undo      A = Auto Defaults     R = Delete+Merge

Use F1 or ? to get more help, arrow keys to select.
```

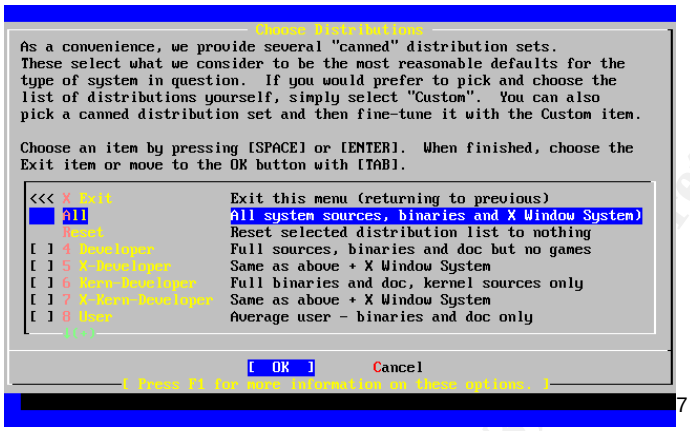
6

Set up the following file systems.

/	1000MB	To assist in not filling up the root file system
swap	500MB	doubles system memory [256MB]
/var/syslog-ng	10000MB	enlarged for for logging space
/usr	6000MB	“usr” is the main source of disk usage

Press “q” when done to save and quit.

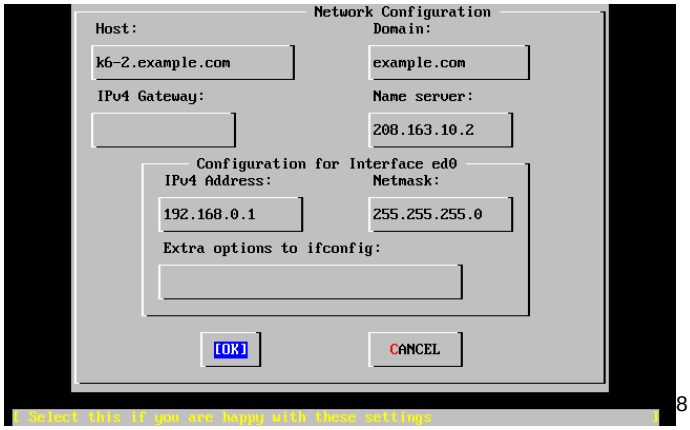
2f Select the distributions to install on the system.



When prompted for distribution sets, select option 6 “user” (move cursor to user and press space) and then select “Custom”, that is option 8. Most of the other components are not useful for our hosts and installing them will only increase the size of the OS and increase the burden to maintain them. At the “Custom” menu, select “man”; this is manual for all binaries, very important component for all administrators. Additional binaries can be later installed from the ports collection on an as-needed basis. After selecting the ”user“ distribution, an opportunity to install the ports collection is presented. (Answer YES to this prompting).

3 - System network configuration

Now we are asked if we would like to configure the network configuration (answer “yes”). Then we are asked to configure IPv6, (to this answer “no”); next we are asked to configure DHCP, (to that we also answer “no”). Finally we are asked to configure IPv4 (to this we answer “yes” we will be putting our ip and net mask here for each system). You should now see this menu,



[8]

3a When prompted for whether to bring up the network interface now, answer (YES). {We will need to pull down the latest sources for building the OS from ftp}.

3b When prompted for this host being a gateway, answer (NO) {there is no need for packet forwarding on this machine}.

3c Then the prompt for configuring inetd and simple Internet services will come up, answer (NO).

3d When prompted for time zone, select (YES) and choose the correct time zone.

4 – Finalization of system install

4a Answer (NO) to Linux Compatibility.

4b We will now be offered to view the options again, answer (YES) and review all options before exiting and beginning the install.

4c We are now given a warning about data loss –

User Confirmation Requested
Last Chance! Are you SURE you want to continue the installation?

If you're running this on a disk with data you wish to save then WE STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!

[Yes] No[9]

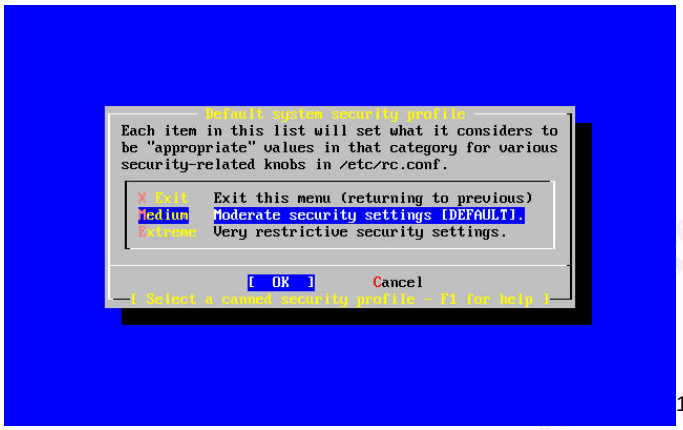
Enter (YES) and the system begins to write the new files system to the disks, and then proceeds to login to the FTP servers and begin the operating system install. {Now … go get a cup of coffee}

5 – Post install

5a Once the system is done downloading and installing, we have an option to do any post-install configurations. At this point we will set the system some of the base security. First we select the (Configure) option from the Main menu.

5b Our first post-install change is to set a new strong root password. We can do this by selecting the (Root Password) option in the configuration menu.

5c Now select the (Security) option, which sets the default system security profile. Select the extreme option {this will disable sendmail, NFS, portmapper, and SSH



[10]

The system will now echo out the following warning:
Extreme security settings have been selected.

Sendmail, SSHd, and NFS services have been disabled, and securelevels have been enabled.
PLEASE NOTE that this still does not save you from having to properly secure your system in other ways or exercise due diligence in your administration, this simply picks a more secure set of out-of-box defaults to start with.

To change any of these settings later, edit /etc/rc.conf

[OK] [11]

Also reference http://www.freebsd.org/cgi/man.cgi?query=init&sektion=8
We will later edit /etc/rc.conf and enable SSHd.

The following will be added to /etc/rc.conf:
kern_securelevel="2"
kern_securelevel_enable="YES"
sendmail_enable="NO"
nfs_server_enable="NO"
sshd_enable="NO"

Section III- securing the base OS

1 – User accounts and security

1a – first time log in

We now need to login and make any modifications to the system or additional configuration changes. At the login prompt, login as the root user

FreeBSD/i386 (machine name) (ttyv0)
Login: root
Password: <Type the password for root user>

1b- Securing password encryption

This option will break RFC compliance. Do not use this option on a web server.

Since this system is running in standalone mode, local authentication requires the storing of an encrypted password file on each local host. FreeBSD comes default with MD5 password encryption, which is considered to be more secure than DES encryption method used by a lot of older UNIX systems (Solaris for example). FreeBSD also supports the "blowfish" cipher for password encryption, which is stronger, that MD5. We will apply the "blowfish" cipher to strengthen this layer of the system security. When editing the file /etc/login.conf we will be setting the default password encryption method for the entire system.

To enable blowfish encryption:
vi /etc/login.conf
Notice the line
":passwd_format=md5:\"
Now change md5 to blf, so it becomes ":passwd_format=blf:\"

Also add the following to set the password defaults.
These changes will do the following, force the password change interval to 90 days. Warn the users to use mixed case passwords. The next change will set the minimum password length to 10 characters. And the final field will log the user out after an idle time of 30 minutes.

```
:passwordtime=90d:\
:mixpasswordcase=true:\
:minpasswordlen=10:\
:idletime=30:\
```

To apply the changes made to the file,
cap_mkdb /etc/login.conf

Now make it the default for all
edit /etc/auth.conf

```
crypt_default  =       blf
```

As the password is still one way encrypted, the system cannot automatically change the encrypted passwords in the password file to the new encryption format. We will have to manually change the passwords for each user on the system in order for the encrypted password in the password file to be in new format. Since there is only one user on the system at this point, the changes should be fairly easy.

```
# passwd root
Changing local password for root.
New password: <Enter password here>
Retype new password: <Enter password again here>
passwd: updating the database...
passwd: done
```

1c - User Creation

Create a normal user and elevate the normal user's group, giving the normal user able to become root user later. use the "adduser" command.

```
#adduser -silent
Use option ``-verbose'' if you want to see more warnings and questions
or try to repair bugs.

Enter username [^[a-z0-9_][a-z0-9_-]*$]:dave
Enter full name []:Normal User Dave Patterson
Enter shell bash csh date ksh no sh tcsh [sh]: (just key return)
Enter home directory (full path) [/home/dave]: (just key return)
Uid [1001]: (just key return)
Enter login class: default []:(just key return)
Login group dave [dave]: (just key return)
Login group is ``dave''. Invite dave into other groups: guest no
[no]:wheel                          { this will allow user dave to su to root }
Enter password []:
Enter password again []:

Name:     dave
Password: **********
Fullname: dave
Uid:      1001
Gid:      1001 (dave)
Class:
Groups:   dave wheel
HOME:     /home/dave
Shell:    /bin/tcsh
OK? (y/n) [y]:
Added user `` dave ''
Send message to `` dave'' and: no root second_mail_address [no]:
dave,
```

your account `` dave '' was created.
Have fun!

See also chpass(1), finger(1), passwd(1)

Add anything to default message (y/n) [n]:
Send message (y/n) [y]: n
Add another user? (y/n) [y]: n

As a “normal” user, FreeBSD only allow minimal privileges. A normal user cannot make changes to the system configuration files. UNIX systems have a “root” account for the purpose of system configuration and maintenance. A root user can do anything to the system. Given this amount of power, it is considered dangerous to login as root for day to day operation because the consequence of a simple and potentially catastrophic mistake are just too great, therefore, it is much safer to operate the system as normal user account and only become the root user as need arises. FreeBSD provides “su” utility to substitute the user’s identity. In order for a given user to have the privileges to ”su” that user must be a member of wheel group (group 0).

Test the newly created account “dave”
Logoff as root user, type
exit
The screen will return to login prompt.
Login as normal user
FreeBSD/i386 (machine name) (ttyv0)
Login: dave
Password: <Type the password for this user>

The user should now be logged in the system (provided that the username and password combination is correct).
> {notice the “>” symbol for the prompt a helpful hint that this is a normal user. Although it is not a good idea to depend on this to verify for certain, run the “id” command}

Now create the second administrators’ account for this machine in the same fashion.

1d - Change permission of root directory

There should not be any normal user other than trusted system administrators logging onto the system, but just in case this happens, the root directory should be properly protected. To enable this enter this command.
chmod 700 /root

1e - Change root's default umask

Root user created files are usually related to system configuration and should be kept as secret. The root user's default file permission should be set to root access only. The default is to allow "group" and "other" read access to the file in which root created. The "umask" essentially says to remove permissions on file creation, essentially the opposite of changing modes with there numeric values.

Simply stated that the umask 077 is specifying that 0 privs are denied (masked) for root and all privs (7) are denied (or masked) for group and user. This should save the root user a lot of time trying to change the permission of each created file. Bear in mind that if there are files that need more access you will need to change there modes as you go along.

At this point, the system needs to be configured with root privileges, so execute "su" to become root.

Edit the .cshrc file or the .profile file
vi /root/.cshrc
Change umask line to
umask 077
This will make all root-created file to be defaulted to root access only

2 – system start-up configuration

2a - Editing rc.conf to altering system configuration

According to the man page (man rc.conf) of rc.conf, "The file rc.conf contains descriptive information about the local host name, configuration details for any potential network interfaces and which services should be started up at system initial boot time. In new installations, the rc.conf file is generally initialized by the system installation utility: /stand/sysinstall." rc.conf is an important file for system configuration, especially true in BSD system architecture where each service does not start in its individual scripts (like Sys V and Linux).

To edit the rc.conf file
vi /etc/rc.conf

Note that in this file there are many lines of configuration already. All of the lines in the file that you see are direct consequences of the options chosen at installation time, such as IP and gateway addresses.
Out of box, FreeBSD is defaulted to accept log entries from other hosts (via the syslog daemon) and a network port (UDP/514) is opened listening to incoming log files. This poses a potential vulnerability and as such we will disable the default syslog daemon. If the port were left open, a person or program with access to that port could potentially write to the log file on the host. With enough log entries, the disk drive may fill up and lead the denial of service (though it is hard to dump that much data without being noticed).

Also, the opened syslog port on the host may be a possible entry point for attacker to take advantage of a vulnerability (there have been many exploits in the past) of syslog and attack the host.
We will be using syslog-ng instead and sending our logs to a remote host. That host will accept our logs and thus be the only host listening on our networks listening on port 514 {though it will be tcp}. With syslog-ng we can configure it to push or pull logs, all hosts on the network will be pushing logs to this host, as it will be the primary log host for our network.

Often, when an attacker attempts to attack on a host they will usually perform reconnaissance on the target host; some of these actions may include "portscan", which scans every port on a system to see what ports are actually open. The opened ports may provide information of weakness for the attacker (such as vulnerable service running on the open port).

FreeBSD provides an option to log all these connection attempts to un-opened ports. By adding the line log_in_vain="YES" to rc.conf, all attempts to closed ports will be logged. {This could produce a lot of log entries and may fill up logging disk. Use with caution especially on the web server.}

tcp_drop_synfin="YES" should be added to rc.conf, this will effectively tell the system to ignore all the TCP packets with SYN and FIN flag set.
The man page points out that the kernel has to be set with "TCP_DROP_SYNFIN" to activate this option. Unfortunately building a custom FreeBSD kernel is outside of the scope of this paper.

2b- Edit sysctl.conf

All parameters in the file /etc/sysctl.conf pass the kernel tunable parameters at boot time. The sysctl feature of the FreeBSD system allows you to pass kernel tunables to the kernel. This can be done either dynamically via the sysctl command or at boot time by the sysctl.conf file in /etc.

In rc.conf the "log_in_vain" option was selected to log all the abnormal or potentially threatening connections. This is only a logging feature and does not eliminate the threat of information leaked. When an attacking host sends a SYN packet to establish a connection, the receiving host would either send a SYN+ACK packet back to continue the connection or send an RST packet to notify that the port is not listening. By monitoring whether SYN+ACK or RST packet is in the reply, the attacker would be able to map out the opened ports on a host. We can configure our machines to not reply and thus the attacker will have to wait till timeout before trying another port that would slow down the scanning process, and possibly dissuade the attacker. FreeBSD has an option to disable sending back the RST packet for unopened ports.
Edit the sysctl.conf by typing

```
# vi /etc/sysctl.conf
```

This file should only contain comments. After the comments, add the following lines

```
net.inet.tcp.blackhole=2
net.inet.udp.blackhole=1
```

2c – sshd configuration

There are a few changes to enable and secure sshd, the secure shell daemon.
1 - edit rc.conf
vi /etc/rc.conf

Change the following line {sshd_enable="NO"} to:
sshd_enable="YES"

2 – edit the sshd_config file {the server file}
vi /etc/ssh/sshd_config

Change the following lines:
Protocol 2,1 {remove the ",1" to take out ssh V1 support}
PermitRootLogin no {verify that it is set to no if not change to no}

3 – edit the ssh_config file {the client file}
vi /etc/ssh/sshd_config

Change the following lines:
Protocol 2,1 {remove the ",1" to take out ssh V1 support}

3 - Additional securing OS

3a - Change permission of system suid and sgid binaries.

Suid binaries allow a user to execute a program as a different user (usually root). Sgid works in similar fashion and allows the user to become another group. Some of the suid binaries are badly implemented and are easily exploited; they could easily lead to local user compromising the machine through these suid and sgid binaries.
The best practice is to use suid and sgid binaries only if necessary and disallow the use of the unnecessary ones. To find all suid and sgid binaries on a machine

Use this simple shell script.
#!/bin/sh

#To find all suid binaries on a machine,
find / -perm -4000 > suid_files
#To find all sgid binaries on a machine,
find / -perm -2000 > sgid_files

Some binaries should never to be used, for those binaries, permission 000 should be given. Setting this will disallow any read, write and execute from any

user. Example, the r-shell commands now very outdated. For the binaries that are only useful to root, permission 500 should be given and should be owned by root, it would only allow root to execute and read them.
The following binaries' permission should be set to 000,

/usr/bin/cu
/usr/bin/uucp
/usr/bin/uuname
/usr/bin/uustat
/usr/bin/uux
/usr/bin/at
/usr/bin/atq
/usr/bin/atrm
/usr/bin/batch
/usr/bin/ypchpass
/usr/bin/ypchfn
/usr/bin/ypchsh
/usr/bin/keyinfo
/usr/bin/keyinit
/usr/bin/lock
/usr/bin/yppasswd
/usr/bin/rlogin
/usr/bin/rsh
/usr/bin/lpq
/usr/bin/lpr
/usr/bin/lprm
/usr/libexec/uucp/uucico
/usr/libexec/uucp/uuxqt
/usr/sbin/mrinfo
/usr/sbin/mtrace
/usr/sbin/sliplogin
/usr/sbin/timedc
/usr/sbin/traceroute6
/usr/sbin/ppp
/usr/sbin/pppd
/bin/rcp
/sbin/ping6

This can be scripted, but be careful in doing so.
To set the above binaries to permission 000, for the above listed binaries,
chmod 000 [binary path/name]

The following binaries' permission should be set to 500, as root should be the only user running these commands.
/usr/bin/crontab
/sbin/shutdown
/usr/bin/chpass
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/su
/usr/local/bin/sperl5.8.0
/usr/local/bin/suidperl

To set the above binaries to permission 500, for each binary,
chmod 500 [binary path/name]

3b - Mount drives

There are several file systems, which shouldn't really have for example suid binaries. If such type of files running on directory such as /tmp is certainly abnormal and should be stopped. Fortunately, FreeBSD allows mount options that will limit the type of operation on a mount point. To set the allowed operation on a mount point,
vi /etc/fstab
For /tmp's option, change to rw,nosuid,noexec,nodev
For /usr's option, change to nodev,rw
For /var's option, change to nodev,nosuid,noexec,rw

The "nodev" option will not allow files to be a device, avoid unnecessary access to hardware devices. This is a clever way to potentially bypass the systems device tree. The "nosuid" option disallows files on the specified file system to run as suid binaries, thus protecting that file-system space. The "noexec" option disallows execution of any files on the specified file system. By limiting the capabilities of files in different file system, normal user's ability on the system is reduced. Even if an attacker gained access as a normal user on the system, it would be harder to exploit local vulnerabilities.

3c – Configuring IPF

Ipfilter and ipnat are part of the base system of FreeBSD. Ipf is a packet filter and works in conjunction with ipnat. First a little info on how IPF works. Ipf is a kernel level packet filter that blocks all by default.
In other words, its reaction to inbound traffic is to block it unless other wise specified by a given "pass in" rule. This will be a fairly quick discussion of each. For a more in depth information please see appendix I for several very good how-tos. To enable ipf we will need to edit /etc/rc.conf again and add the following lines. I would like to point out that it is better to compile ipf into the kernel, rather than call it from rc.conf (but this is beyond the scope of this paper). We will discuss how to add ipf via the rc.conf facility. To do so, edit /etc/rc.conf

#vi /etc/rc.conf
We will now need to add the following lines:

```
ipfilter_enable="YES"            # Set to YES to enable ipfilter functionality
ipfilter_rules="/etc/ipf/ipf.conf"  # rules definition file for ipfilter
ipmon_enable="YES"               # Set to YES for ipmon; needs ipfilter or ipnat
ipmon_flags="-Ds"                # typically "-Ds" or "-D /var/log/ipflog"
```

This will allow for ipf to be loaded from the conf in /etc/ipf/ipf.conf and run ipmon in daemon mode logging to syslog-ng. Below are some elemental concepts that are important to the effective use of ipfilter. Rule sets are read from top to bottom.

1. By default ipf BLOCKS any traffic not specifically specified in the conf file.
2. The "quick" statement in a rule will pass all traffic ONLY via that rule, and bypass all others
3. Traffic type must be specified tcp/udp/icmp.
4. Traffic can be blocked, or passed, as well as logged based on source and destination. This can be granulated by port, port-range, single ip, range of ips, or subnets.
5. UDP rules takes NO flags; remove the "flags S keep state" statement.
6. If you have multiple interfaces EACH one needs its own group groups allow you to track and have the appropriate rule sets for each interface
7. The ipfstat command {run only by root since it accesses /dev/iplog} will display the current stats and or loaded rule sets based upon given flags
8. To view or log to a file packets, run ipmon:

   # ipmon |grep {ip, port or MAC address in question}

   [Also try the ipmon flags for more granularity] ipmon by default logs to STDOUT, as such we will also have syslog-ng configured to log the same traffic.

9. If you wish to see the logged packet, you MUST have the "log" statement in a rule so that ipmon will pick it up.

Given all that has been mentioned there is a base configuration file in appendix I at the end of this document. All ipf configurations will have that base configuration and I will mention only additions or subtractions from that list. We will now need to create both the /etc/ipf directory and the appropriate files.

```
#mkdir –p /etc/ipf
#touch /etc/ipf/ipf.conf /etc/ipf/ipfboot.sh
```

Now copy the contents of the appendices for ipf.conf, and ipfboot.sh. This is your configuration file, and your startup script.

The “ipfboot.sh” will not only reload ipfilter, and apply the new rules based on ipf.conf. To apply the newly created rules run

#ipfboot.sh reload {To verify that your rules actually got applied now run}
Set 1 now inactive

4 – Additional security software

4a – building and configuration of Syslog-NG

Syslog-NG is a replacement for the standard UNIX syslog daemon. It has several features built in both in terms of security and functionally. In terms of security it can run on TCP instead of UDP. In terms of functionality, it can create logs in several different styles. One of the big advantages is that it can create logs on a per program basis and thus finding log “x” is a bit easier. To install do the following:

#cd to /usr/ports/sysutils/syslog-ng/
#make
#make install

This will install the configuration files in /usr/local/etc/syslog-ng Appendix IV contains an example file. The default configuration file is essentially equivalent to the stock FreeBSD /etc/syslog.conf file. The provided sample will log in a different format. The format is essentially to create a directory for each the system as a whole and then a sub-directory for each program running, finally creating a log file for each day name in a “month_day” format. For example it will look like so:

/var/syslog-ng/$HOST/$PROGRAM/$MONTH_$DAY

If the example file is used there will be no need to do anything. The log host configuration file will need to have the below new line, UN-commented so that it can receive log files from other hosts.

“#source net { tcp(ip("192.168.0.0/16") port(514)); };”

We now need to add the following lines to /etc/rc.conf
syslogd_program="/usr/local/sbin/syslog-ng"
syslogd_flags=""

We will need to have the following line added to ipf.conf to allow inbound traffic on tcp port 514

Allow syslog-ng log files from allowed hosts
pass in quick proto tcp from 192.168.0.0/16 to 192.168.5.46/32 port = 514 flags S keep state

4b – building and configuration of AIDE

AIDE is short for Advanced Intrusion Detection Environment. AIDE is intended to be a replacement and extension for Tripwire.

> "Tripwire is a tool that aids system administrators and users in monitoring a designated set of files for any changes. Used with system files on a regular (e.g., daily) basis, Tripwire can notify system administrators of corrupted or tampered files, so damage control measures can be taken in a timely manner."[12]

The use of AIDE will make the monitoring of system binaries much easier and help to keep a close eye on system changes. To install go the following directory and install the port.

#cd to /usr/ports/security/aide
#make
#make install

This will install the configuration files in /usr/local/etc/aide. There is a default aide.conf file in appendix V. A good way to run aide is out of the cron facility. Add the following line:
45 12 1 * * /usr/local/etc/aide/aide.sh

#crontab –e {"e" for edit}

The script /usr/local/etc/aide/aide.sh will need to be modified to the send the output to the appropriate email address. This script will now be run out of cron and send mail notifying the recipient of any changes to the file system specified in the aide.conf file.

Section III -Testing of configuration
1 – verifying ipf rule sets are loaded

In order to verify that IPF rules are loaded correctly.
To do this run ipfstat –i {for inbound rules}

#ipfstat –a {you should now see something similar to this output}

pass out on sis0 from any to any head 150
block out from 127.0.0.0/8 to any group 150
block out from any to 127.0.0.0/8 group 150
pass out quick proto tcp/udp from any to any keep state
pass out quick proto icmp from any to any keep state
block in log quick from any to any with frag
block in log quick from any to any with short

```
block in log quick from any to any with ipopt
block in log quick proto tcp from any to any with short
pass in on sis0 from any to any head 200
block in log quick from 192.168.3.0/24 to 192.168. 5.46/32 group 200
pass in quick proto icmp from 192.168.3.0/24 to 192.168. 5.46/32 keep state
group 200
pass in quick proto icmp from 192.168.3.0/24 to 192.168. 5.46/32  keep state
group 200
block in quick from 127.0.0.1/32 to any
pass in quick proto tcp from 192.168.3.0/24 to 192.168. 5.46/32 port = 22 flags
S/FSRPAU keep state
block in log on sis0 from any to any
```

2 - Portscan – testing for open ports

Almost all production server systems should be tested with portscans before they are put into production. Portscans are used to find un-intended open ports on a given system. In procedure section II 2.a, option “log_in_vain” is activated to log any connection attempts to non-listening port at syslog. However, there is also local firewall software blocking all the unauthorized connection attempts, if there is any leak of packets by the firewall, the “log_in_vain” option should respond by logging the attempt to syslog-NG. To test we will use Nmap, to verify that no un-intended ports are left open.

A testing host is located on the same network segment as the log host server without the isolation by any firewall host. Now we run an Nmap scan, attempting to connect to our log host on each of the 65,535 available ports. The command below will start a scan on the log host, searching for any listening services on the host.

```
# nmap –P0 –p 1-65535 –sS 192.168.5.46

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost (192.168.5.23):
(The 65528 ports scanned but not shown below are in state: closed)
Port       State      Service
22/tcp     open       ssh
514/tcp    open       shell       {although this is really syslog-NG}

Nmap run completed -- 1 IP address (1 host up) scanned in 379 seconds
```

There are no open ports on the server {other than what we expected}; also due to the ipf blocking access this is now confirmed. The log should show all the unsuccessful connection attempts done by the portscan. To verify this, execute:

tail –n 400 /var/syslog-ng/kernel/month_day | less

This should show many unsuccessful connection attempts which resulted from the portscan.

ksrc@host Connection attempt to TCP 192.168.5.23:61718 from 192.168.
5.46:34722
ksrc@ host Connection attempt to TCP 192.168. 5.23:19958 from 192.168.
5.46:34722
ksrc@ host Connection attempt to TCP 192.168. 5.23:64996 from 192.168.
5.46:34722

This proves that the unsuccessful attempts are logged.

To verify that ipf is blocking portscans

#tail –f /var/syslog-ng/loghost/ipmon/"month_day" {tail the current day's file of course}

sis0 @0:37 b 192.168.5.23,50235 -> 192.168.5.46,44470 PR tcp len 20 40 -S IN
sis0 @0:37 b 192.168.5.23,50235 -> 192.168.5.46,26265 PR tcp len 20 40 -S IN
sis0 @0:37 b 192.168.5.23,50235 -> 192.168.5.46,28645 PR tcp len 20 40 -S IN

The Nmap result on this test shows the only port opened for this host is TCP ports 514 and 22 which is the port for syslog-ng and sshd, this is exactly as intended.

3 - Testing passwords
Verifying that the blowfish cipher is used:

The use of the blowfish cipher confirmed by the "2" at the beginning of the /etc/master.passwd file.
#grep dave /etc/master.passwd
dave:$2a$04$548VvjMi88NIeoCnUmj1puKpSXts.sjaGVnXQxbmHCkhNc6kmnfY
S:1000:1000::0:0:me:/home/dave:/bin/tcsh

Testing min-length
#passwd dave {now put in a 5 character password to test length enforcement}
Changing local password for dave.
New password:
Please enter a password at least 10 characters in length.

4 - Testing ssh

Logging blocked packets from internet:

sis0 @0:37 b 207.55.99.157,52170 -> 192.168.5.46,22 PR tcp len 20 60 -S IN
sis0 @0:37 b 207.55.99.157,52170 -> 192.168.5.46,22 PR tcp len 20 60 -S IN

Tailing the sshd log shows:

src@host sshd[3013]: Server listening on 0.0.0.0 port 22.
src@host sshd[3026]: ROOT LOGIN REFUSED FROM 192.168.0.13 {blocking locally
src@host sshd[362]: ROOT LOGIN REFUSED FROM 207.55.99.157 {blocking remotely }
src@host sshd[338]: Illegal user toor from 207.55.99.157 {blocking guessed user accounts}
src@host sshd[338]: Failed unknown for illegal user toor from 207.55.99.157 port 52707 ssh2
src@host sshd[355]: User mysql not allowed because shell /nonexistent does not exist {blocking user accounts /w no shell}
src@host sshd[355]: Failed unknown for illegal user mysql from 207.55.99.157 port 52739 ssh2

5 - Testing AIDE

Using the aide.sh script from appendix VII, we can easily verify the status of the files on the system.

A clean system will email you back the following:

Below is the output from loghost's AIDE check that occurred
on Thursday.

A file system that has been modified will email the following: {notice the added files}

Below is the output from loghost's AIDE check that occurred
on Thursday.

Not implemented in db_readline_file 311
"@@end_db"AIDE found differences between database and file system!!
Start timestamp: 2003-05-29 16:28:49
Summary:
Total number of files=2508,added files=3,removed files=0,changed files=0
Added files:
added:/usr/local/dave
added:/usr/local/dave/testfile1
added:/usr/local/dave/testfile2

6 - Testing logging

To verify that the system is logging locally run the following:

```
logger -p auth.info -t DAVE testing authlog
cd /var/syslog-ng/loghost/DAVE/
tail –f $MONTH_$DAY
src@host DAVE: testing authlog
```

To verify that the system is logging remotely run the following:

```
On a remote system
logger -p auth.info –h loghost -t DAVE.remote testing authlog
cd /var/syslog-ng/remotehost/DAVE.remote/
tail –f $MONTH_$DAY
src@host DAVE: testing authlog
```

Section IV – On going system maintenance

1 - OS updating

This section is included mid-way through the process of securing the OS in order to make sure that we have the most up-to-date sources to start from. Updating is an essential part of maintaining a secure system; it is also the way to get rid of vulnerabilities on a system. In particular the ports-collection, {the repository of downloadable software} can become out of date and it's a good idea to update is regularly.

As discussed above in the system installation stage, FreeBSD uses CVSup utility in order to update the source code of the system as well as the ports tree. It is essential that the system have the most up to date source and port tree for the system re-compile to be effective.

A cron job can be setup so the host will automatically update the source and port tree every night. Due to the effective CVSup utility, only the source updated on that day will be downloaded. Setup a cron job to automatically update the source and port tree at night, it is an easy way to "set it and forget it"

vi /etc/crontab

Add the following line

0 5 * * * root /usr/local/bin/cvsup -g -L 2 /root/cvs-supfile

This will activate CVSup every day at 5 AM to update the source and port tree. You will of course need to copy the example cvs-supfile to /root.

#cp /usr/local/share/examples/cvsup/cvs-supfile /root/

This way, if the system administrator received a notice from FreeBSD's mailing list regarding a vulnerability in FreeBSD OS, the new source can be downloaded very quickly (due to minimal difference within 24 hours) and then re-compile to get rid of new vulnerability.

There is also a script that runs from /etc/periodic/weekly/ that checks the version of packages against the cvs repository, and mails that output to root.

2 - ports updating

For ports, the procedure is similar, if a vulnerability of a software package on the database server is known and the update is available on the port tree. To make sure the port tree is synchronized, run CVSup again and then upgrade that specific port. There is however, one issue. The ports are compiled from source and may require other software dependencies. So in the process of upgrading port "x", the intention of upgrading only port "x" may imply upgrading a few dependant ports as well. There is a very handy tool that can take care of this dependency problem by upgrade all the dependant ports. This is the function of "portupgrade", a utility in a port tree. It is located in "/usr/ports/sysutils/portupgrade". When using portupgrade, just type "portupgrade –rv <name of package to update>".

Software packages are updated for a variety of reasons, often in fact for a new feature and not a security bug fix. These are not announced in the security

announcement lists. Given that security and stability matter more that might given new features, as such be aware of what you are updating and why. To examine the installed packages having newer version available, the command "pkg_version" would provide the packages status (whether a new version is available) provided that the port tree is up to date. Once again the "/etc/periodic/weekly" script will send email to that box's root user account.

It is important to note that during a system update (build installworld); some of the disabled suid binaries will be reset to original suid state. It might be a wise idea to write a script to change all the unnecessary suid binaries back to disabled state instead of doing it manually.

3 - Subscribe to mailing list

It is important to have quick and accurate information regarding any possible vulnerability on hosts we administer, for effective patches and fix to get ahead of the attackers. One of the best sources of such security information is usually available directly from the software developer. That being said it is important to subscribe to security bulletin mailing lists.

FreeBSD also has a webpage dedicated to the OS security at
http://www.freebsd.org/security/index.html
FreeBSD security related announcement mailing list:
freebsd-security-notifications@freebsd.org
freebsd-announce@freebsd.org
Refer to: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html

Bugtraq mailing list usually has the latest bug or vulnerabilities covered or discussed.
Bugtraq mailing list
Refer to: http://online.securityfocus.com/cgi-bin/subscribe.pl

Appendices:

Appendix I – Software sources

FreeBSD:

- ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/4.8-RELEASE/tools/fdimage.exe
- ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/4.8-RELEASE/floppies/

IPF: {Is part of the default FreeBSD OS but can also be obtained}

- http://coombs.anu.edu.au/~avalon/
- Several helpful how-tos
- http://www.obfuscation.org/ipf/ipf-howto.html
- http://ipf.phildev.net/

Appendix II. – ipf.conf

```
#  Begin IPF.conf

# outbound rules
# 1.  Disallow spoofing
# 2.  Allow all other outbound traffic, replies are inspected by stateful filter (not just SYN/ACK).
pass out on sis0 all head 150
block out from 127.0.0.0/8 to any group 150
block out from any to 127.0.0.0/8 group 150
pass out quick proto tcp/udp from any to any keep state
pass out quick proto icmp from any to any keep state
#
# inbound rules
#Block all fragmented packets
block in log quick all with frag
#Get rid of all short IP fragments (too small for valid comparison)
block in log quick all with short
block in log quick from any to any with ipopts
block in log quick proto tcp from any to any with short
pass in on sis0 all head 200
block in quick from 127.0.0.1/32 to any
block in log quick from 192.168.0.3/32 to any group 200
# 1. Allow SSH from allowed networks
pass in quick proto tcp from 192.168.5.0/24 to 192.168.5.46  port = 22 flags S keep state
# 2. allow syslog-ng for entire private net
```

```
pass in quick proto tcp from 192.168.0.0/16 to 192.168.5.46 port = 514 flags
S keep state group 200
# 3. Allow pings from allowed networks {helps limit port scans}
pass in quick proto icmp from 192.168.0.0/24 to any keep state group 200
# 6. Default is DENY {blocks all ports not specified above}
#
block in log on sis0
```

Appendix III. – ipfboot.sh {startup script}

```
#!/bin/sh
#
PIDFILE=/var/run/ipmon.pid
PATH=${PATH}:/sbin
IPFILCONF=/etc/ipf/ipf.conf
IPNATCONF=/etc/ipf/ipnat.conf
case "$1" in
        start)
                if [ -r ${IPFILCONF} ]; then
                        if `/sbin/ipf -V | \
                             nawk '$1 == "Default:" && $2 == "pass" { exit 1 }'` ; then
                        fi
                        ipf -IFa -f ${IPFILCONF}
                        if [ $? != 0 ]; then
                                echo "$0: load of ${IPFILCONF} into alternate set failed"
                        else
                                ipf -s
                        fi
                fi
                ipf -y
                        if [ $? != 0 ]; then
                                echo "$0: load of ${IPFILCONF} into alternate set failed"
                        else
                                ipf -IF a
                        fi
                if [ -r ${IPNATCONF} ]; then
                        ipnat -CF -f ${IPNATCONF}
                        if [ $? != 0 ]; then
                                echo "$0: load of ${IPNATCONF} failed"
                        fi
                fi
                ipmon -Ds
                ;;
        stop)
                if [ x"$pid" != x ] ; then
                        kill -TERM $pid
```

```
        fi
        ;;

    reload)
        if [ -r ${IPFILCONF} ]; then
            ipf -I -Fa -f ${IPFILCONF}
            if [ $? != 0 ]; then
                echo "$0: reload of ${IPFILCONF} into alternate set failed"
            else
                ipf –s
            fi
        fi
        if [ -r ${IPNATCONF} ]; then
            ipnat -CF -f ${IPNATCONF}
            if [ $? != 0 ]; then
                echo "$0: reload of ${IPNATCONF} failed"
            fi
        fi
        ;;
    reipf)
        if [ -r ${IPFILCONF} ]; then
            ipf -I -Fa -f ${IPFILCONF}
            if [ $? != 0 ]; then
                echo "$0: reload of ${IPFILCONF} into alternate set failed"
            else
                ipf -s
            fi
        fi
        ;;
    *)
        echo "Usage: $0 (start|stop|reload)" >&2
        exit 1
        ;;
esac
exit 0
```

Appendix IV – syslog-ng.conf

```
#
options { long_hostnames(off); keep_hostname(yes); sync(0); create_dirs(yes);
chain_hostnames(on); time_reopen (30); };
##### SOURCES ############################
#-- network config                              # The network must be heard...
#source net { tcp(ip("192.168.0.0/16") port(514)); };

#-- localhost config
```

```
source src { unix-dgram("/var/run/log"); tcp(); internal(); };
source ksrc { file("/dev/klog"); };

############## DESTINATIONS #############
destination loghost { tcp("192.168.5.46" port(514)); };
destination d_program  { file("/var/syslog-
ng/$HOST/$PROGRAM/$MONTH_$DAY" perm(0640));
};
destination d_kernel  { file("/var/syslog-ng/$HOST/kernel/$MONTH_$DAY"
perm(0600));
};

#--  local stuff goes local...
log { source(src); destination(d_program); };
log { source(ksrc); destination(d_kernel); };

# Send everything to the local monitor machine...
log { source(local); destination(loghost); };
```

Appendix V. – aide.conf

aide.conf – preliminary comments are removed for brevity

```
database=file:///var/db/aide/databases/aide.db
database_out=file:///var/db/aide/databases/aide.db.new

#  First, root's traditional "home".  Note that FreeBSD's root's home (/root)
#  is protected by R-tiger-rmd160-sha1 protections in the default config file.
=/$            L
/.rhosts       R
/.profile      R
/.cshrc        R
/.login        R
/.exrc         R
/.logout       R
/.forward      R

# UNIX itself
/kernel        R

# /bin
/bin           R-tiger-rmd160-sha1

# /dev
/dev           L
```

```
!/dev/tty*              L

# /etc
/etc                  R-tiger-rmd160-sha1
/etc/aliases            L
/etc/dumpdates          L
/etc/motd               L

/etc/passwd             L
 /etc/master.passwd     L
/etc/pwd.db             L
/etc/spwd.db            L

# /home
=/home$                 L-c

# /lkm
/lkm                  R-tiger-rmd160-sha1

# /root
/root                 R-tiger-rmd160-sha1
/root/.history          L
!/root/.viminfo         L

# /sbin
/sbin                 R-tiger-rmd160-sha1
# /stand
/stand                R-tiger-rmd160-sha1

# /usr/bin
/usr/bin              R-tiger-rmd160-sha1

/usr/include          R-tiger-rmd160-sha1

/usr/lib              R-tiger-rmd160-sha1

/usr/libdata          R-tiger-rmd160-sha1

/usr/libexec          R-tiger-rmd160-sha1

/usr/local/bin        R-tiger-rmd160-sha1

/usr/local/etc          L

/usr/local/lib        R-tiger-rmd160-sha1
```

```
/usr/local/libexec      R-tiger-rmd160-sha1
/usr/local/sbin         R-tiger-rmd160-sha1

/usr/local/share        R-tiger-rmd160-sha1

/usr/sbin               R-tiger-rmd160-sha1

/usr/share              R-tiger-rmd160-sha1

######## End ############
```

Appendix VI. – aide.sh {aide notification script}

```
#!/bin/sh

# Variables can be our friend.
DAY=`date '+%A'`
MAIL_ADDR="root_user@domain.com"
SERVER=`uname -n`
SUBJ="$SERVER Aide Notification"
AIDE="/usr/local/bin/aide --config=/etc/aide/aide.conf"
INIT="/usr/local/bin/aide --init --config=/etc/aide/aide.conf"

# Let's verify if there is an active DB
if
  [ -f "/var/db/aide/databases/aide.db.new" ]
then
mv /var/db/aide/databases/aide.db.new /var/db/aide/databases/aide.db
fi
if
  [ ! -f "/var/db/aide/databases/aide.db" ]
then
echo "Initializing AIDE DB,..."
$INIT
mv /var/db/aide/databases/aide.db.new /var/db/aide/databases/aide.db
else

# Of course we need to differentiate between days.
$AIDE > /etc/aide/aidelog 2>&1

chmod 440 /etc/aide/aidelog
# Let's e-mail the log file to everyone who cares.
if
  [ -s "/etc/aide/aidelog" ] then

AIDELOG=`cat aidelog`
```

```
mailx -s "$SUBJ" $MAIL_ADDR << END || echo "Could not send e-mail" >>
aidelog
Below is the output from $SERVER's Aide check that occurred on $DAY.
-------------------------------------------------------------------
$AIDELOG
_____________________________________________________
END
fi
```

Sources:

FreeBSD install how-to
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install.html#INSTALL-SYNOPSIS

Silver, Mark. “A basic guide to securing FreeBSD 4.x-STABLE”
http://draenor.org/securebsd/secure.txt

FreeBSD Security how-to
http://people.freebsd.org/~jkb/howto.html

FreeBSD man pages

The definitive IPF how-to
http://www.obfuscation.org/ipf/ipf-howto.html

Phil Dibowiz’s IPF FAQ
http://ipf.phildev.net/

Syslog-NG reference
http://www.balabit.com/products/syslog_ng/reference/book1.html

AIDE reference
http://www.cs.tut.fi/~rammer/aide/manual.html

[1] Taken from http://www.freebsd.org
[2] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-start.html
[3] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-start.html
[4] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/using-sysinstall.html
[5] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-steps.html
[6] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-steps.html
[7] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-choosing.html
[8] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-post.html
[9] Taken from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-final-warning.html
[10] http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-post.html
[11] http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-post.html
[12] Taken from http://www.freebsd.org/cgi/url.cgi?ports/security/tripwire/pkg-descr