Global Information Assurance Certification Paper

Securing Slackware 8.0

Written by
Peter Seth DeVries

SANS GCUX Practical Assignment
Version 1.6d - Option 1

This page intentionally left blank.

This paper will describe the setup process for installing and securing Slackware 8.0. This configuration will always strive to be more secure rather than more feature filled. The 2.2.19 kernel will be used. Applications that are not needed to have a functioning Linux box will be removed. Later we will get into installing a few different applications to make the server more secure.

While progressing through this install there will be points where you will need to compile applications. Since the computer being built will not have any of the development tools, the compilation will need to get done on a second machine. Files will then have to be copied over to the new system via CDROM or secure network connection as you will see below. The second machine does not have to be exactly like the secured machine but the closer the two systems are the better. The systems I used to step through this document were as follows. The compile machine is a 1.4Ghz Athlon, 384 Ram, 60+ gigs HD, running slackware 8.0 with kernel 2.4.7. Source code for the 2.2.19 kernel is installed in /usr/src/linux-2.2.19 and when compiling a application for the secure machine a symbolic link /usr/src/linux points to the 2.2.19 directory rather than the 2.4.7 directory. This way any compiling done that looks for include files will use the appropriate ones for the 2.2.19 kernel. The secured machine is a 450Mhz Intel processor, 128 Megs Ram, 8 Gigs HD, running the install you will see below. As you can see my two boxes are running different configurations and no problems running code on the secured box have be had.

At the end of this document you will have a system that is as secure as can be. There is no such thing as a completely secure system. The threat will always be out there and people will always find new ways to circumvent different mechanisms. Be sure to stay up to date on the systems configuration and logs to ensure the highest level of security you can achieve.

The system built will be running a DNS server, and some security applications. The intention of the document is not to isolate the reader to building one type of system, but rather to educate them on the securing of the operating system and it's components. The DNS service will be set up to show an example of chrooted environments (See the chrooted environment section below). A user who is looking to set up a workstation machine would not want to follow this document exactly because it would lack a X windows environment and other basic functionality that one might expect on a desktop. The primary use of the system built would be as a server of some sort, below we will have a DNS server. If you do not need DNS services or sendmail feel free to skip those sections, they are there for informational instructions not because they are required.

You will need a little vi experience to get through this document. I've tried to make it as easy as possible but you'll need to know how to navigate in vi. Throughout the document text is printed different ways. *Italics are things that you will type.* small type is text from the test computer or scripts you can use.

Installing the OS

Before installing the operating system unplug the network cable. This will prevent an attack before you are able to harden the system. This may sound a little far fetched but it has

happened before. A systems administrator I work with had this problem before when installing a box for a bank. Before the system was even finished the developers had found the box and started to leave themselves backdoors so they could avoid asking for help when root access was needed.

___ Unplug the network cable
___ Boot from the Slackware 8.0 boot media (This install will follow the CD installation but you should be able to use any media you like)

If this is a production box you should definitely use the 2.2.19 kernel. If you are building something for test you can use 2.4.5 and most of this document should work for you. Your mileage may vary (ymmv). For this example the 2.2.19 kernel will be used.

___ boot with *bare.i* or enter. If you need scsi use *scsi.i*
___ Select your keyboard map (<*Enter*>)
___ login as *root*
___ Partition your harddrive

I use *cfdisk* to do this. There are no specific requirements on how you need to partition your drive. You will want to have at least separate /, /usr, /var, and /home partitions. This is done to allow us for more flexibility in configuring the drives (See the fstab section later in this paper) and to make it easier to modify the system configuration in the future. For example if you wanted to upgrade your distribution to Slackware 9 in the future but don't want to lose data in /home you can accomplish this very easily since the data is on it's own partition. It also allows for different access restrictions for each partition, which as you will see later, helps lock down the system considerably

I will use the following as a guideline in this paper.

24 megs	- /boot	primary
1 gig	- /usr	logical
2 gig	- /usr/local	logical
1 gig	- /var	logical
1 gig	- /	logical
<2x RAM>	- swap	logical
500	- /home	logical

Partition assignments get created during the setup phase *cfdisk* just partitions the drive space. Don't worry if you can't name the partitions here. You shouldn't be able to.

Depending on the use of the system you may want to adjust the size of the some of the partitions. In our scenario we will assume that only sysadmins will have an account on this box, consequently the home partition will be very small <500. You also may want to have a /opt or /srvr if you will be using the system for a lot of different servers in the long term. If you are going to do a lot of logging you may also want to increase the /var partition.

© SANS Institute 2000 - 2005 Author retains full rights.

___ change one partition to swap - (*type, 82*)
___ *write* the partition information to disk (type *yes)* You may get a message that says no partition is set to bootable. Since DOS or Windows won't be running on the system this can be ignored.
___ *quit*

___ run *setup*
___ Add swap space (*ADDSWAP*) your swap partition should be detected. If not you need to go back into cfdisk and set one partition as swap.
___ Add target drives (You should be prompted after setting a swap partition. If not select *TARGET*). The first drive you select must be the one you want as root (/). After this has been assigned you can create your other partitions.
___ Select Source media (*YES* or *SELECT*)
___ Go to Select (*YES* or *SELECT*)

The only package series that are needed to run a system are

- Base Linux System
- Networking (Actually this isn't needed but what fun is a system without networking. On the other hand if you have no use for a network, you can leave this off and gain some more security)

You may want to install other packages if they are needed for the use of your system. Some server packages such as mySql and Apache are included and compiled on the CD. If you're building one of these servers check how it was compiled and see if it fits your needs. For this example we will install the minimum requirement so we can achieve a higher level of security.

___ Select only the packages:
A) Base Linux system
N) Networking
___ Click *OK*
___ goto the INSTALL section (*YES* or *INSTALL*)
___ Choose *expert* as prompting mode.
___ Remove the following from the Package A select packages screen: scsi, i245, sc245, bash1, cpio, getty, gpm, isapnp, kbd, loadlin, lpr, minicom, mod245, pciutils, pcmcia, reiserfs, tcsh, umsprogs
___ OK - The system will install the packages you selected and then bring you to the select screen for networking
___ Select only the following from the Networking select packages screen
___ ipchains - Later this will be used to build a personal firewall on the system.
___ ntp4 - Network Time Protocol
___ openssh - For secure connections to the system
___ openssl - Secure Sockets
___ pine - So mail messages can be read

___ tcpip1 - The TCP/IP networking programs
If you need sendmail on the system do not use the package included here (version 8.11.4). There is a security flaw that should be fixed by using 8.11.6. You will need to manually install sendmail later (see the patches section)

___OK
___ *Yes*, Continue to configure your system
___*format* - format a floppy disk
___ *lilo* - make lilo bootdisk - *yes* to confirm
___ *continue*
___ *no modem* (Unless you have one)
___ *simple* lilo setup.
___ *standard* - use the standard screen display.
___ *MBR* - I usually install on the MBR with no problems. If you have trouble installing here you can make a bootdisk to boot off and fix lilo later. Help docs can be found at www.linuxdocs.org.
___ Say *No* to configure your network. This will be done manually later.
___ *UTC* or *Local*
___ Select timezone
___ Select *Y*es to set the root password

A word about Passwords and Passphrases

A few things to know when setting up applications or userids that require passwords and passphrases. Passwords are the first line of defense against compromise. If you are not going to bother having a good password then the rest of this document really isn't going to help you. If you are still reading and do want to have a good password here are some guidelines for choosing one.

- Alphanumeric
- No words or phrases. Scramble things up a bit.
- Do not write your passwords down
- NEVER store your passwords in electronic format
- The special characters below are available for use in passwords

` ~ ! @ # $ % ^ & * ( ) - _ = + { [ ] } \ | ' " ; : , < . > / ?

My tip: Try to concatenate the first letters of phrases to remember
ex. I hate dealing with my developers - IhdwmD#%9

Passphrases should be longer than passwords and should follow the same rules. Never write your passphrases down.

Setup complete. Remove your boot media and floppies and reboot. Your system should boot into lilo and then after selecting *Linux* should end up at a login prompt.

Hardening the OS

Create user id

You should now be able to login with root and the password you set before. The first thing to be done is to set up a user account. While most of the steps require root privileges it's still good to get in the process of su'ing into root only when needed. In fact later in the paper root will be locked out for direct logins so a userid will be required.

___ login as root
___ *useradd <userid> -d /home/<userid> -s /bin/bash -g users* -m
___ *chmod 700 /home/<userid>*
___ *passwd <userid>* *<enter>* Enter a new password twice

Now that a user account has been created for logins the system can be navigated safer since the account does not have access to write in the configuration areas. While continuing through the setup look at the files to be changed before opening them with root in case mistakes are made.

/etc/inittab

The /etc/inittab devices controls the way the init process is run. Within it you can disable a few potentially dangerous options.

___ *vi /etc/inittab*
    ___ Disable ctrl-alt-delete reboots by commenting out this line
        ca::ctrlaltdel:/sbin/shutdown -t5 -rf now
    ___ Require the root password to enter single user mode by adding the line
        ~~:S:wait:/sbin/sulogin right after the su:1S:sysinit:/etc/rc.d/rc.S
    ___ Close the file and save with *:wq*
___ *init q* - Will make changes active immediately

/etc/securetty

You can prevent anyone from logging in as root at the terminal by doing the following. Note Beforehand: It is especially important if you can't guarantee the physical security of the box. Also make sure that you have created a user account to login with before this is done.

___ *vi /etc/securetty*
___ *Comment out console and tty1 through tty6*. This prevents root from login directly into the system. Users will need to login through their UserID and then *su* to root.
___ Close the file with *:wq*
Changes will take effect immediately

/etc/rc.d/rc.modules

© SANS Institute 2000 - 2005 Author retains full rights.

By default a lot of modules that aren't needed are loaded at bootup. You can see these modules by typing *lsmod.* There are slip, ppp, and printer support modules, if you do have a use for one of these you can leave it, but be sure to check up on any possible security risks. You can prevent these from booting by modifying /etc/rc.d/rc.modules. Open the file with *vi* and comment out the following sections to the next blank line:

___ "### PC parallel port support"
___ "### Parallel print support"
___ /sbin/modprobe slip
___ "# Load PPP by default:"
___ Close the file *:wq*

The modules that are currently installed can be removed using *rmmod* or the machine can be rebooted and the modules will not be loaded.

Reconfiguring lilo

Personally, I don't like the way Slackware 8.0 sets up lilo and more importantly the location of the kernel. If a /boot partition was created as the first 16 - 24 megs on the disk the it is suggested this is done.

___ *mv /vmlinuz /boot/*
___ *vi /etc/lilo.conf*
___ change *image = /vmlinuz* to *image = /boot/vmlinuz*
___ */sbin/lilo* -You always need to run lilo if you make changes to the kernel or lilo.conf

If the system isn't physically secured then it may be desired for lilo to require a password when it boots. On the other hand, if you remotely control the system a lot and may want to reboot when not around, this may not be a good idea.

___ *vi /etc/lilo.conf*
___ add the following lines after the prompt line
password = <secure password>
restricted

___ *chmod 600 /etc/lilo.conf* Now that lilo.conf has a password it should be readable only by root.
___ */sbin/lilo* - Remember to run lilo after making changes to lilo.conf or kernel changes.

User and Password Files

The userid, passwords and groups are stored in three files: /etc/passwd, /etc/shadow, and /etc/group respectively. Examples of each are shown:

/etc/passwd root:x:0:0::/root:/bin/bash
/etc/shadow root:<your encrypted passwd>:0:::::

/etc/group root::0:root

Slackware does a good job of locking down all the accounts that you have no use for. You can see this in the /etc/shadow file. The * in the second field designates a character that will never be generated by encrypting a password. You may also see people use =LOCK= here since = also is an excluded character.

Since all the accounts are already locked we won't bother going through removing them. However, it is important that you know what these files are and what they do in case you need to lock out accounts in the future.

Unneeded Files

By default some files get installed with packages that are not actually needed by the system. The first set of files that can be remove are the man pages. Since the above install didn't include *man* there is no reason for the files. There actually is a security flaw in /usr/man/cat*. The problem is that all of the directories are world writeable allowing a user to create symlinks and further exploit the system. This does of course require *man* to be installed as well, which hasn't been done here. You can see the full exploit at: http://www.securityfocus.com/vdb/bottom.html?vid=3054. If you do install the man pages and *man* remember to fix this (*chmod 755 /usr/man/cat**)

___ *rm -rf /usr/man*
___ *rm -rf /usr/doc*
___ *rm -rf /usr/info*
___ *rm -rf /usr/share/man (Symbolic link)*
___ *rm -rf /usr/share/doc (Symbolic link)*
___ *rm -rf /usr/share/locale*

Removing unneeded rc files. The following can be deleted safely.
___ rc.nfsd ___ rc.serial (unless you need serial port access)

Compiling a new kernel

The default install of kernel 2.2.19 comes with everything installed as a module. The reason for this is that it allows the system to be more compatible on the different types of hardware it may be installed on. Unfortunately it also adds a lot of space and security issues to the system. I shall not go too deeply into the complexities of compiling the kernel and what the options are. Documentation on that can be found at www.linuxdocs.org or in the kernel source. However, if you get rid of all the modules and compile the drivers you need directly into the kernel, you'll buy back that space and security.

Quick steps to building a new kernel:
___ make config or menuconfig or xconfig
___ make dep

___ make clean
___ make bzImage
___ make modules - if you needed to compile modules.
___ Copy over the new kernel to the production system.
___ edit lilo.conf and **run lilo**

Network Hardening

Before we even set up the network we are going to disable any services that will accept connections to your box. To do this we will take 2 steps. Either one of these steps will disable services but it's better to do both to be safe. In addition if you only do the first step you will have an unneeded daemon running on the system

/etc/inetd.conf

___ Comment out ever line from the /etc/inetd.conf (*vi /etc/inetd.conf*, add comments, *:wq)*

/etc/rc.d/rc.inet2

Remove the inetd daemon startup from the /etc/rc.d/rc.inet2
___ *vi /etc/rc.d/rc.inet2*, comment all lines between "# Start the inetd server:" and
"# Done starting the inetd meta-server"

Since we're already editing the /etc/rc.d/rc.inet2 file lets make some other changes as well.

Turn of IP Forward in rc.inet2.
___ Change IPV4_FORWARD=1 to *0* - Unless you are building a firewall system in which case this needs to be on.

Disable NFS services from rc.inet2. If you plan to mount nfs volumes than you need to leave this enabled. Since NFS is very insecure we are going to turn it off.
___ Comment out everything between "# This must be running in order to mount NFS" and
"#Done mounting remote (NFS) filesystems"

Disable samba filesystems
___ Comment out everything between "# Mount remote (SMB) filesystems" and
"# Done mounting remote (SMB) filesystems"

Turn of the NFS server
___ Comment out everything between "# Start the NFS server." and
"# Done starting the NFS server."
Turn of the BSD line spooler daemon
___ Comment out everything between "# Start the BSD line..." and
"# Done starting the BSD line..."

___ *:wq* - Done editing /etc/rc.d/rc.inet2

You can manually stop the services that run the above programs to get these settings to take affect or you can reboot now. In this example the system was rebooted.

Patches

There are a few things that need to be patched in Slackware 8 due to some known exploits. The first one is the /usr/man/cat* directories that we talked about before. The second is a flaw with 2.4.3+ kernels where the umask is left at 0000 after booting. Any files created during the boot process will be world writeable. This flaw doesn't affect this system since we are using the 2.2.19 kernel. If you decided to use the 2.4.5 kernel you will need to patch this, either by upgrading to the newer kernel 2.4.9 or by adding umask 022 to the top of all your bootup scripts. Details can be found at http://www.securityfocus.com/vdb/bottom.html?vid=3031.

Another security flaw exists in the GNU Locate program that would enable an attacker to run arbitrary code on the system. The attack requires the attacker to add malicious code to a locate database created with GNU locate prior to version 4. The system you have built will not have a database created with version 4.1 of locate, which should help reduce the risk of compromise. Most likely this is not going to be a big issue on your system, if you want to be cautious remove the locate program and the database files (*/usr/bin/locate*, */var/lib/locate/locatedb*). If you don't remove the files be sure to take note of what your tripwire reports show for the above two files. (Tripwire is an application that will be installed later in this paper) You can read about this flaw at http://www.securityfocus.com/vdb/bottom.html?vid=3127.

Yet another flaw is in the telnetd daemon that slackware uses. Since we're using ssh instead of telnet we're not going to worry about this one. You can if you like patch this bug by following the same procedure described below to install the new slackware sendmail package.

Mid paper and another security flaw is out. Keep in mind that you need to check the security lists or websites for new vulnerabilities if you want to maintain the security of your system. This vulnerability is a result of sendmail and should be patched before proceeding, if you installed sendmail. You will need remove it and install version 8.11.6.

http://www.securityfocus.com/bid/3163??
http://www.linuxsecurity.com/advisories/slackware_advisory-1573.html
http://www.packetstormsecurity.com/0108-exploits/alsou.c

Update: Okay, there are now more flaws in versions of sendmail up to and including 8.12.0. So the piece written below is still going to have security flaws. I suggest you leave sendmail off your system for right now since you will have vulnerabilities if you install it.

Logging

System logging is done through the syslogd daemon and kernel logging is done through the

klogd dameon. Both of these use the */etc/syslog.conf* configuration file. If you look at the current /etc/syslog.conf you'll see that warning messages are turned off. We'll turn them back on since we're interested in security. Also an entry will be added that will log reboots, failed logins, and su attempts.

___ *vi /etc/syslog.conf,* remove comment from before **.warn*
___ add a line *auth,authpriv.** */var/adm/ authlog* (Make sure you use tabs between the two fields otherwise it will fail.)

The files then need to be created and the permissions set. To do so complete the following.

___ *touch /var/log/authlog*
___ *chown root:sys /var/log/authlog*
___ *chmod 600 /var/log/authlog*
___ *kill -HUP <syslogd pid>* - Restart the syslogd daemon. Use *ps -ef|grep syslog* to find the daemon. The first number on the left is the pid.

There are three main files that will have logs, they are */var/log/syslog, /var/log/authlog, /var/log/messages*. Syslog contains all warning and error messages. On boot the kernel will log data in here mostly about the hardware. It will also contain information about failed logins, successfully and unsuccessfully su's. Authlog also contains the failed login attempts and failed, successful su's, ssh logins and user/group creation messages. /var/log/messages contains more bootup information pertaining to drivers and other devices.

The man pages for syslog.conf explain the facilities auth, authpriv, etc, the priorities, info, warn, etc and how you can use them to log other information you may be interested in (The man pages will have to be looked at on the compile machine). Remote logging is available and lets you log the data to another systems logs. This helps prevent a hacker from cleaning up their tracks before leaving the system. Remote logging wont be explained here since it requires another machine.

Log rotation is important to keep things organized and to allow for easy searching. The logs should be compressed to minimize space. It might also be required to transfer the logs off the computer in case of compromise. This can be done with scp. The script below just moves the compressed files into /var/log/archive. First create the directory for the compressed log files.

___ *touch /var/log/archive*
___ *chown root:root /var/log/archive*
___ *chmod 700 /var/log/archive*

Now create the script.

___ *vi /usr/sbin/logrotate*
Paste the following text:

```
#!/bin/sh
```

```
compress -f /var/log/messages
touch /var/log/messages
chown root:root /var/log/messages
chmod 600 /var/log/messages
mv /var/log/messages.Z /var/log/archive/messages.`date +%d%m%y`.Z

compress -f /var/log/syslog
touch /var/log/syslog
chown root:root /var/log/syslog
chmod 600 /var/log/syslog
mv /var/log/syslog.Z /var/log/archive/syslog.`date +%d%m%y`.Z

compress -f /var/log/authlog
touch /var/log/authlog
chown root:root /var/log/authlog
chmod 600 /var/log/authlog
mv /var/log/authlog.Z /var/log/archive/authlog.`date +%d%m%y`.Z

kill -1 `cat /var/run/syslogd.pid`
```

___ chmod 700 /usr/sbin/logrotate

Now the script needs to be added to crontab. Crontab is a job scheduling program that is installed on most *nix systems. The format of the file is shown in the first step. You can see what jobs are scheduled with crontab -l, currently nothing. Complete this step to create the job.

___ *crontab -e*

Add the following lines

```
#MIN    24HR    DAY OF      MONTH       DAY OF      COMMAND
#                MONTH                   WEEK
0       0       *       *   *       /usr/sbin/logrotate
```

The two comment lines just show how the cron file works, they can be added for reference. The third line is the actual command which is set to kickoff at 0 min, 0 hour, everyday or 12:00am everyday. Changes take effect immediately

Logging services are now setup and a rotation / archiving script has been put in place. Remember to monitor the logs for any anomalies. Compressed logs can be viewed without uncompressing by using *zcat <filename>*.

Network

The network components of the system will now be installed. The reason for bringing the network up now is that it allows more convenience in the latter sections. It does unfortunately point out a security issue since all of the security apps have not been installed yet. In the lab being used for this install it was possible to physically unplug from the rest of the network while completing the install. If this luxury is available, you should take advantage of it. Another way to accomplish the following parts without bringing the network online would be to run through all the steps and burn the final output to a cdrom or other media to be moved over.

© SANS Institute 2000 - 2005 Author retains full rights.

Configuring the Network

The first thing to do is to bring up the module for you network card.

___ *insmod <module>* - You'll need to find the name of your manufactures card and the appropriate module on your own. You can look in /lib/modules/2.2.19/net and see if anything strikes you as correct.
___ edit /etc/rc.d/rc.modules so that the module is started on bootup
___ add the line */sbin/modprobe* <module> somewhere after the dependencies section. The command to load the module is probably already in the file, if you search for the string and find it you can just uncomment the line.
___ edit /etc/rc.d/rc.inet1 to reflect you ip and settings

```
IPADDR="10.10.10.250" # REPLACE with YOUR IP address!
NETMASK="255.255.255.0" # REPLACE with YOUR netmask!
NETWORK="10.10.10.0"  # REPLACE with YOUR network address!
BROADCAST="10.10.10.255"    # REPLACE with YOUR broadcast address, if you
                                    # have one. If not, leave blank and edit below.
GATEWAY="10.10.245.254"   # REPLACE with YOUR gateway address!
```

___ edit /etc/resolv.conf to have your domain in the search order and a *nameserver* option with you nameserver.

```
search testing.net
nameserver 256.100.100.2
```

___ edit /etc/HOSTNAME to reflect your computers hostname

```
secure.testing.net
```

___ edit /etc/hosts to reflect you computers hostname

```
127.0.0.1       localhost
127.0.0.1       secure.testing.net secure
10.10.10.250 secure.testing.net secure  (I haven't had any problems with giving
127.0.0.1 the same name as the machine like the message at the top says.)
```

___ Plug in the network cable
___ run */etc/rc.d/rc.inet1*

Your network should now be up and running. Ping a few sites to make sure everything is okay. You can use *ifconfig* to check that the card is up. If you reboot now the ssh port (22) on your system will be open. You should read on and make a few changes to ssh before you reboot to be on the safe side.

Installing Other Applications

For the following options I am going to explain how to install some of the packages by compiling the binaries on a computer with the compile tools installed (gcc, cc, make, etc). The reason for compiling the applications on a different computer is that you avoid putting the compilers on the production system creating extra steps for an attacker. For example, most attackers will try to compile their rootkits on the compromised system to achieve a higher level of compatibility, without compilers the attacker will not be able to compile the rootkit on the system. There may be situations where you want to have the compilers on the system, remember that they can be used against you if they are there. For this example we will do without the compilers. Another

option would be to install the compilers in the original slackware install and remove them with pkgtool after the system is finished.

OpenSSH

Since the box is now network ready a secure channel will need to be created so files can be copied to the machine and remote administration is possible. The defacto standard today for creating this secure channel is SSH. In this case OpenSSH will be used, which is a GPL'd version of the ssh protocol. OpenSSH can be obtained at www.openssh.com.

Since OpenSSH and OpenSSL were installed during the setup process compiling and moving the binaries is not needed. At the time of writing both programs were the newest versions, the installation of ssh will not be discussed here but it's use will. Also OpenSSH comes compiled with the --with-tcp-wrappers option on. Tcp-Wrappers allows for control of the connection by it's ip or domain. You can check what options are used to build a application by checking the build file in /pub/mirrors/slackware/slackware-8.0/source/ on a ftp server the source directory on the cd.

When SSH installs it creates public and private host keys. These keys are used to create the initial encryption channel so if you are using password authentication the password is still encrypted when it goes over the wire.

In all honesty the way we are about to do these next steps may be a little overly paranoid. You can, if you choose to, create the keypair first and use scp to move the public key to the secured box. After that is completed password authentication can be turned off. This will save you the hassle of pulling out one of those archaic floppy disks.

First we need to edit the sshd_config file which is located in /etc/ssh/. By default ssh comes with password authentication allowed. We are going to turn it off in favor of PKI. Root login access through ssh sessions will also be turned off. This will require a user to login to the system first before getting into root.

___ *vi /etc/ssh/sshd_config*
    ___ change *PasswordAuthentication* to *no*
    ___ change *PermitRootLogin* to *no*

Now the system will accept only PKI authentication. Keypairs need to be generated to support PKI. The following steps should be completed on the system that will be remotely connecting into the secured box.

___ Log in to the system as a user. Do not use root.
___ type *ssh-keygen -t rsa*

___ Select file or press enter (usually enter)
___ Enter your passphrase twice - Don't forget this. You will need it when you try to connect to the remote box.

Your Public/Private keypair has now been created. They will be located in the path you specified above and be called either id_rsa/id_rsa.pub or id_dsa/id_dsa.pub depending on what encryption you choose to use.

To log into the secure machine we are currently building you need to copy over your public key from the remote system. It should be noted that when you attempt to ssh into the system it will look for the public key in <userid>/.ssh/. If the same username is used on both system then *ssh <ipaddr>* will work. If the userid's are different then ssh will need to be used with the userid in the command like this *ssh <userid>@<ipaddr>*.

For example, lets say this new system is named **secured** and your other system is named **remote**.

___ *mkdir ~/.ssh* - Do this on **secured** as a user, not root.
___ Copy ~/.ssh/id_rsa.pub from **remote** into ~/.ssh/remote.pub on **secured**.
You'll want to change the name for two reasons. First if you create a public/private key pair on the secured machine you run the risk of overwriting the public key for the remote system. Secondly, you will probably want to be able to distinguish which public key is which. This becomes more of an issue when you are running large environments using PKI with OpenSSH. You can also delete the public keys after you complete the next step.
___ *cat remote.pub >> authorized_keys2* (**secured**)
This will add the public key to the authorized keys file on the secured system. Doing this enables the host with the matching private key to connect. SSH only looks in this file for keys not in the .pub files.
___ *cat id_rsa >> identity* (**remote**)
This step is optional if you used the default name for the rsa or dsa key type. The default ssh_config file specifies identity, id_rsa, & id_dsa as identity files. If you gave the keys a different name then replace the id_rsa with the private key file name.

You should now be ready to SSH into the system.

___ *ssh <username>@<ip>* (**remote**) If the usernames are the same on both boxes as talked about above, username can be eliminated. Your first login will require you to accept the host key of the remote server. This should only happen the first time you connect to the box (for each user).
The authenticity of host '10.10.10.250 (10.10.10.250)' can't be established.
RSA key fingerprint is <fingerprint:aa:vv:bb:cc>
Are you sure you want to continue connecting (yes/no)?
___ type yes
___ enter your passphrase

You should now be connected to your secured machine. SSH connects can be seen in the authlog.

A few other notes about SSH. SSH can handle X11 Forwarding and portforwarding which make it a perfect secure remote admin tool on *nix systems. Portforwarding works really well as a poor man's VPN. Both of these can be controlled in the /etc/ssh/sshd_config.

TCP Wrappers

TCP Wrappers allows you to limit access to the system to specific subnets, domains, or IP's. The application comes compiled as part of the tcpip1 package. Since OpenSSH already has TCP Wrapper support compiled in our job has become much easier. Here we will set up TCP wrappers to check for incoming ssh connections. The files that control TCP Wrappers behavior are stored in /etc. The files are: hosts.allow and hosts.deny

___ By default you'll probably want to deny all connections. This can be done by adding the following line to */etc/hosts.deny*

ALL: ALL

___ Now access will need to be allowed for clients that should be allowed into the system. For the current system access should only be allowed for ssh clients on certain subnets. This can be done by adding the following to */etc/hosts.allow*
sshd: 10.1.1. 192.168.10.

Changes take effect immediately.

If you are running other services that support tcpwrappers you can add them here as well. The server name will be what is listed in the inetd.conf. If the daemon doesn't run from inetd.conf like sshd you'll need to check the documentation (usually it's the servers name or binary file). You can check whether your hosts.allow and hosts.deny are set up correctly by running *tcpdchk*. It will check that all services listed in those files match with ones in the inetd.conf file. A error is generated on applications that use tcp wrappers but are not listed in inetd.conf, like our install of ssh here. The following error will occur with the setup described above.

warning: /etc/hosts.allow, line 12: sshd: no such process name in /etc/inetd.conf

It's okay to get this warning.

When installing a server daemon remember to check the documentation to see if any compile options are need to enable tcp wrappers.

IpChains

IpChains is the firewall software built into the 2.2.19 kernel. You can use ipchains to build a personal firewall on the system. In appendix A is the ipchains script that will be used to tighten

your system. You may even want to remove lines from this script if you have no use for them. I scp all my data from a box that is connected to the network so to have ftp or even browsing abilities on my hardened box is useless. As always ymmv. I've tried to save some space by making the examples text small (8pt).

___ To use the script on your system.

___ *vi /etc/rc.d/rc.firewall*

___ paste all the script into the file, edit the top sections that need your input:

Example:

```
IPCHAINS=/sbin/ipchains
IF=eth0
IP=10.10.10.200/32
INT_NET=10.10.10.0/24
WORLD=0.0.0.0/0
NS=255.123.123.123/32
```

___ Save the file.

___ *chown root:root /etc/rc.d/rc.firewall*

___ c*hmod 700 /etc/rc.d/rc.firewall*

___ */etc/rc.d/rc.firewall*

___ *vi /etc/rc.d/rc.inet2*

To make the firewall start at boot add the following lines.

```
# Start the ipchains firewall:
if [ -x /etc/rc.d/rc.firewall ]; then
 echo "Starting ipchains based firewall /etc/rc.d/rc.firewall"
  /etc/rc.d/rc.firewall
fi
# Done starting the ipchains firewall.
```

The firewall script is fairly straightforward. The first thing it does is clear your rulesets (you can see what they are with *ipchains -nvL*). Then the systems allows icmp (you may want to disable this or limit it to the local network if you don't want outsiders to see your system) and it allows localhost traffic which you need. Next it opens the various applications for communication between this system (*$IP*) and the remote systems (either *$WORLD, $NS, or $INT_NET*). There are two rules for each with the exception of ftp (ssh has 4 rules because it can accept or initiate connections) The first rule is usually the input rule, which tells the system what it should do if someone tries to connect. The second rule tells the system what to do if a packet tries to leave the system. Since most connections are two way we need both rules to maintain a useful connection. FTP is an exception because it uses one port to open the connection between systems and then transfers the data on a different port. We define these with ftp and ftp-data.

The last rules log denied packets to */var/log/messages*. If an nmap scan is run in the testing system section of this document a lot of data will be generated in this log. It's good to know what the log entries look like in case they are seen in the future.

When you add a new server such as a web server, sendmail or dns you will need to add rules to the rc.firewall script. For example if we wanted to make this system a web server you would

need to add the following rules:

$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $WORLD -d $IP http
$IPCHAINS -A output -j ACCEPT -o $IF -p tcp -s $IP http -d $WORLD

The first rule allows for requests to come in on port 80 and the second allows the system to output data on port 80.

Tripwire

Tripwire is a host based intrusion detection system. It allows you to make a database that contains information about your filesystems. Based on that information reports can be run that will tell you if files or directories have been modified.

When you do the *make release* tripwire will compile with it's libraries static linked so no changes have to made. Also interestingly enough, it appears that the files in the tripwire source directory get their sgid bit turned on. It's not a big issue since the group id is set to *users*, but to avoid problems with this delete the tripwire source dir as soon as it gets installed. The tripwire binaries do not get sgid when they are installed.

The default policy file that comes with tripwire is set up for RedHat 7.1 full install. Obviously in our scenario this will generate a lot of useless warnings. Creating a new policy file is actually quite a tedious task, it requires going through the file system and adding each file to the policy file and then assigning rules for groupings of files and how they should be handled by tripwire. I've already done most of the legwork for you and edited the default file to work better with this install. It's located below in appendix A. You'll notice that it still generates some warnings. This is because there are some files that we'd like to know about if they have been created but we don't have them by default.

The following steps should be done on the compile machine:
___ Download tripwire-2.3.1-2.tar.gz from sourceforge
___ unpack the archive (*tar -zxf tripwire-2.3.1-2.tar.gz*)
___ *cd tripwire-2.3.1-2/src*
___ *make release* -This will take a while.
___ *cd ../..*
___ Copy the tripwire-2.3.1-2 directory to the new machine
scp -rp tripwire-2.3.1-2/ <username>@<ip>:/home/<username>/

The following steps should be done on the new machine:
___ *cd ~/tripwire-2.3.1-2*
___ *cp install/install.* .*
___ *vi install.cfg*
___ type */bin\/vi* and change that line to /usr/bin/vi
___ Modify email support settings

The default tripwire install will install with sendmail support rather than SMTP. You'll need to change the line *TWMAILMETHOD=SENDMAIL* to *TWMAILMETHOD=SMTP*. *Y*ou can just comment out the first instance of the line and uncomment the second instance. Uncomment the host and port lines as well and modify the to reflect your smtp server

___ *vi install.sh*

For some reason when *sh install.sh* is run it fails when looking for the binaries because it is searching in / rather than in the bin dir of the current working directory. To fix this I made the following changes:

___ Find the line BASE_DIR='basename $0' and change it to
BASE_DIR="<cwd>" - ex. BASE_DIR="/root/tripwire-2.3.1-2/"
Notice the change in quotes.

___ Comment out (#) the next line BASE_DIR=`echo $0 | sed s/$BASE_DIR\$//'

___ sh install.sh

___ Enter to view license agreement, read or *q*, type *accept* at the prompt.

___ *y* to install

You'll get errors that the system can't install the man pages and some docs. This is ok on this system since we've removed them. If you choose to leave the man pages and docs you shouldn't get any errors.

___ Enter a Site key passphrase twice

___ Enter a local key passphrase twice

___ Enter the site key passphrase twice more to finish the install. This will sign the configuration and policy files.

___ copy the new twpol.txt file to */etc/tripwire/* on the new system (see appendix a)

___ change the hostname in the twpol.txt file.

___ sign the policy file with *twadmin -m P -c /etc/tripwire/tw.cfg twpol.txt*

___ enter the site passphrase

Once we have the new policy file installed and signed the next thing to do is to initialize the database. This is done by typing *tripwire -m i*. You will most likely get a message that tripwire cannot find /usr/sbin/tw.cfg. This can be changed in the code before compiling but in our case we will use one of 2 shortcuts. The easy one (and probably more secure) is to use *tripwire -m i -c /etc/tripwire/tw.cfg*. The second is to create symlink from /usr/sbin/tw.cfg to /etc/tripwire/tw.cfg. However, if the system was compromised the attacker could replace the symlink with a file and cause your tripwire setup to fail or worse.

___ Run *tripwire -m i -c /etc/tripwire/tw.cfg*

___ Enter your local passphrase - The database will be generated and encrypted during this process. You will get the file not found errors that were discussed before.

To later run a report against that database you can use:

tripwire -m c -c /etc/tripwire/tw.cfg to check the system for discrepancies and

twprint -m r -r /var/lib/tripwire/report/<reportname> -c /etc/tripwire/tw.cfg
to print the report out

A good test of the system, since you've already initialized the database, is to step through the rest of the install process and then run a report to see what files have changed. After you get a chance to look at the report using the above command you'll need to update the tripwire database to reflect the changes. This is done with the command:

tripwire -m u -c /etc/tripwire/tw.cfg -d /var/lib/tripwire/<hostname.twd> -r \ /var/lib/tripwire/report/<reportname>

Don't forget that you have tripwire installed. It's a great application to tell you if anything on your system has been modified, but it's not very useful if you don't run reports and monitor the data.

Snort

Snort is a "lightweight network intrusion detection system". It analyses the packets that transfer to and from your box for data that looks like it may be part of a attack. It does this by using a ruleset that you can customize. There are places on the web that you can get signature files from such as www.snort.org or www.whitehats.com/ids/index.html. In this example the vision18.conf from the second link is used.

All of the following should be done on your compile system. First we'll need to get the source code from www.snort.org. As of this writing the newest version of snort is 1.8.1. You should also get yourself a copy of the vision18.conf file that we just talked about.

Next we'll run through the compile steps:
___ *tar -zxvf snort-1.8.1-RELEASE.tar.gz*
___ *cd snort-1.8.1-RELEASE*
___ *./configure --without-mysql* - (my compile machine had mysql and when I copied snort to the new box it would fail because of a missing library)
___ *make*
___ *scp -p snort <username>@<ip>:/home/<username>* - we won't copy over the man pages here since the man program and directories have been deleted.
___ download a copy of the vision18.conf file to */etc* on the secure box. You should get the one that is made for snort 1.8. Don't forget to un-gzip the file.
Everything else should be completed on the secure box.

First create a user to change snort to when it's running. This way if there is a compromise in snort the intruder will not get root access.
___ *groupadd snort*
___ *useradd -g snort -s / snort*
___ *vi /etc/shadow* - change the ! in *snort* to *
___ As root - *mv /home/<username>/snort /usr/sbin/*

___ *chown root:root /usr/sbin/snort*
___ *chmod 700 /usr/sbin/snort*
___ *chown root:root /etc/vision18.conf*
___ *chmod 600 /etc/vision18.conf*
___ *mkdir /var/log/snort*
___ *chown root:snort /var/log/snort*
___ *chmod 730 /var/log/snort*
___ *vi /etc/vision18.conf*
change the any in *var INTERNAL any* to be your IP address for example
var INTERNAL 10.10.10.240/32

___ Try executing snort to make sure it's working. s*nort -c /etc/vision18.conf -u snort*
You will get some output as snort parses through the config and then it should stop with
-*> Snort! <*-
Version 1.8.1-RELEASE (Build 74)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)

Your snort setup should now be ready to go into daemon mode. I normally execute snort with the following command.

/usr/sbin/snort -c /etc/vision18.conf -d -D -u snort

You can add this to you *rc.firewall* or *rc.inet2* to start it on bootup. There is also a startup file lying around somewhere out there that you may be able to get working. The addition to rc.inet2 is shown here.

```
# Start the Snort Network IDS system:
if [ -x /usr/sbin/snort ]; then
 echo "Starting the Snort Network IDS system  /usr/sbin/snort"
  / usr/sbin/snort -c /etc/vision18.conf -d -D -u snort
fi
# Done starting the Snort Network IDS system.
```

Snort stores it's logs under /var/log/snort. You'll want to check there to see if there have been any intrusions. There are some nice scripts in the */contrib* directory of the snort source that can help you parse through the data and make sense out of it. You should remember to check the logs and update your signature files. No monitoring system is useful if you don't look at what it produces.

Network Time Protocol

The Network time protocol or NTP allows you to sync your clock with various time servers on the internet to ensure as accurate as possible clock time. It's important to have the correct clock time when looking for an intrusion because any incidents that happen in a snort log or syslog can

be checked against all the other logs more efficiently. Also in large environments you can check multiple systems faster when they all have accurate times. Another reason for a accurate clock would be if you intend to prosecute an attacker on your systems.

NTP was installed at boot time under the network option ntp4. The configuration has to be fixed and the service started at bootup to complete the installation. First three timeservers will need to be chosen for use. Public timeservers can be found at http://www.eecis.udel.edu/~mills/ntp/ along with other ntp information. Please read the rules of using the timeservers and send emails to the administrators, don't be a time leach. List the servers here for future reference. The IP address of the system will need to be used rather than the hostname since the restrict section does not accept hostnames.

1) _____
2) _____
3) _____

Once the servers have been chosen the ntp.conf file needs to be modified to reflect the changes.

___ vi /etc/ntp.conf

The first server listed is 127.127.1.0. This is a special value for the internal clock. The line immediately after sets the clocks stratum level, which tells how far out of sync the clock is. Since the internal clock should not be used unless the server is way out of sync that stratum level is set at 10. The servers that will be connected to will be stratum 2 or 3. Servers at stratum level 1 should only be used by secondary servers that are serving many other users and servers.

___ add the three servers after the fudge line as shown

```
server    127.127.1.0       # local clock
fudge     127.127.1.0 stratum 10

server  128.4.40.12         # louie.udel.edu
server  130.126.24.53       # ntp-0.cso.uiuc.edu
server  128.46.103.93       # gilbreth.ecn.purdue.edu
```

Now that the servers are defined some security issues need to be handled.

___ Un-Comment the last line which should read *#restrict default ignore.* This will set the default policy for any systems attempting to communicate with this server to ignore. Of course some things need to be able to communicate, like the servers we want to use and the system itself. To allow for this add the lines as follows.

```
restrict 128.4.40.12 nomodify noquery
restrict 130.126.24.53 nomodify noquery
restrict 128.46.103.93 nomodify noquery
restrict 127.0.0.1 nomodify
```

The first three lines are the external servers. Communication needs to be allowed to them but there's no reason for any of those servers to communicate with us. The forth line allows queries to be made on the box. Modifications aren't allow since theoretically a hacker could spoof their ip address to 127.0.0.1.

Sometimes ntp has problems syncing time if the two times are off by over 20 to 30 minutes. Before running *ntpd* run *ntpdate* to set the time against one of the servers.

___ *ntpdate <server name>*

Now with a accurate clock time *ntpd* can be started.

___ *ntpd*

Ntpd will take a while to sync with the external servers and start setting the time, be patient while it syncs. Restarting will just cause the system to start syncing from scratch again. You can check on the progress with *ntpq -p* or *ntpq -c as*.

Ntpq -p gives output about the servers, their name, stratum, access, offset, etc. The reach field tells you what type of access your allowed to the server. When connected that number should be at 377. When the system starts that number will be 1 and progressively work it's way up as the system syncs. Below is output of a system that has been up a while. Notice how the third server has a reach of 367, obviously not at full sync. Upon closer look the *when* field is 15, much less than the other three systems, indicating that the system probably lost connectivity to the server and had to restart the syncing process.

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)        16 l   21   64  377    0.000    0.000   0.015
*louie.udel.edu  ntp1.nss.udel.e  2 u   34   64  377  551.965   71.200  77.075
 ntp-0.cso.uiuc. 128.174.38.133   2 u   15   64  367  263.553   36.989  64.567
+gilbreth.ecn.pu humbug.antnet.c  3 u   26   64  377  321.774   21.005  38.906
```

Ntpq -c as gives similar output but is more of a brief status. It tells if the server has been configured successfully, whether it's reachable, if authentication is being used, condition, last event, etc. The most important field is condition. If the field says reject then the system will not communicate with that server. This is the default setting for the localhost (1) because that should only be used in a situation where other servers are not connected. Below is the output of a system that has already synced.

```
ind assID status  conf reach auth condition  last_event cnt
===========================================================
  1 14316  9034   yes   yes  none    reject   reachable  3
  2 14317  9634   yes   yes  none sys.peer   reachable  3
  3 14318  9434   yes   yes  none  candidat   reachable  3
  4 14319  9434   yes   yes  none  candidat   reachable  3
```

___ Once everything is working ntpd needs to be set to boot at startup. This can be done in the

© SANS Institute 2000 - 2005 Author retains full rights.

rc.inet2 file, add the following lines at the end of the file.

```
# NTP startup
if [ -x /usr/sbin/ntpd ]; then
echo "Starting NTPD:  /usr/sbin/ntpd"
/usr/sbin/ntpd
fi
# Done NTP startup
```

Ntpd will now start at bootup.

Chrooted Daemons

Running a daemon in a chrooted environment means that when the application is run it becomes bound to the directory you have it installed in. For example if you've installed your server in /usr/local/ftp when you run the daemon /usr/local/ftp looks like / to the daemon. This is especially useful because if the service crashes the chrooted environment remains. So an attacker might think they compromised the / directory when in fact they have just compromised /usr/local/ftp which if things are done correctly won't have anything of interest in it. The other advantage is that exploits based on malformed file requests will fail since a link to ../etc/passwd will still be trying to access /usr/local/ftp/etc/passwd.
For this example a chrooted bind environment will be installed.

To chroot a process you need to do one of two things. Either the application has to be statically linked, or it needs to have all the required libraries available to it within the directory that is used for the chrooted hierarchy, or the jail as it's called. So if the application needs /usr/lib/example.so then you will need to copy example.so to /usr/local/ftp/usr/lib/example.so. You can find out what files are needed by an application by using ldd <executable>.

To begin setting up the chrooted environment download the newest version of bind 8 from http://www.bind.com to you compile machine. Remember that we are using bind 8 to show how to set up a chrooted environment it is not required to complete the building of the system. Bind is not the only server you can run in a chrooted environment, ftp, www, etc can be run this way as well. The installation process for building servers are usually different so you may have to look up other documentation from the internet. The process below is quick and very vanilla. If you want more complete instructions please check one of the following two sites:

http://www.psionic.com/papers/dns/dns-linux
http://www.linuxdocs.org/HOWTOs/Chroot-BIND8-HOWTO.html

Step through the following instructions to build the BIND8 daemon on you compile system.

___ download the source code from www.bind.com
___ *tar -zxvf bind-src.tar.gz*
___ *cd src*
___ *vi port/linux/Makefile.set*

change *'CDEBUG=-O -g'* to *'CDEBUG=-O -static'*
change 'DESTRUN=/usr/local/named/var/run'

___ *vi bin/named/named.h*
Find the line #include "pathnames.h"
Add *#define _PATH_NDCSOCK "/var/run/ndc"* immediately after the above line

___ *make*

___ Confirm the system has compiled statically by running *ldd bin/named/named*. You should get a message saying "not a dynamic executable"

BIND should not run as root so create a userid and groupid for named. This should be done on the secure machine being built.

___ *groupadd named*
___ *useradd named -g named -s /*
___ *vi /etc/shadow*
Change the *!* In the named entry to *=LOCK=*. There's no reason to log into the account.

Now that the binaries are compiled the jail needs to be set up. We call the directory structure that will be chrooted to the jail because everything is contained inside. Here we will build the chrooted jail on the secure system and scp the appropriate files over when needed.

___ *mkdir /usr/local/named* - This is where the jail will be built in this example
___ *cd /usr/local/named*
___ *mkdir bin dev etc etc/named var var/run usr usr/sbin*
___ *mknod dev/null c 1 3*
___ *chmod 666 dev/null*
___ *chown named:named /usr/local/named/var/run*
___ *cp /etc/localtime /usr/local/named/etc/localtime*
___ *echo 'named:x:102:' > /usr/local/named/etc/group* - The *102* group must be the same number as in your /etc/group

Now you'll need to create you */usr/local/named/etc/named.conf* and */usr/local/named/etc/namedb/** files. Below are very basic examples.

```
--- named.conf---
options {
      directory "/etc/namedb";
      version "whatever";
      query-source address * port 53;
};

zone "example.net" {
      type master;
      file "example.net.zone";
};

zone "10.168.192.in-addr.arpa" {
```

```
        type master;
        file "10.168.192.in-addr.arpa.zone";
--- named.conf ---

--- example.net.zone---
@ IN SOA secure.example.net. Admin.example.net. (
            2001092202      ; Serial - year/month/date/revision
            86400           ; Refresh from server - daily
            3600            ; Retry after failure - hourly
            604800          ; Expire data- 7 days
            86400 )         ; Time to live - 1 day

@               IN      NS      secure.example.net.
secure          IN      A       192.168.10.1
thegate         IN      A       192.168.10.254
laptop          IN      A       192.168.10.250
--- example.net.zone ---

--- 10.168.192.in-addr.arpa.zone ---
@ IN SOA secure.example.net. Admin.example.net. (
            2001092202      ; Serial - year/month/date/revision
            86400           ; Refresh from server - daily
            3600            ; Retry after failure - hourly
            604800          ; Expire data- 7 days
            86400 )         ; Time to live - 1 day

@               IN      NS      secure.example.net.
1               IN      PTR     secure.example.net.
254             IN      PTR     thegate.example.net.
250             IN      PTR     laptop.example.net.
--- 10.168.192.in-addr.arpa.zone --
```

The binary files need to be moved to the new machine.
___ *scp <compile machine>:<??/src/bin/named/named> /usr/local/named/usr/sbin/*
___ *scp <compile machine>:<??/src/bin/named-xfer/named-xfer > /usr/local/named/usr/sbin/*

Since this application is being run in a chrooted jail it needs to be told how to log data from the dns server. Slackware starts the syslogd server during the rc.inet2 bootup (or in rc.M if rc.inet2 does not exist).

___ *vi /etc/rc.d/rc.inet2* - Find the line /usr/sbin/syslogd and replace it with
/usr/sbin/syslogd -m 0 -a /usr/local/named/dev/log. The line above this one can be edited to reflect the changes as well although it doesn't have to be.

Now when you restart the daemon, either reboot or kill syslogd and restart it with the above command, you should see a *log* file in the */usr/local/named/var* directory.

The server can now be executed with
/usr/local/named/usr/sbin/named -u named -g named -t /usr/local/named

The named server should be running, but unfortunately if you try and connect to the dns server

you won't be able to. You'll need to add the following three lines to the firewall script to open port 53 for incoming connections. Add them anywhere after the -F and -P lines, I usually put them with the other dns rules.

```
#Allow DNS Queries - Inbound
$IPCHAINS -A input -j ACCEPT -i $IF -p udp -s $INT_NET  -d $IP domain
$IPCHAINS -A output -j ACCEPT -i $IF -p udp -s $IP domain -d $INT_NET
```

Bind 8 does not have support for tcpwrappers.

Nslookup or dig should now be able to use the secure box as a DNS server. Run either one on the compile machine, do a *server <secure machine>,* and then test the system with some of your domains names. If everything works ok you can make the following changes to make the named daemon boot at startup.

___ Open */etc/rc.d/rc.inet2*
___ Find the section # # Start the BIND name server daemon:
___ Uncomment *if* to *fi* under option 2 and change the text to read as follows (adjust to your jail directory if necessary).

```
if [ -x /usr/local/named/usr/sbin/named ]; then
echo "Starting BIND:  /usr/sbin/named -u named -g named -t /usr/local/named"
/usr/local/named/usr/sbin/named -u named -g named -t /usr/local/named
fi
```

Sendmail

Earlier we talked about the security issue with the sendmail that comes with slackware 8.0, and then about how versions up to 8.12.0 have vulnerabilities. Here are the instructions for installing version 8.11.6 and configuring it for outgoing email. It is recommended that sendmail not be installed as a daemon to maintain the security of the system.

Slackware has a package manager called pkgtool that can be used to apply patches and upgrades to the system. Since there is a package for sendmail 8.11.6 on the slackware ftp server we can use this to patch the system a little easier. When a upgrade to a more secure version (> 8.12.0) comes out you should be able to use the same instructions. First download the package from
ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/sendmail.tgz
also, you'll need to download
ftp.slackware.com/pub/slackware/slackware-8.0/slakware/n1/smailcfg.tgz. Which should also be on your cdrom under slakware/n1/smailcfg.tgz. This program has not been updated so downloading a new one is optional.

___ *cd <direcotory where sendmail.tgz is>*
___ *pkgtool*
___ *Current*
___ You should be prompted to install the sendmail package. If there is another package in the same directory you will be prompted to say yes or no for it. Say Yes to install the

sendmail package.

Sendmail will now be installed or patched depending on whether you installed it earlier. As you can see the package tool can save you a lot of compile time if you have the luxury to wait for the slackware team to update the packages. Sendmail of course needs to be configured to be operational, something that is beyond the scope of this document. A short example will be given that may work for you. Documentation is given at the bottom for help configuring other sendmail setups.

Sendmail should not be run as a daemon. This way it will facilitate sending external email but will not be running as a daemon and opening vulnerabilities on the system. To prevent it from starting at boot complete the next steps.
___ *vi /etc/rc.d/rc.M*
___ Comment out the four lines after "*# Start the sendmail daemon:"*

If you have a SMTP server that handles relaying outbound email traffic you can setup the following options in */etc/mail/sendmail.cf* to have mail directed at it.

___ *cp /usr/src/sendmail/linux.smtp.cf /etc/mail/sendmail.cf*
___ Change the following fields to the correct settings in /etc/mail/sendmail.cf.
DSsmtp.example.net
DRsmtp.example.net
DHsmtp.example.net
DMexample.net

Pine by default should invoke sendmail to send the email to the smtp server. Other changes may be required to have your installation of sendmail working correctly. Documentation for the full configuration can be found at http://www.sendmail.org/ .

Re-Hardening the Operating System

Read-Only Filesystems

One thing that could not be done before was to make the hard drive partitions read-only. If we did we would have received a lot of errors when trying to modify or move files on the system before. So now that the system is at a good stage we can lock it down. As you'll see the drives that become read-only are /, /boot, and /usr. /usr/local remains read-write so servers can be installed. If you need to create symlinks or want to install other apps in any of the read-only partitions you'll need to remount it as read-write before you can do so. Before we make those changes lets discuss setuid and setgid files.

SetUID/GID Files

Some files on the system have the ability to change the user they are run as to the owner of the

files. These files are called suid or setuid files. Likewise files that change the group id are sgid or setgid files. We can find these files with the following 2 commands.
Files on the system w/ suid bit set - *find / -perm -4000)*

Files on the system w/ sgid bit set - *find / -perm -2000*)

As you can see from the example the suid bit is 4 and the sgid bit is 2. You create a suid file by using chmod 4555. When you ls -l the file you will see a "s" in the user or group execute bit.

On our system there are relatively few of these files. We should take note of them and take appropriate steps to secure them. There are a two good ways to deal with these files.

1) You can turn the setuid or guid bit off by chmod'ing the file with 555 or whatever the file is using without the leading 4
2) You can add nosuid to the /etc/fstab and the suid & sgid will be useless

The benefits of turning off the suid bit and sgid bit are that none of those programs can be run outside of root. You must be careful if you turn off suid on some programs though. For example, su has the suid bit set and if it is not working you will not be able to su into root.

Since most of the programs that we don't want to have suid are on the /usr partition we shall use option 2. The other benefit we can gain by using option two here is that an intruder cannot upload a file called *ls* with some malicious code and then set it to suid. If this was the case we may be tricked into running it. Also we can gain the ability to turn them back on quicker if we should need to. Now we can edit the */etc/fstab* and make both of the above changes in one step.

___ *vi /etc/fstab*
change the *default* field for */usr* and */boot* to *ro,nosuid*
change the *default* field for */home, /var,* and */usr/local* to *rw,nosuid*
change the *default* field for */* to *ro*

Changes will take effect when you reboot. Of course this does add some other issues, for instance why not try ssh'ing from this system to another. You'll get a message that says Operation not permitted. This is because ssh uses suid and since it's on a nosuid partition it gets denied. We can fix this be moving the application from /usr/bin to /bin, or you can use root to ssh out of the system as another user. SSH isn't really the app that we intented to disable, things like newgrp, crontab, rlogin, etc that might end up having a exploitable hole were the primary targets. Of course ssh might have an exploit some day too so if you don't need to ssh out of the system leave it where it is.

Testing the System

Now that you have the system built and sufficiently hardened some testing should be done to ensure that there aren't exploits left open.

Ps -ef will let you know what process are running. You should have a fairly small number of processes. Below is the output from my machine after done securing. You can see the initial processes started to boot the system: init, and the [k**]. System logging daemons klogd, and syslogd. The ssh server /usr/sbin/sshd. Crond runs scheduled jobs along with atd. Agetty 1-6 are your login terminals alt-f1 through f6, one isn't listed because it's already logged into. Bash is the shell that is logged on, you can see what terminal by the tty1 on the left. Lastly named is your DNS server. Some of the data will probably get truncated off the right side of the screen so if something looks like half a command you'll know why.

Basically you'll want to look here ever now and then to see that processes that you don't know about are running. If an attacker wants to run a keylogger or some other daemon that will capture data for them you'll probably see it in here first.

```
root@secure:/etc/rc.d# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Sep20 ?        00:00:05 init
root         2     1  0 Sep20 ?        00:00:00 [kflushd]
root         3     1  0 Sep20 ?        00:00:00 [kupdate]
root         4     1  0 Sep20 ?        00:00:00 [kswapd]
root         5     1  0 Sep20 ?        00:00:00 [keventd]
root        57     1  0 Sep20 ?        00:00:06 /usr/sbin/klogd -c 3
root        60     1  0 Sep20 ?        00:00:09 /usr/sbin/sshd
root        62     1  0 Sep20 ?        00:00:00 /usr/sbin/crond -l10
daemon      64     1  0 Sep20 ?        00:00:00 /usr/sbin/atd -b 15 -l 1
root       101     1  0 Sep20 tty2     00:00:00 /sbin/agetty 38400 tty2 linux
root       102     1  0 Sep20 tty3     00:00:00 /sbin/agetty 38400 tty3 linux
root       103     1  0 Sep20 tty4     00:00:00 /sbin/agetty 38400 tty4 linux
root       104     1  0 Sep20 tty5     00:00:00 /sbin/agetty 38400 tty5 linux
root       105     1  0 Sep20 tty6     00:00:00 /sbin/agetty 38400 tty6 linux
ltfiend    245     1  0 Sep24 tty1     00:00:00 -bash
root       255   245  0 Sep24 tty1     00:00:00 bash
root       671     1  0 Sep24 ?        00:00:00 syslogd -m 0 -a /usr/local/named
named      676     1  0 Sep24 ?        00:00:00 /usr/local/named/usr/sbin/named
root       734    60  0 08:25 ?        00:00:00 /usr/sbin/sshd
ltfiend    735   734  0 08:25 pts/0    00:00:00 -bash
root       751   735  0 08:40 pts/0    00:00:00 bash
root       754   751  0 08:43 pts/0    00:00:00 ps -ef
```

Netstat

Netstat will provide a listing of all connections on this server. You will see your syslogd and klogd daemons running under there. If you do a netstat -p you will get info about network connections as well as the ports they connected to. Below is my output. Notice the ssh session listed.

```
root@secure:/etc/rc.d# netstat -p
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
```

```
tcp        0      0 secure.testing.net:ssh  10.4.2.90:34460         ESTABLISHED 734/sshd
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node PID/Program name    Path
unix  1      [ ]         DGRAM                    710    671/syslogd         /usr/local/named/dev/log
unix  0      [ ]         DGRAM                    708    671/syslogd         /dev/log
unix  0      [ ]         DGRAM                    714    676/named
unix  0      [ ]         DGRAM                    37     57/klogd
```

This data is useful if you see a network connection acting up when you know it should be quiet. This will give you a quick listing of what might be going on.

Nmap

Nmap is a port scanning utility, it can tell you whether the machine has any ports that we inadvertently left open during the hardening process. You'll need to get nmap from www.nmap.org. This document isn't going to go into the installing of nmap. It is fairly simple and you can probably get binaries as well as the source. Nmap can run on the secure machine and scan local ports but the proper way to use it is to install on the compile machine and scan the ports across the network. Below is a scan of my tcp ports and you can see that ssh is the only thing allowed in (sendmail was turned of at this point, if you have installed it and configured it as a daemon you will see export 25 open as well)

```
[root@test]# nmap -sS -O 10.4.2.92

Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
Interesting ports on  (10.4.2.92):
(The 1533 ports scanned but not shown below are in state: filtered)
Port       State      Service
22/tcp     open       ssh

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3329454 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.16

Nmap run completed -- 1 IP address (1 host up) scanned in 181 seconds
```

A scan with the *-sU* (UDP) option would show that port 53 is open for the dns server.

Any ports that are open are a risk and should be understood or turned off. If you have a port open that shouldn't be you'll have to use either *ps* -ef or *netstat* to figure out what it is. Then you can stop the daemon wherever it resides. Both of our open ports are being used for daemons we want running and we have properly secured them

Other things you should check on.

___ SSH is functional
___ cannot execute setuid files from /boot, /home, /var, /usr, /usr/local
___ cannot write to /, /usr, /boot

___ cannot mount or export NFS volumes
___ logs have data
___ Run a Nessus scan. You can get the software from www.nessus.org Documentation can be found there as well. Nessus performs vulnerability analysis on the system by scanning the system with know exploits. Be careful when using this cause they will crash a unsecured machine on the network
___ Physical Security - Is the device in a location that you can guarantee the physical security? If not have you taken appropriate steps to prevent against tampering? BIOS Passwords, LILO Password (mentioned above), Cable lockdown (If you fear hardware being stolen), and keyboard locking all help prevent access from an attacker that is in front of the machine. It becomes extremely difficult to maintain the internal security of a machine if you can't keep the machine physically secured, so take note of the area where the machine will finally reside and make changes appropriately.

Final Remarks

You should now have a fairly secure Slackware 8.0 install. Your system has host based intrusion detection with tripwire, network based ids with snort and a firewall using ipchains. You have tightened down access on your filesystems and incoming connections. So what's next?

If you intend to keep the system secure you'll need to start reading up on info from some sites around the Internet. Here are a few of my favorites to get you started:

www.linuxsecurity.com
www.securityfocus.com
www.packetstormsecurity.com
www.whitehats.com

There are also quite a few mailing lists out there that you may want to subscribe to. Keep in mind that new vulnerabilities come out daily and there are no completely secure systems.

This paper walked through getting your base Slackware setup up and running. It is up to you to now install the server(s) that are needed for your environment. As you proceed to put services on make sure that you read up on possible vulnerabilities and how to protect against them. Wherever possible run your daemons in chrooted environments and use TCP Wrappers where available. Remember to monitor and update tripwire with changes. Good Luck!

This page intentionally left blank.

Appendix A - Example Files

---rc.firewall---begin---

```
# Ipchains Firewaill Script

IPCHAINS=/sbin/ipchains
IF=eth0
IP=<PUT YOUR IP HERE!>
INT_NET=<PUT YOU NETWORK HERE>
WORLD=0.0.0.0/0
NS=<PUT YOU DNS HERE>
# EXT_SERVER1=<if you use option B in ssh set this.  Otherwise delete it>

# Flush and set policy
$IPCHAINS -F input
$IPCHAINS -F output
$IPCHAINS -F forward
$IPCHAINS -P input DENY
$IPCHAINS -P output DENY
$IPCHAINS -P forward DENY

# Allow ICMP and localhost traffic
$IPCHAINS -A input -j ACCEPT -i $IF -p icmp
$IPCHAINS -A output -j ACCEPT -i $IF -p icmp
$IPCHAINS -A input -j ACCEPT -i lo -p all
$IPCHAINS -A output -j ACCEPT -i lo -p all

# deny w/o logging all broadcast and other non-directed traffic
$IPCHAINS -A input -i $IF -d ! $IP -j DENY

# Allow ssh to the secure system from internal  systems
$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $INT_NET -d $IP ssh
$IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP ssh -d $INT_NET

#Allow ssh out from the secure machine
$IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP -d $INT_NET ssh
$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $INT_NET ssh -d $IP

# Allow ssh to the secure system from external systems.

###
# A) From any external systems
###
#  $IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $WORLD -d $IP ssh
#  $IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP ssh -d $WORLD
###

###
# B) From specific external systems
###
#  $IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $EXT_SERVER1 -d $IP ssh
#  $IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP ssh -d $EXT_SERVER1

# Allow web surfing
$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $WORLD http -d $IP
$IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP -d $WORLD http

# Allow anonymous FTP
$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $IP -d $WORLD ftp
$IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $WORLD ftp -d $IP
$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $WORLD ftp-data -d $IP
$IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP -d $WORLD ftp-data

# Allow DNS Queries to external server
$IPCHAINS -A input -j ACCEPT -i $IF -p udp -s $NS domain -d $IP
$IPCHAINS -A output -j ACCEPT -i $IF -p udp -s $IP -d $NS domain
```

```
# Allow NTP
$IPCHAINS -A input -j ACCEPT -i $IF -p udp -s $WORLD ntp -d $IP
$IPCHAINS -A output -j ACCEPT -i $IF -p udp -s $IP -d $WORLD ntp

# Allow SMTP
$IPCHAINS -A input -j ACCEPT -i $IF -p tcp -s $WORLD smtp -d $IP
$IPCHAINS -A output -j ACCEPT -i $IF -p tcp -s $IP -d $WORLD smtp


# Log all other traffic
$IPCHAINS -A input -j DENY -l
$IPCHAINS -A output -j DENY -l
```

---rc.firewall---end---

--- twpol.txt --- begin ---

```
 ##############################################################################
 #                                            ##
############################################################################### #
#                                            # #
#                Policy file for Slackware Linux 8.0                  # #
#                        V1.0.0                          # #
#                     August 29, 2001                          # #
#                                            ##
# modifed by peter@devries.tv                 ##
###############################################################################


 ##############################################################################
 #                                            ##
############################################################################### #
#                                            # #
# This is the example Tripwire Policy file.  It is intended as a place to   # #
# start creating your own custom Tripwire Policy file.  Referring to it as  # #
# well as the Tripwire Policy Guide should give you enough information to   # #
# make a good custom Tripwire Policy file that better covers your           # #
# configuration and security needs.  A text version of this policy file is  # #
# called twpol.txt.                                  # #
#                                            # #
# The example policy file is best run with 'Loose Directory Checking'       # #
# enabled. Set LOOSEDIRECTORYCHECKING=TRUE in the Tripwire Configuration    # #
# file.                                       # #
#                                            # #
# Email support is not included and must be added to this file.             # #
# Add the 'emailto=' to the rule directive section of each rule (add a comma # #
# after the 'severity=' line and add an 'emailto=' and include the email    # #
# addresses you want the violation reports to go to).  Addresses are        # #
# semi-colon delimited.                                  # #
#                                            ##
###############################################################################



 ##############################################################################
 #                                            ##
############################################################################### #
#                                            # #
# Global Variable Definitions                                  # #
#                                            # #
# These are defined at install time by the installation script.  You may    # #
# Manually edit these if you are using this file directly and not from the  # #
# installation script itself.                                  # #
#                                            ##
###############################################################################

@@section GLOBAL
TWDOCS="/usr/doc/tripwire";
TWBIN="/usr/sbin";
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
TWPOL="/etc/tripwire";
TWDB="/var/lib/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME=darkstar;

@@section FS
SEC_CRIT      = $(IgnoreNone)-SHa ;  # Critical files that cannot change
SEC_SUID      = $(IgnoreNone)-SHa ;  # Binaries with the SUID or SGID flags set
SEC_BIN       = $(ReadOnly) ;        # Binaries that should not change
SEC_CONFIG    = $(Dynamic) ;         # Config files that are changed infrequently but accessed often
SEC_LOG       = $(Growing) ;         # Files that grow, but that should never change ownership
SEC_INVARIANT = +tpug ;              # Directories that should never change permission or ownership
SIG_LOW       = 33 ;                 # Non-critical files that are of minimal security impact
SIG_MED       = 66 ;                 # Non-critical files that are of significant security impact
SIG_HI        = 100 ;                # Critical files that are significant points of vulnerability
# Tripwire Binaries
(
 rulename = "Tripwire Binaries",
 severity = $(SIG_HI)
)
{
 $(TWBIN)/siggen                  -> $(SEC_BIN) ;
 $(TWBIN)/tripwire                -> $(SEC_BIN) ;
 $(TWBIN)/twadmin                 -> $(SEC_BIN) ;
 $(TWBIN)/twprint                 -> $(SEC_BIN) ;
}

# Tripwire Data Files - Configuration Files, Policy Files, Keys, Reports, Databases
(
 rulename = "Tripwire Data Files",
 severity = $(SIG_HI)
)
{
 # NOTE: We remove the inode attribute because when Tripwire creates a backup,
 # it does so by renaming the old file and creating a new one (which will
 # have a new inode number).  Inode is left turned on for keys, which shouldn't
 # ever change.

 # NOTE: The first integrity check triggers this rule and each integrity check
 # afterward triggers this rule until a database update is run, since the
 # database file does not exist before that point.

 $(TWDB)                          -> $(SEC_CONFIG) -i ;
 $(TWPOL)/tw.pol                  -> $(SEC_BIN) -i ;
 $(TWPOL)/tw.cfg                  -> $(SEC_BIN) -i ;
 $(TWLKEY)/$(HOSTNAME)-local.key  -> $(SEC_BIN) ;
 $(TWSKEY)/site.key               -> $(SEC_BIN) ;

 #don't scan the individual reports
 $(TWREPORT)                      -> $(SEC_CONFIG) (recurse=0) ;
}


# Tripwire HQ Connector Binaries
#(
#  rulename = "Tripwire HQ Connector Binaries",
#  severity = $(SIG_HI)
#)
#{
#  $(TWBIN)/hqagent                 -> $(SEC_BIN) ;
#}
#
# Tripwire HQ Connector - Configuration Files, Keys, and Logs

 ##############################################################################
```

```
 #                                                ##
################################################################################ #
#                                                # #
# Note: File locations here are different than in a stock HQ Connector       # #
# installation.  This is because Tripwire 2.3 uses a different path         # #
# structure than Tripwire 2.2.1.                                            # #
#                                                # #
# You may need to update your HQ Agent configuation file (or this policy     # #
# file) to correct the paths.  We have attempted to support the FHS standard # #
# here by placing the HQ Agent files similarly to the way Tripwire 2.3       # #
# places them.                                                              # #
#                                                ##
################################################################################

#(
#  rulename = "Tripwire HQ Connector Data Files",
#  severity = $(SIG_HI)
#)
#{
#   ###############################################################################
#  ###############################################################################
#  # NOTE: Removing the inode attribute because when Tripwire creates a backup ##
#  # it does so by renaming the old file and creating a new one (which will    ##
#  # have a new inode number).  Leaving inode turned on for keys, which        ##
#  # shouldn't ever change.                                                    ##
#  ###############################################################################
#
#  $(TWBIN)/agent.cfg                 -> $(SEC_BIN) -i ;
#  $(TWLKEY)/authentication.key       -> $(SEC_BIN) ;
#  $(TWDB)/tasks.dat                  -> $(SEC_CONFIG) ;
#  $(TWDB)/schedule.dat               -> $(SEC_CONFIG) ;
#
#  # Uncomment if you have agent logging enabled.
#  #/var/log/tripwire/agent.log      -> $(SEC_LOG) ;
#}



# Commonly accessed directories that should remain static with regards to owner and group
(
  rulename = "Invariant Directories",
  severity = $(SIG_MED)
)
{
  /                          -> $(SEC_INVARIANT) (recurse = 0) ;
  /home                      -> $(SEC_INVARIANT) (recurse = 0) ;
  /etc                       -> $(SEC_INVARIANT) (recurse = 0) ;
}
  ################################################
 #                                ##
################################################ #
#                                # #
# File System and Disk Administration Programs # #
#                                ##
################################################

(
  rulename = "File System and Disk Administraton Programs",
  severity = $(SIG_HI)
)
{
  # /sbin/accton                  -> $(SEC_CRIT) ;
  /sbin/badblocks                -> $(SEC_CRIT) ;
  /sbin/dosfsck                  -> $(SEC_CRIT) ;
  /sbin/e2fsck                   -> $(SEC_CRIT) ;
  /sbin/debugfs                  -> $(SEC_CRIT) ;
  /sbin/dumpe2fs                 -> $(SEC_CRIT) ;
```

```
 /sbin/e2label                -> $(SEC_CRIT) ;
 /sbin/fdisk                  -> $(SEC_CRIT) ;
 /sbin/fsck                   -> $(SEC_CRIT) ;
 /sbin/fsck.umsdos            -> $(SEC_CRIT) ;
 /sbin/fsck.ext2              -> $(SEC_CRIT) ;
 /sbin/fsck.ext3              -> $(SEC_CRIT) ;
 /sbin/fsck.minix             -> $(SEC_CRIT) ;
 /sbin/fsck.msdos             -> $(SEC_CRIT) ;
 /sbin/fsck.hpfs              -> $(SEC_CRIT) ;
 /sbin/fsck.reiserfs          -> $(SEC_CRIT) ;
 /usr/sbin/hdparm             -> $(SEC_CRIT) ;
 /sbin/mkdosfs                -> $(SEC_CRIT) ;
 /sbin/mke2fs                 -> $(SEC_CRIT) ;
 /sbin/mkfs                   -> $(SEC_CRIT) ;
 /sbin/mkfs.ext2              -> $(SEC_CRIT) ;
 /sbin/mkfs.minix             -> $(SEC_CRIT) ;
 /sbin/mkfs.msdos             -> $(SEC_CRIT) ;
 /sbin/mkswap                 -> $(SEC_CRIT) ;
 /sbin/resize2fs              -> $(SEC_CRIT) ;
 /usr/sbin/sfdisk             -> $(SEC_CRIT) ;
 /sbin/tune2fs                -> $(SEC_CRIT) ;
 /sbin/update                 -> $(SEC_CRIT) ;
 /bin/mount                   -> $(SEC_CRIT) ;
 /bin/umount                  -> $(SEC_CRIT) ;
 /bin/touch                   -> $(SEC_CRIT) ;
 /bin/mkdir                   -> $(SEC_CRIT) ;
 /bin/mknod                   -> $(SEC_CRIT) ;
 /usr/bin/mktemp              -> $(SEC_CRIT) ;
 /bin/rm                      -> $(SEC_CRIT) ;
 /bin/rmdir                   -> $(SEC_CRIT) ;
 /bin/chgrp                   -> $(SEC_CRIT) ;
 /bin/chmod                   -> $(SEC_CRIT) ;
 /bin/chown                   -> $(SEC_CRIT) ;
 /bin/cp                      -> $(SEC_CRIT) ;
}

 ###################################
 #                                 ##
################################### #
#                                 # #
# Kernel Administration Programs # #
#                                 ##
###################################

(
 rulename = "Kernel Administration Programs",
 severity = $(SIG_HI)
)
{
 /sbin/depmod                 -> $(SEC_CRIT) ;
 /usr/sbin/ctrlaltdel         -> $(SEC_CRIT) ;
 /sbin/insmod                 -> $(SEC_CRIT) ;
 /sbin/insmod_ksymoops_clean  -> $(SEC_CRIT) ;
 /usr/sbin/klogd              -> $(SEC_CRIT) ;
 /sbin/ldconfig               -> $(SEC_CRIT) ;
 /sbin/modinfo                -> $(SEC_CRIT) ;
}

 ########################
 #                ##
######################## #
#                # #
# Networking Programs # #
#                ##
########################

(
```

```
 rulename = "Networking Programs",
 severity = $(SIG_HI)
)
{
 /sbin/arp                      -> $(SEC_CRIT) ;
 /sbin/ifconfig                   -> $(SEC_CRIT) ;
 /sbin/ipchains                   -> $(SEC_CRIT) ;
 /sbin/ipchains-restore             -> $(SEC_CRIT) ;
 /sbin/ipchains-save              -> $(SEC_CRIT) ;
 /sbin/ipfwadm-wrapper              -> $(SEC_CRIT) ;
 /sbin/ipmaddr                    -> $(SEC_CRIT) ;
 /sbin/iptunnel                   -> $(SEC_CRIT) ;
 /sbin/plipconfig                 -> $(SEC_CRIT) ;
 /sbin/rpc.portmap                  -> $(SEC_CRIT) ;
 /sbin/rarp                     -> $(SEC_CRIT) ;
 /sbin/route                    -> $(SEC_CRIT) ;
 /usr/sbin/slattach               -> $(SEC_CRIT) ;
 /bin/ping                      -> $(SEC_CRIT) ;
}

 ###################################
 #                         ##
################################### #
#                         # #
# System Administration Programs # #
#                         ##
###################################

(
 rulename = "System Administration Programs",
 severity = $(SIG_HI)
)
{
 /usr/sbin/atd                    -> $(SEC_CRIT) ;
 /usr/sbin/crond                  -> $(SEC_CRIT) ;
 /usr/bin/crontab                 -> $(SEC_CRIT) ;
 /usr/bin/fuser                   -> $(SEC_CRIT) ;
 /sbin/halt                     -> $(SEC_CRIT) ;
 /sbin/init                     -> $(SEC_CRIT) ;
 /bin/killall                   -> $(SEC_CRIT) ;
 /sbin/killall5                   -> $(SEC_CRIT) ;
 /sbin/rmt                      -> $(SEC_CRIT) ;
 /usr/sbin/rpc.lockd                -> $(SEC_CRIT) ;
 /usr/sbin/rpc.statd              -> $(SEC_CRIT) ;
 /sbin/shutdown                   -> $(SEC_CRIT) ;
 /bin/sulogin                     -> $(SEC_CRIT) ;
 /sbin/sulogin                    -> $(SEC_CRIT) ;
 /sbin/swapon                     -> $(SEC_CRIT) ;
 /usr/sbin/syslogd                -> $(SEC_CRIT) ;
 /bin/pwd                       -> $(SEC_CRIT) ;
 /usr/bin/pwd                     -> $(SEC_CRIT) ;
 /bin/uname                     -> $(SEC_CRIT) ;
 /usr/bin/uname                   -> $(SEC_CRIT) ;
}

 ########################################
 #                         ##
######################################## #
#                         # #
# Hardware and Device Control Programs # #
#                         ##
########################################
(
 rulename = "Hardware and Device Control Programs",
 severity = $(SIG_HI)
)
{
```

```
 /sbin/hwclock                  -> $(SEC_CRIT) ;
 /sbin/losetup                  -> $(SEC_CRIT) ;
}

 ###############################
 #                             ##
################################ #
#                             # #
# System Information Programs # #
#                             ##
###############################
(
 rulename = "System Information Programs",
 severity = $(SIG_HI)
)
{
 /sbin/kernelversion            -> $(SEC_CRIT) ;
 /sbin/runlevel                 -> $(SEC_CRIT) ;
}

 ###################################
 #                               ##
################################### #
#                               # #
# Application Information Programs # #
#                               ##
###################################

(
 rulename = "Application Information Programs",
 severity = $(SIG_HI)
)
{
 /sbin/genksyms                 -> $(SEC_CRIT) ;
 /bin/sln                       -> $(SEC_CRIT) ;
}

 ############################
 #                          ##
########################### #
#                          # #
# Shell Related Programs # #
#                          ##
 ################
 #           ##
################ #
#           # #
# OS Utilities # #
#           ##
################
(
 rulename = "Operating System Utilities",
 severity = $(SIG_HI)
)
{
 /bin/cat                       -> $(SEC_CRIT) ;
 /bin/date                      -> $(SEC_CRIT) ;
 /bin/dd                        -> $(SEC_CRIT) ;
 /bin/df                        -> $(SEC_CRIT) ;
 /bin/echo                      -> $(SEC_CRIT) ;
 /bin/egrep                     -> $(SEC_CRIT) ;
 /usr/bin/elvis                 -> $(SEC_CRIT) ;
 /usr/bin/egrep                 -> $(SEC_CRIT) ;
 /bin/false                     -> $(SEC_CRIT) ;
 /usr/bin/false                 -> $(SEC_CRIT) ;
 /bin/fgrep                     -> $(SEC_CRIT) ;
 /usr/bin/fgrep                 -> $(SEC_CRIT) ;
```

```
  /usr/bin/gawk                 -> $(SEC_CRIT) ;
  /usr/bin/igawk                -> $(SEC_CRIT) ;
  /bin/grep                   -> $(SEC_CRIT) ;
  /usr/bin/grep                 -> $(SEC_CRIT) ;
  /bin/true                   -> $(SEC_CRIT) ;
  /usr/bin/true                 -> $(SEC_CRIT) ;
  /bin/arch                   -> $(SEC_CRIT) ;
  /usr/bin/basename               -> $(SEC_CRIT) ;
  /bin/dmesg                  -> $(SEC_CRIT) ;
  /bin/ed                     -> $(SEC_CRIT) ;
  /bin/gunzip                   -> $(SEC_CRIT) ;
  /usr/bin/gunzip               -> $(SEC_CRIT) ;
  /bin/gzip                   -> $(SEC_CRIT) ;
  /bin/hostname                 -> $(SEC_CRIT) ;
  /bin/kill                   -> $(SEC_CRIT) ;
  /bin/ln                     -> $(SEC_CRIT) ;
  /bin/login                  -> $(SEC_CRIT) ;
  /bin/ls                     -> $(SEC_CRIT) ;
  /bin/more                   -> $(SEC_CRIT) ;
  /usr/bin/more                 -> $(SEC_CRIT) ;
  /bin/mv                     -> $(SEC_CRIT) ;
  /bin/netstat                  -> $(SEC_CRIT) ;
  /sbin/netconfig.color             -> $(SEC_CRIT) ;
  /sbin/netconfig.tty             -> $(SEC_CRIT) ;
  /usr/bin/nice                 -> $(SEC_CRIT) ;
  /bin/ps                     -> $(SEC_CRIT) ;
  /usr/bin/sed                  -> $(SEC_CRIT) ;
  /sbin/setserial               -> $(SEC_CRIT) ;
  /sbin/setserialbits             -> $(SEC_CRIT) ;
  /bin/sleep                  -> $(SEC_CRIT) ;
  /usr/bin/sleep                -> $(SEC_CRIT) ;
  /usr/bin/sort                 -> $(SEC_CRIT) ;
  /usr/bin/tsort                -> $(SEC_CRIT) ;
  /bin/stty                   -> $(SEC_CRIT) ;
  /usr/bin/stty                 -> $(SEC_CRIT) ;
  /bin/su                     -> $(SEC_CRIT) ;
  /bin/sync                   -> $(SEC_CRIT) ;
  /bin/tar                    -> $(SEC_CRIT) ;
  /usr/bin/vi                   -> $(SEC_CRIT) ;
  /bin/zcat                   -> $(SEC_CRIT) ;
  /usr/bin/zcat                 -> $(SEC_CRIT) ;
}

  ##############################
 #                            ##
############################## #
#                            # #
# Critical Utility Sym-Links # #
#                            ##
##############################
(
  rulename = "Critical Utility Sym-Links",
  severity = $(SIG_HI)
)
{
  /sbin/clock                 -> $(SEC_CRIT) ;
  /sbin/kallsyms                -> $(SEC_CRIT) ;
  /sbin/ksyms                 -> $(SEC_CRIT) ;
  /sbin/lsmod                 -> $(SEC_CRIT) ;
  /sbin/modprobe                -> $(SEC_CRIT) ;
  /sbin/netconfig               -> $(SEC_CRIT) ;
  /sbin/pidof                 -> $(SEC_CRIT) ;
  /sbin/poweroff                -> $(SEC_CRIT) ;
  /sbin/swapoff                 -> $(SEC_CRIT) ;
  /sbin/reboot                -> $(SEC_CRIT) ;
  /sbin/rmmod                   -> $(SEC_CRIT) ;
  /sbin/telinit                 -> $(SEC_CRIT) ;
```

```
 /usr/bin/awk                    -> $(SEC_CRIT) ;
 /bin/dnsdomainname              -> $(SEC_CRIT) ;
 /bin/domainname                 -> $(SEC_CRIT) ;
 /usr/bin/ex                     -> $(SEC_CRIT) ;
 /bin/nisdomainname              -> $(SEC_CRIT) ;
 /bin/red                        -> $(SEC_CRIT) ;
 /bin/ypdomainname               -> $(SEC_CRIT) ;
}


 ##########################
#                  ##
########################## #
#                  # #
# Temporary directories # #
#                  ##
##########################
(
 rulename = "Temporary directories",
 recurse = false,
 severity = $(SIG_LOW)
)
{
 /usr/tmp                        -> $(SEC_INVARIANT) ;
 /var/tmp                        -> $(SEC_INVARIANT) ;
 /tmp                            -> $(SEC_INVARIANT) ;
}

 ###############
#           ##
############### #
#           # #
# Local files # #
#           ##
###############
(
 rulename = "User binaries",
 severity = $(SIG_MED)
)
{
 /sbin                           -> $(SEC_BIN) (recurse = 1) ;
 /usr/sbin                       -> $(SEC_BIN) (recurse = 1) ;
 /usr/bin                        -> $(SEC_BIN) (recurse = 1) ;
}

(
 rulename = "Shell Binaries",
 severity = $(SIG_HI)
)
{
 /bin/sh                         -> $(SEC_BIN) ;
 /bin/bash                       -> $(SEC_BIN) ;
 /usr/bin/bash                   -> $(SEC_BIN) ;
}

(
 rulename = "Security Control",
 severity = $(SIG_HI)
)
{
 /etc/group                      -> $(SEC_CRIT) ;
 /var/spool/cron/crontabs        -> $(SEC_CRIT) ; # Uncomment when this file exists
 /var/spool/cron/crontabs/cron.update -> $(SEC_CRIT) ; # Uncomment when this file exists
}

#(
#  rulename = "Boot Scripts",
```

```
#  severity = $(SIG_HI)
#)
#{
  /etc/rc.d/rc.6                -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.K                -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.modules          -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.M                -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.S                -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.cdrom            -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.4                -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.0                -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.sshd             -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.inet1            -> $(SEC_CONFIG) ;
  /etc/rc.d/rc.inet2            -> $(SEC_CONFIG) ;
#  /etc/rc.bsdnet               -> $(SEC_CONFIG) ;
#  /etc/rc.dt                   -> $(SEC_CONFIG) ;
#  /etc/rc.net                  -> $(SEC_CONFIG) ;
#  /etc/rc.net.serial           -> $(SEC_CONFIG) ;
#  /etc/rc.nfs                  -> $(SEC_CONFIG) ;
#  /etc/rc.powerfail            -> $(SEC_CONFIG) ;
#  /etc/rc.tcpip                -> $(SEC_CONFIG) ;
#  /etc/trcfmt.Z                -> $(SEC_CONFIG) ;
#}

(
  rulename = "Login Scripts",
  severity = $(SIG_HI)
)
{
  /etc/csh.cshrc                -> $(SEC_CONFIG) ;
  /etc/csh.login                -> $(SEC_CONFIG) ;
  # /etc/tsh_profile            -> $(SEC_CONFIG) ; #Uncomment when this file exists
  /etc/profile                  -> $(SEC_CONFIG) ;
}

# Libraries
(
  rulename = "Libraries",
  severity = $(SIG_MED)
)
{
  /usr/lib                      -> $(SEC_BIN) ;
  /usr/local/lib                -> $(SEC_BIN) ;
}


#####   STOPPING HERE IN SLACK CUSTOM!  DELETE THIS LINE WHEN DONE #####



  ######################################################
 #                                    ##
###################################################### #
#                                    # #
# Critical System Boot Files                 # #
# These files are critical to a correct system boot. # #
#                                    ##
######################################################

(
  rulename = "Critical system boot files",
  severity = $(SIG_HI)
)
{
    /boot                       -> $(SEC_CRIT) ;
    /sbin/lilo                  -> $(SEC_CRIT) ;
    !/boot/System.map ;
```

```
    !/boot/module-info ;

    # other boot files may exist.  Look for:
    #/ufsboot                       -> $(SEC_CRIT) ;
}
  ################################################
 ################################################
 # These files change every time the system boots ##
 ################################################
(
 rulename = "System boot changes",
 severity = $(SIG_HI)
)
{
    !/var/run/ftp.pids-all ; # Comes and goes on reboot.
    !/root/.enlightenment ;
    /dev/log                    -> $(SEC_CONFIG) ;
    /dev/cua0                   -> $(SEC_CONFIG) ;
    # /dev/printer              -> $(SEC_CONFIG) ; # Uncomment if you have a printer device
    /dev/console                -> $(SEC_CONFIG) -u ; # User ID may change on console login/logout.
    #/dev/tty2                  -> $(SEC_CONFIG) ; # tty devices
    /dev/tty3                   -> $(SEC_CONFIG) ; # are extremely
    /dev/tty4                   -> $(SEC_CONFIG) ; # variable
    /dev/tty5                   -> $(SEC_CONFIG) ;
    /dev/tty6                   -> $(SEC_CONFIG) ;
    /dev/random                 -> $(SEC_CONFIG) ;
    /dev/urandom                -> $(SEC_CONFIG) ;
    /dev/initctl                -> $(SEC_CONFIG) ;
    /var/lock/syslog            -> $(SEC_CONFIG) ;
    # /var/lock/subsys/inet     -> $(SEC_CONFIG) ; #Uncomment when this file exists
    # /var/lock/subsys/named    -> $(SEC_CONFIG) ; #Uncomment when this file exists
    # /var/lock/subsys/nfs      -> $(SEC_CONFIG) ; #Uncomment when this file exists
    # /var/lock/subsys/sound    -> $(SEC_CONFIG) ; #Uncomment when this file exists
    # /var/lock/subsys/smb      -> $(SEC_CONFIG) ; #Uncomment when this file exists
    /var/run                    -> $(SEC_CONFIG) ; # daemon PIDs
    #/var/spool/lpd/lpd.lock    -> $(SEC_CONFIG) ; #Uncomment when this file exists
    /var/log                    -> $(SEC_CONFIG) ;
    /etc/issue.net              -> $(SEC_CONFIG) -i ; # Inode number changes
    /etc/ioctl.save             -> $(SEC_CONFIG) ;
    /etc/issue                  -> $(SEC_CONFIG) ;
    /etc/.pwd.lock              -> $(SEC_CONFIG) ;
    /etc/mtab                   -> $(SEC_CONFIG) -i ; # Inode number changes on any mount/unmount
    /lib/modules                -> $(SEC_CONFIG) ;
    # /lib/modules/preferred    -> $(SEC_CONFIG) ; #Uncomment when this file exists
}

# These files change the behavior of the root account
(
 rulename = "Root config files",
 severity = 100
)
{
    /root                       -> $(SEC_CRIT) ; # Catch all additions to /root
    /root/mail                  -> $(SEC_CONFIG) ;
    /root/Mail                  -> $(SEC_CONFIG) ;
    /root/.mc                   -> $(SEC_CONFIG) ;
    /root/.bashrc               -> $(SEC_CONFIG) ;
    /root/.bash_profile         -> $(SEC_CONFIG) ;
    /root/.bash_logout          -> $(SEC_CONFIG) ;
    /root/.bash_history         -> $(SEC_CONFIG) ;
}

  ################################
 #                      ##
################################ #
#                      # #
# Critical configuration files # #
```

```
#                       ##
################################
(
  rulename = "Critical configuration files",
  severity = $(SIG_HI)
)
{
    /etc/conf.linuxconf             -> $(SEC_BIN) ;
    # /etc/conf.modules               -> $(SEC_BIN) ; # No longer used?
    #/etc/crontab                   -> $(SEC_BIN) ;
    #/etc/cron.hourly                -> $(SEC_BIN) ;
    #/etc/cron.daily                 -> $(SEC_BIN) ;
    #/etc/cron.weekly                -> $(SEC_BIN) ;
    #/etc/cron.monthly               -> $(SEC_BIN) ;
    /etc/default                    -> $(SEC_BIN) ;
    /etc/fstab                      -> $(SEC_BIN) ;
    /etc/exports                    -> $(SEC_BIN) ;
    #/etc/group-                    -> $(SEC_BIN) ;  # changes should be infrequent
    /etc/host.conf                  -> $(SEC_BIN) ;
    /etc/hosts.allow                -> $(SEC_BIN) ;
    /etc/hosts.deny                 -> $(SEC_BIN) ;
    /etc/httpd/conf                 -> $(SEC_BIN) ;  # changes should be infrequent
    /etc/protocols                  -> $(SEC_BIN) ;
    /etc/services                           -> $(SEC_BIN) ;
    /etc/rc.d/init.d                -> $(SEC_BIN) ;
    /etc/rc.d                       -> $(SEC_BIN) ;
    #/etc/mail.rc                   -> $(SEC_BIN) ;
    /etc/motd                       -> $(SEC_BIN) ;
    # /etc/named.boot                 -> $(SEC_BIN) ;
    /etc/passwd                     -> $(SEC_CONFIG) ;
    /etc/passwd-                    -> $(SEC_CONFIG) ;
    /etc/profile.d                  -> $(SEC_BIN) ;
    /var/lib/nfs/rmtab              -> $(SEC_BIN) ;
#    /usr/sbin/fixrmtab              -> $(SEC_BIN) ;
    /etc/rpc                        -> $(SEC_BIN) ;
    /etc/sysconfig                  -> $(SEC_BIN) ;
    /etc/smb.conf                   -> $(SEC_CONFIG) ;
    /etc/gettydefs                  -> $(SEC_BIN) ;
    /etc/nsswitch.conf              -> $(SEC_BIN) ;
    /etc/hosts                      -> $(SEC_CONFIG) ;
    /etc/inetd.conf                 -> $(SEC_CONFIG) ;
    /etc/inittab                    -> $(SEC_CONFIG) ;
    /etc/resolv.conf                -> $(SEC_CONFIG) ;
    /etc/syslog.conf                -> $(SEC_CONFIG) ;
}

 ####################
 #               ##
#################### #
#               # #
# Critical devices # #
#               ##
####################
(
  rulename = "Critical devices",
  severity = $(SIG_HI),
  recurse = false
)
{
    /dev/kmem                       -> $(Device) ;
    /dev/mem                        -> $(Device) ;
    /dev/null                       -> $(Device) ;
    /dev/zero                       -> $(Device) ;
    /proc/devices                   -> $(Device) ;
    /proc/net                       -> $(Device) ;
    /proc/sys                       -> $(Device) ;
    /proc/cpuinfo                   -> $(Device) ;
```

```
    /proc/modules               -> $(Device) ;
    /proc/mounts                -> $(Device) ;
    /proc/dma                   -> $(Device) ;
    /proc/filesystems           -> $(Device) ;
    /proc/pci                   -> $(Device) ;
    /proc/interrupts            -> $(Device) ;
#   /proc/rtc                   -> $(Device) ;
    /proc/ioports               -> $(Device) ;
#   /proc/scsi                  -> $(Device) ;
    /proc/kcore                 -> $(Device) ;
    /proc/self                  -> $(Device) ;
    /proc/kmsg                  -> $(Device) ;
    /proc/stat                  -> $(Device) ;
    /proc/ksyms                 -> $(Device) ;
    /proc/loadavg               -> $(Device) ;
    /proc/uptime                -> $(Device) ;
    /proc/locks                 -> $(Device) ;
    /proc/version               -> $(Device) ;
#   /proc/mdstat                -> $(Device) ;
    /proc/meminfo               -> $(Device) ;
    /proc/cmdline               -> $(Device) ;
    /proc/misc                  -> $(Device) ;
}

# Rest of critical system binaries
(
  rulename = "OS executables and libraries",
  severity = $(SIG_HI)
)
{
    /bin                        -> $(SEC_BIN) ;
    /lib                        -> $(SEC_BIN) ;
}

#=============================================================================
#
# Copyright 2000 Tripwire, Inc. Tripwire is a registered trademark of Tripwire,
# Inc. in the United States and other countries. All rights reserved.
#
# Linux is a registered trademark of Linus Torvalds.
#
# UNIX is a registered trademark of The Open Group.
#
#=============================================================================
#
# Permission is granted to make and distribute verbatim copies of this document
# provided the copyright notice and this permission notice are preserved on all
# copies.
#
# Permission is granted to copy and distribute modified versions of this
# document under the conditions for verbatim copying, provided that the entire
# resulting derived work is distributed under the terms of a permission notice
# identical to this one.
#
# Permission is granted to copy and distribute translations of this document
# into another language, under the above conditions for modified versions,
# except that this permission notice may be stated in a translation approved by
# Tripwire, Inc.
#
# DCM
```

---twpol.txt --- end ---

References

- Securing Linux - The SANS Institute
- SANS Baltimore Track 6 Course Material - Hal Pomeranz, Dr. Matt Bishop, Steve Acheson, Lee Brotzman
- Solaris 8 Installation Checklist - Jeff Campione, SANS GCUX Practical
- IpChains Firewall scripts - Lee Brotzman
- Chroot-BIND8 Howto Scott Wunsch