Global Information Assurance Certification Paper

Step by Step: Secure Slackware 8.0 Workstation
Andrew Nall
GCUX Practical Assignment v.1.9 – Option 1

INTRODUCTION TO LINUX:

In October of 1991 Linus Torvalds, then a University of Helsinki student in the Republic of Finland, posted the original version of Linux to USENET. Based on Minix (and subsequently UNIX), Torvalds began hacking out the shortcomings in Minix to develop an operating system based on UNIX, but freely available and specifically designed for X86 architecture machines [Parker, Tim. Linux Unleashed.9-10].

The immediate acceptance of Linux by the USENET community was advantageous in providing a large number of programmers with a new toy to play with and write code for. Many skilled developers immediately began refining and tweaking Torvalds' experiment, as well as porting existing code from UNIX to Linux.

Since that time, Linux has grown into a full-featured operating system. Countless Linux-related projects have been completed, further expanding the capabilities of the "little OS that could". Countless more projects are ongoing. Linux has become the operating system of choice for computer science students, developers, and even first amendment activists (based in large part on the prevailing principles of the GPL licensing scheme).

Available for many different architectures, and incorporating many previously UNIX-only services, Linux has grown into a viable server operating system, particularly for small high-tech companies who are loathe to spend large amounts of money for licenses and support contracts. Linux allows them to spend the money instead on network architecture, bandwidth, and hardware.

Linux's capabilities as an operating system for desktops and laptops easily rival those of expensive closed source operating systems. Linux-only shops are becoming a more frequent sight in the tech industry, wherein every system on the company intranet from servers to workstations, is running Linux.

Since the original posting of Torvald's Linux kernel, dozens of different distributions have been released. Red Hat, Mandrake, SuSe, Slackware, Debian, Yellow Dog, Storm, and Stampede are just a few of the many distributions available. Some are sold as "premium" versions and offer support contracts (Red Hat, the now-defunct Corel Linux). Some exhibit incredibly intuitive graphical installers, to offer the Linux neophyte an easier jumping-in point (Mandrake, most notably). Some are designed with Apple PPC hardware in mind (Yellow Dog, PPCLinux).

These different distributions, however, are all built around the same kernel, which is updated and maintained independently of any one company or distribution. The current stable kernel version is 2.4.5, but it is constantly being refined.

Parallel to Linux's phenomenal popularity with computer professionals, is its popularity with black-hat, or malicious, hackers. Linux allows the black-hat a large repository of tools capable of scanning for and exploiting vulnerabilities on remote systems. Things such as forging TCP/IP packets or scanning large blocks of IP addresses for specific vulnerabilities, which are very difficult to do on other operating systems, can often be accomplished easily on Linux systems. This is not to say that malicious activity is impossible under other operating systems such as Microsoft or Apple, but the power given to a superuser or root account in Linux allows greater opportunity for abuse than that given to an administrator of a Windows NT system. The tools written for Linux are also much more potent than those written for other operating systems.

Another reason exists for Linux's popularity with malicious hackers. If administrated properly, Linux has the potential to be significantly more secure than nearly any other operating system. Malicious hackers have wisely latched onto the Linux movement not only as a means of compromising other systems, but also as a means of protecting their own data.

Creating a secure Linux installation for use on a single-user workstation is not particularly difficult, but it does require some forethought, discretion, and maintenance. This paper will discuss the method by which to install and configure Slackware Linux 8.0 on a typical home system. We will cover installation, configuration, hardening, and maintaining a secure Slackware 8.0 desktop workstation.

Slackware is one of the oldest Linux distributions, and maintains a loyal following. It is a very basic Linux distribution, which makes it ideal for the purposes of this paper. Although natively bereft of some of the flashier features of other distributions (Debian's apt-get for example), its beauty lies in the very fact that it is a "vanilla" distribution. The ncurses based, menu driven installer is easy to understand, and will work on virtually any system, requiring far less resources than some of it's flashier GUI-based (graphical user interface) cousin distributions.

HARDWARE:

Although the basic processes set forth in this paper should work with virtually any x86-based hardware configuration, the paper was written with the following specific hardware in mind:

Central Processing Unit:
    Intel PentiumIII CPU @ 550MHz
Motherboard:
    Asus P3V4X
    Award Plug and Play BIOS Ver. 1.0a
Storage:
    30 Gigabyte Maxtor Hard Drive
    (Primary IDE Master)
Memory:
    384 MB RAM
    (3 128MB DIMMs @ 100MHz bus)
Video Adapter:
    Diamond V770 Video Card
        Based on NVIDIA TNT2 Ultra Chipset
        32MB VDRAM
        (AGP)
Network Interface Card (NIC):
    Netgear 10/100 Mb/s Ethernet Network Interface Card
        (PCI)
Input Devices:
    1.44 Megabyte Floppy Drive
    Samsung 52x CD-ROM Drive
        (Secondary IDE Master)
    Standard PS/2 Keyboard
    Standard PS/2 Mouse
Output Devices:
    17" SVGA Monitor
    Creative Labs Sound Blaster AWE64 Sound Card
        (PCI)

By no means is the purpose of this paper confined or restricted to the hardware listed above, but deviations on the part of the reader may be necessary to compensate for deviations in hardware. I will attempt to explicitly mention when one should be mindful of hardware concerns, but the burden must ultimately fall on the reader, as the possible hardware configurations are nigh-limitless, and specific errata for all possible hardware is far outside the scope of this endeavor.

OBJECTIVE:

Our completed system will be a single-user desktop workstation based on Slackware Linux 8.0 (available at http://www.slackware.com). We will install and configure a windowed GUI (Xfree86 4.1.0 with KDE 2.1.1) for ease of use, and offer secure shell services (OpenSSH 2.9p1) to allow remote access to the host and secure file transfers via scp, the secure copy program. The system will be used for development (C, C++, and Perl) as well as day-to-day tasks such as web, mail, news, and IRC.

My primary goal in assembling this system is to provide access to files on my home system whilst I'm off site, and vice versa. We will achieve this through use of the aforementioned SSH service, which will be the only service running. It is important to note that if the reader does not require the ability to access the machine remotely, that SSH should also be disabled, as any running services present the opportunity for abuse. The reader should carefully consider whether or not the SSH service (and any other services, for that matter) is truly necessary to accomplish their goals; if not it should be disabled. The system will reside on my home network, which consists of less than ten hosts running various *nix and Microsoft operating systems. All hosts on the network are in NAT'ed address space using arbitrary IP addresses in the 192.168.x.x/16 range. Between the network and the DSL Internet connection lies a Linux based firewall/intrusion detection system. Although it will be necessary to enable port forwarding for SSH access to work, we will largely ignore this extra line of defense, and secure the system as though it were to be placed directly on the Internet with a routable IP address. I only mention this to allay any confusion about the use of non-routable IP addresses during the system's configuration.

RISK ANALYSIS:

As we are building a workstation rather than a server, we will have very few services running. Services are traditionally the weakest link in a Linux system, but after hardening the system only SSH, a service built primarily with security in mind, will be running. Our main security concerns will stem from the large amount of software installed on the system, some of which may contain known or undiscovered vulnerabilities. After applying patches and locking down the file system the risk of such exploits will be considerably decreased, but it will be imperative to constantly maintain the system with security in mind as new patches and exploits become known. Our biggest concern for the completed, online system will be users with physical access to the host and users with legitimate SSH accounts. Because this is a home system, it is unlikely that it would be targeted for abuse through physical access methods other than outright theft or mundane occurrences (a roommate tripping over and disconnecting a CAT-5 cable is a surprisingly effective denial of service attack). However, this does not excuse us from considering physical security. At the very least, we can move the system into a padlocked closet, secure all cables together with Velcro strips, and make sure that no cables lie in high traffic areas where they could inadvertently or purposefully be tampered with. As a countermeasure against theft, it is also a good idea to save receipts from equipment purchases and to write down and secure, preferably offsite, a list of the serial numbers of each piece of hardware. This would make recovery and identification of stolen goods considerably easier for law enforcement agencies. A useful and inexpensive trick is to mark each piece of hardware with a yellow highlighter marker, inscribing your name, initials, or another identifying mark. The highlighter markings will be next to invisible on the components, but when held under a backlight in a darkened room they become very legible (the same principle can be applied with much more specialized, expensive, and durable inks).

A larger concern is our SSH service, which provides access to the system from the Internet. The unfortunate side effect being that it presents an opportunity for anyone on the Internet to also access the system. SecureShell, as its name implies, is relatively secure, and has been rigorously tested for vulnerabilities. Although not inconceivable that it could be compromised, there is always the possibility, which is why we will find it necessary to use other methods to regularly verify system integrity. A larger concern is other users with legitimate system accounts. Frequently friends and co-workers will ask for a shell account on your system when they discover that you host a shell server with an always-on connection. Although tempting, it is recommended that we limit shell account access as much as possible and abstain from creating accounts for other users.

STEP BY STEP GUIDE:

Disconnect Host from Network:

The first thing we will do is disconnect the system from the network. We needn't do anything fancy, just unplug the CAT-5 cable from the Ethernet card. This will prevent access to the system from the Internet until we have secured it.

Setting a BIOS Password

The first step will be to create passwords in the BIOS. The BIOS, or Basic Input-Output System is responsible for the boot strap, or boot, process. BIOS handles IDE and SCSI checks, maintains the hardware clock, CPU voltage settings, and initial IRQs (Interrupt Requests). Assuming all goes well in the BIOS checks, the system is then ready for the operating system to be loaded.

During boot, there will usually be a memory check, followed by several hardware checks (IDE check, Plug and Play hardware check). During this time a message will be displayed which instructs the user in how to access the BIOS. On most motherboards using the Award BIOS, the message will state:

```
Press DEL to enter setup
```

On Gateway, Dell, and other manufactured systems, there is usually a different key or combination of keys necessary to reach the BIOS (ALT+B for example).

In the BIOS screen, under the MAIN tab (assuming your motherboard is based on the Award BIOS) we have two relevant fields: Supervisor Password and User Password. Supervisor Password is used to restrict access to the BIOS. Select and set a Supervisor Password. After confirming, you will be unable to make any changes to the BIOS without first entering the password. After setting the supervisor password, we can disable "Auto-Detect" on our IDE devices as well as disabling CDROM booting. This would prevent someone who has gained physical access to our system from placing another hard drive in the system or booting from an Operating System install disk; two methods of gaining root access on a system. The User Password is requested and checked during normal boot up after the boot strap process has completed but before the operating system is loaded. This provides another layer of protection. Unfortunately, neither of these methods is incredibly effective, as an attacker with physical access to the system can simply remove the CMOS battery from the motherboard, wiping these password settings. Nonetheless, theses are good steps to take as a deterrent if nothing else.

Finally, a brief digression on password security: Poor password choosing is an issue that has plagued system administrators for years. The vast majority of user selected passwords are grossly inadequate. Passwords should contain a combination of upper and

lower case letters, numbers, and other ASCII characters such as punctuation marks. Passwords based on the user's name, user's spouse's name, family pet name, birthdates, social security numbers, etc. are invariably poor selections for passwords, but are far too common. When selecting passwords; keep security in mind. Don't reuse passwords. Don't write passwords down. Any word or phrase that can be found in a dictionary (including foreign language dictionaries) is a poor password choice. Be mindful of password security at all times. A properly chosen password can greatly increase system security, just as a poorly chosen password can easily cripple ones ability to keep data and systems secure. An important point to keep in mind while creating passwords for this system is that the biggest point of failure lies in the primary user. It would be significantly easier for an attacker to Social Engineer system passwords than to discover them through brute force algorithm cracking or a dictionary attack. For this reason, it's a bad idea to use personal information of any type as a basis for choosing passwords. More information on choosing appropriate passwords can be found in section 4.3.1 of RFC 1244 (http://www.faqs.org/rfcs/rfc1244.html).

Installation:

Most modern personal computers are capable of booting from CDROM. If you are using older hardware, it may be necessary to download the Slackware boot disks. This install will assume that you have the capability to boot from CDROM.

At boot you will be presented with the Slackware splash screen and a `boot:` prompt. If you need to pass extra parameters to the kernel, this is the place to do so. We will assume this to be unnecessary, so we will simply hit ENTER at this prompt.

Keyboard Map:

The next prompt we encounter allows us to select a non-standard keyboard map. We will be using a normal US QWERTY keyboard map, so we will simply hit ENTER at this prompt.

Login:

Next we must login to the system. At the `slackware login:` prompt we will login as root. Password is null, you will not be prompted for one.

Disk Partitioning:

There are many different ways to partition your disk for Linux, and virtually everyone who has been using Linux for any period of time has a method they consider to be the best. The simplest method is to create only two partitions: a swap partition and a root (/) partition.

There are, however, some security advantages in spreading your Linux system over multiple partitions. For example, separating /var from the rest of your system will prevent log files from consuming all of your hard drive space, a potential Denial of Service attack. Also, keeping /home separate confines the user accounts to a specific partition, making it easier to control user disk usage. At the very least, I'd suggest mounting root on a separate partition to prevent SUID exploits. SUID files are executed using owner permissions, meaning that a root-owned SUID file has great potential for exploit. Isolating root SUID files to the root partition can significantly reduce an attacker's ability to wreck havoc.

An excellent introduction to creating a multiple partition system is the one written for linuxnewbie.org by Danny DiPaolo, which can be found at http://www.linuxnewbie.org/nhf/intel/installation/partition/multi_partition.html

We will be using a multiple partition file system in this paper. After logging in as root we are presented with a # prompt, at which we will enter `cfdisk /dev/hda` (assuming the hard drive is IDE Primary Master).

We will begin by deleting any partitions that exist on the disk. Please keep in mind that this step will permanently destroy any and all data on the hard drive. If necessary, backup any data that you wish to retain before proceeding.

Choose the `DELETE` option for each partition currently on the drive, until the only thing listed under `FS Type` is Free Space.

Now we will create our swap partition. Select `New`. The swap partition will be a Primary partition. After selecting `Primary` you will be prompted for `Size (in MB)`. We will create a swap partition of 768 MB (double the amount of physical memory) and place it at the Beginning of the drive. You'll notice that the Size displays as 764.96 MB, even though we specified 768MB. This is normal due to the way hard drive cylinder boundaries are delineated. The name of your newly-created partition will appear as hda1. With hda1 highlighted, select `TYPE`. Type 82 is Linux Swap, so input `82` and hit return.

Next we will create our root partition. Since we have a 30 gigabyte drive, we can make the partitions a bit larger than is expressly necessary. We will create a 2 gigabyte root partition by first highlighting `Free Space`, then selecting `NEW` → `Primary` → `2000` → `Beginning`. The new partition should show up as FS Type: Linux, but if not, highlight it, select Type and choose 83 (Linux Native).

Next we will create three separate 6 gigabyte partitions, one each for /home, /usr, and /usr/local. For each we will select `New` → `Logical` → `6000` → `Beginning`. Make sure the File System type is Linux (83) for each. We are creating these as Logical

partitions, which does not affect functionality and allows for the creation of more than four partitions. You may only have four separate primary partitions, a restriction not applicable to logical partitions.

Next we will create two more logical partitions of one gigabyte each. These will act as our /tmp and /var partitions. Select `New → Logical → 1000 → Beginning` and make sure they are type Linux Native.

We should now have 7 separate partitions as follows:

Name	Part Type	FS Type	Size (MB)
Hda1	Primary	Linux Swap	764.96
Hda2	Primary	Linux	1998.75
Hda5	Logical	Linux	5996.23
Hda6	Logical	Linux	5996.23
Hda7	Logical	Linux	5996.23
Hda8	Logical	Linux	1003.49
Hda9	Logical	Linux	1003.49
	Pri/Log	Free Space	7970.30

We are leaving just under eight gigabytes of unpartitioned space. The reader is encouraged to use it as he wishes. Options include installing another operating system for a dual-boot configuration or just keeping the space available for future use as the system evolves.

Finally, we must set the root partition as bootable and write the new partition table to disk. Highlight hda2 and select the `Bootable` option. In the Flags column you should now see Boot next to hda2. Select `Write` and enter yes at the confirmation prompt. After the partition table is written, select `Quit`.

After being returned to a command prompt, enter `setup` and press enter to begin the installation process.

Initial Installation Menu:

You will be presented with a menu. The first item, HELP, contains a wealth of useful information. If you encounter installation problems the Slackware Setup HELP file is a good place to begin looking for answers.

The next option in the menu is KEYMAP. We previously specified our keymap, but if you made a mistake or need to change keymaps for installation, this is the place to do it.

To begin the actual installation process, select ADDSWAP. Assuming you set the FS Type of /dev/hda1 to Linux Swap, the installer should detect the swap partition. You will be prompted to confirm that you wish to configure this as your swap partition. Select Yes and wait as the partition is formatted. You should receive a SWAP SPACE CONFIGURED message when the formatting is complete.

Next, you are asked if you would like to continue with installation by setting up your target drives. Select yes.

You are prompted to select a root (/) partition. /dev/hda2 should be the first choice in the list. Next, select Format and 4096 respectively on the next two screens. After the format of /dev/hda2 is complete, you will be presented with the option to distribute your installation across the remaining partitions. Select hda5 → Format → 4096. When the format is completed you will be asked to specify where you want the new partition mounted. Hda5 will be /home. Repeat the process for the remaining partitions specifying hda6 as /usr; hda7 as /usr/local; hda8 as /var; and /hda9 as /tmp.

You should be presented with a screen as follows:

```
     DONE ADING LINUX PARTITIONS TO /etc/fstab
     Adding this information to your /etc/fstab:

/dev/hda2      /             ext2 defaults  1    1
/dev/hda5      /home         ext2 defaults  1    1
/dev/hda6      /usr          ext2 defaults  1    1
/dev/hda7      /usr/local    ext2 defaults  1    1
/dev/hda8      /var          ext2 defaults  1    1
/dev/hda9      /tmp          ext2 defaults  1    1
```

Next you will be prompted to continue to the SOURCE section to select install media. We will select option one: Install from a Slackware CD-ROM.

Package Selection:

Here we are presented with fifteen package series to choose from. These are broad categories containing many sub-packages. Depending on the uses you envision for the completed system, careful selection of packages will determine a base level of security for the completed system. Installing unnecessary packages not only consumes system resources, but can also increase potential for abuse. Below is a list of all package categories, along with suggestions as to whether they should be installed or not:

A – Base Linux System – Necessary to install.

AP – Applications that do not require X – Recommended that you select this package series, although we will be selective about the individual packages we choose to install.

D – Program Development – These are compilers, programming languages, and the like. Regardless of whether you are a developer or not, some of these should be installed.

E – GNU Emacs – Emacs word processor. Not necessary, but recommended as many applications use Emacs for text formatting.

F – FAQ Lists, HOWTO documentation – Not necessary, but recommended.

GTK – GTK and GNOME Programs for X – Definitely select if you will be using GNOME. If you are not using GNOME but will be using some flavor of Xwindows, it may be a good idea to select this package series if you'd like access to GNOME applications. Because we will be using KDE as our window manager and wish to minimize the number of potentially exploitable applications, we will not select the GTK package series.

K – Linux Kernel Source – Not necessary unless you plan on recompiling your kernel. We will not select the K package series, but you may wish to install the kernel source if you feel you will have need of it.

KDE – Qt and the K Desktop Environment for X – Defiantly select if you will be using KDE. If you are not using KDE but will be using some flavor of Xwindows, it is a good idea to select this package series. We will be using KDE as our primary window manager.

N – Networking – It is assumed that this system will eventually go on a network or the Internet. Networking should be selected. If you will not be connecting the system to a network do not select this package series.

T – TeX Typesetting Software – Not necessary, but recommended if you will be doing word processing or printing.

TCL – TCL/Tk Script Languages – Not necessary, but recommended for compatibility with some third party applications.

X – Xfree86 X Window System – Not necessary if you will be exclusively using a CLI (Command Line Interface). However, as we are building a personal workstation, a GUI (Graphical User Interface) will be preferred.

XAP – X Applications – Not necessary, but adds many programs which may be useful if you plan on using X Windows. Because we are attempting to minimize the number of potentially exploitable applications, we will not select the XAP package series.

XV – Xview (OpenLook Window Manager, apps) – Not necessary, but if you plan on using X Windows it is recommended. Because we are using KDE as our window manager and wish to limit the number of exploitable applications on the system, we will not select the XV package series.

Y – Games (that do not require X) – Unnecessary, but left to the discretion of the reader. We will not be selecting the Y package series for the purposes of this paper.

For our install we will select package series A, AP, D, E, F, KDE, N, TCL, and X. After selecting these package series, we will continue.

Prompting Mode:

We are next presented with the option to select our prompting mode. We will select "menu", which allows us to select groups of related packages. Certain packages within the package series will be denoted with an asterisk (*), which marks them as required for the base software of the package series. Two other classifications exist; recommended, which are already selected in the menu but can be deselected and optional which are not pre-selected but can be selected if desired. Appendix A contains a list of each package series selected above, and the package groups we wish to install. Keep in mind that these are not hare and fast rules. If you are unsure whether you require the functionality provided by a package group, you can choose not to install it. If you later decide you'd like the a specific package installed it is easy to retroactively add it.

Make Boot Disk:

Here we will make a LILO boot disk, which we can use for recovery in the event of a crash. Place a formatted floppy in the drive, select lilo, and follow the onscreen prompts.

Modem Configuration:

We are not using a modem, so we will select "no modem" on this screen. If you will be using a modem in your system configure it here but be sure to configure it to not accept incoming calls if at all possible, as that presents an obvious security risk

Screen Font Configuration:

Here we are presented with the opportunity to change our console font. Select "no" and continue with the installation.

Install LILO:

LILO is our boot loader, which tells the computer how to boot the operating system. On the LILO screen, we will select simple, and allow LILO to configure itself to boot Linux. If you are using a dual-boot system, or booting multiple kernels on the same machine, it is recommended you select the "expert" option, or manually configure your lilo.conf file at a later time.

Configure LILO To Use Frame Buffer Console:

Because we are using such a powerful video card, we will select "1024x768x64k" as our console. This should simplify our X Windows configuration, and allow more system messages on screen at boot.

Select LILO Destination:

Here we are presented with several options as to where LILO will be installed. We will install directly to the Master Boot Record by selecting the "MBR" option.

Mouse Configuration:

Here we will instruct Linux to create a /dev/mouse link to our pointing device by selecting "ps2" from the list.

GPM Configuration:

We do want to run gpm at boot time, so select Yes here. This allows us to cut and paste text in the console.

Configure Network:

We will use netconfig to install our Ethernet card and configure our adapter. Enter a hostname and domain as appropriate. I used "test" and "sans.org", respectively. If you are on an actual domain, you may want to speak with your administrator about adding your host info to the DNS zone file. Unfortunately, doing so could increase your visibility to a potential malicious hacker, and we won't be serving files anyway, so it is not recommended. For this system I will specify a "Static IP" address of "192.168.20.12", as my system lies in NAT'd space behind a firewall/router. Normally you would input the IP address provided by your ISP or network admin. We will use the default netmask of "255.255.255.0". Gateway will be "192.168.20.1". Nameserver will be "207.69.188.187", my ISP's nameserver. We will allow netconfig to probe for network cards, where it should find and load the necessary kernel module (in my case tulip.o).

Hardware Clock Set to UTC?:

Specify whether your hardware clock is set to local time or Universal time (local in my case). If you answer local time, you will be prompted to select your time zone. It's worth noting that if you use Local time rather than that UTC, you system may not handle daylight savings time transitions properly if the system is not on during the Sunday transition. If you use Local time, be sure to verify that the time adjusted properly, and if necessary, adjust it manually.

Select Default Window Manager for X:

Next you will be prompted for your default window manager, where we will select KDE.

Set Root Password:

Prompted for root password, we will set one. Keep in mind the previous discussion about proper password selection. After setting a root password, we are informed that setup has successfully completed. Exit setup and reboot using CTRL+ALT+DEL.

HARDENING THE SYSTEM:

Creating a User Account:

When the system finishes rebooting we receive a login prompt:

```
test login:
```

We will login as root with the password we specified earlier. It is seldom a good idea to perform tasks as root, as the possibility of inadvertently damaging the system is high. Because Slackware uses the Shadow Suite for added password security, we will create a new user account to work from by issuing the `useradd` command with some options:

```
# /usr/sbin/useradd dnall –m –s/bin/bash
```

The –m option tells useradd to create a home directory, -s specifies the user's shell (in this case bash). Useradd does not create a password for the user, so we will do that manually by issuing the passwd [username] command.

If for some reason you are not using the Shadow Suite, you can add a user instead with the adduser command, which would look similar to this:

```
Login name for new user []: dnall
User id for dnall [defaults to next available]: ENTER
Initial group for dnall [users]: ENTER
Additional groups for dnall []: ENTER
Dnall's home directory [/home/dnall]: ENTER
Dnall's shell [/bin/bash]: ENTER
Dnall's account expiry date []: ENTER
```

You will be presented with a list of the selections you have just made, and asked to confirm. Press ENTER. You will then be prompted for additional information, such as Full Name, Phone Number, and Room Number. This is unnecessary information that has historically been provided in response to finger queries. Providing this information allows potential access to sensitive data, and serves no purpose other than to create a security hole. It is recommended that you leave this information blank. A crafty system administrator may actually input false data as a diversionary tactic against would-be attackers. Finally, you will be prompted for a password. Once again, keep in mind the previous discussion on choosing secure passwords. Also, remember that it is bad form to

use passwords in multiple places on the system. You should use different passwords for BIOS, root, user account, LILO, etc. It is recommended that you use the Shadow Suite if at all possible, as it adds the security of the MD5 encryption algorithm to the passwords stored on the system. Conveniently, the Shadow Suite should be in use by default on a Slackware 8 install.

After creating our user account, log out of the root account by issuing the `exit` command, then log back in as the newly created user. You may want to issue the startx command to test whether X Windows is working properly. After testing X Windows use the `su` command and root password to gain root privileges. Issue the `su` command and root password to gain root privledges for further configuration.

Lilo.conf Configuration:

Our LILO boot loader is still the operating system's first interaction with the computer. After the BIOS checks the hard disk the Master Boot Record is loaded, which brings up LILO. LILO then references our root partition and the operating system is loaded. The LILO configuration file is /etc/lilo.conf, and contains values for the boot image(s), target drive(s), and root partition. It also has other variables which we will use to add another, albeit minor, layer of security. Open /etc/lilo.conf in a text editor. In the "global" section at the top, you will see a reference to the boot message:

```
Message = /boot/boot_message.txt
```

This is one of several places that we will add a generic warning message, essentially stating that unauthorized use of the system is prohibited, and that usage will be monitored. This is a minor point, but can be useful if it ever becomes necessary to pursue legal action against an intruder. We will use our text editor to change /boot/boot_message.txt to something along the lines of:

```
* * * * * * * * * ** * * * * * * * * * * * * * * * * *
UNAUTHORIZED USAGE OF THIS SYSTEM IS PROHIBITED BY LAW.
THE ADMINISTRATOR RETAINS THE RIGHT TO MONITOR, EDIT,
DELETE, OR VIEW ANY AND ALL DATA STORED ON OR PASSING
THROUGH THIS SYSTEM. VIOLATERS WILL BE PROSECUTED TO THE
FULLEST EXTENT OF THE LAW
* * * * * * * * * ** * * * * * * * * * * * * * * * * *
```

Back to /etc/lilo.conf, where we will add a password to LILO which must be entered before the system will boot. The LILO password is stored in an unencrypted format, which could be easily obtained if the system were compromised, but this is still a useful additional level of security. We will add a line to lilo.conf, substituting a password of your choosing:

```
Password = #g4=TegdSt
```

Remember to select a secure, unique password. After adding the password parameter, save changes and issue the lilo command at your shell prompt. Finally, reboot using the shutdown –r now command. When the system comes back to the LILO boot screen, you should see the changes and be prompted for your LILO password.

Securing /etc/login.defs:

Now we will edit the /etc/login.defs file to tweak our login settings. Login.defs contains configuration control definitions for the login package.

--Change DIALUPS_CHECK_ENAB to "no"
We are not using dial-up.
--Uncomment SULOG_FILE /var/log/sulog
We do wish to log su activity
--Change PASS_MIN_LEN from 5 to 8
Require minimum of 8 characters for passwords
--Change PASS_MAX_DAYS from 99999 to 90
Force password change after 90 days

Save changes and exit.

Disabling Unneeded Services:

As root, issue the netstat –a command to get an idea for what services are running. There will likely be several daemons running that could present problems. Below is the output of the netstat command:

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q LocalAddress   ForeignAddress State
Tcp  0         0    *:login        *:*            LISTEN
Tcp  0         0    *:shell        *:*            LISTEN
Tcp  0         0    *:time         *:*            LISTEN
Tcp  0         0    *:netbios-scan *:*            LISTEN
Tcp  0         0    *:finger       *:*            LISTEN
Tcp  0         0    *:sunrpc       *:*            LISTEN
Tcp  0         0    *:auth         *:*            LISTEN
Tcp  0         0    *:ftp          *:*            LISTEN
Tcp  0         0    *:ssh          *.*            LISTEN
Tcp  0         0    *:telnet       *:*            LISTEN
Tcp  0         0    *:biff         *:*
Tcp  0         0    *:ntalk        *:*
Tcp  0         0    *.netbios-ns   *:*
```

```
Tcp  0          0     *:time          *:*
Tcp  0          0     *:sunrpc        *:*
```

Each of these represents a different service, and a different potential security hole. The only service we want to run is SSH, so we must disable the rest manually. Open /etc/inetd.conf in a text editor and comment out all lines. Write changes and exit. You may consider this an unnecessary step, as we are going to keep inetd from running anyway, but we will comment out the contents just to be on the safe side. Next we will open /etc/rc.d/rc.inet2 and comment out all lines except for the uncommented script which begins with:

```
# Start the OpenSSH SSH daemon
```

Write changes, exit, and issue the following command (as su'ed root):

```
# /etc/rc.d/rc.inet2
```

Finally, reboot the system by issuing the command `shutdown -r now`. After rebooting issue the `netstat -a` command to verify that SSH is the only service running:

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q LocalAddress   ForeignAddress State
Tcp    0       0    *:ssh              *:*       LISTEN
```

Securing SSH:

Now we will further secure our SSH client and server by editing their configuration files, which are /etc/ssh/ssh_config and /etc/ssh/sshd_config, respectively. Let's start with /etc/ssh/sshd_config, as our SSH server daemon is our primary concern. This config file is read when an SSH session is initialized from a remote host to our Slackware system.

We will change several things in this file. Starting from the top:

`LoginGraceTime 600` to `LoginGraceTime 150`

This reduces the time that SSH will wait to terminate the session after receiving a login request, measured in tenths of a second. A full minute seems a bit excessive, so we will reduce it to 15 seconds.

`PermitRootLogin yes` to `PermitRootLogin no`

Allowing remote root login is seldom a good idea as it gives an attacker an easily-guessable account name to brute force or dictionary attack, which will compromise the entire system if cracked.

After writing our changes to disk and exiting our text editor, we will restart the ssh daemon by issuing the following commands:

```
# /etc/rc.d/rc.sshd stop &&
> /etc/rc.d/rc.sshd start
```

Go ahead and open /etc/ssh/ssh_config in your text editor. You'll notice that everything in this file is commented out. We will leave it as such, but keep in mind that if you plan on ssh'ing from the system to a shell server, you will want to review this file.

Finally, we will use the already installed tcp_wrapper process to control access into and out of the system. Open /etc/hosts.allow in a text editor and add the following:

```
ALL:127.
ALL:192.168.20.
```

These two lines will allow connections from 127.0.0.0/8 (localhost) and 192.168.20.0/24 (internal network, behind firewall). It is not necessary to allow connections to localhost, but it will be helpful in testing SSH. After properly configuring and testing SSH, you may want to come back here and remove localhost from hosts.allow. If you will be SSH'ing into your system from other locations such as work or school, you will also want to add an additional line allowing connections from those IP addresses. For example, if your place of business owns the class-C IP block 207.69.188.0/24, you'll also add a line ALL:207.69.188. We should try to be as specific as possible to reduce the number of hosts which can connect. If you have a specific static IP address that you will be connecting from, add that to the hosts.allow instead (ALL:207.69.188.185, for example). If you will be in NAT'ed space, you can add your gateway IP or the NAT machine IP rather than the entire IP range.

If you plan on logging into the system from many undefined locations, it may be necessary to bypass this step. Hosts.allow and hosts.deny are encouraged if at all possible however, as they greatly reduce the number of hosts on the Internet capable of accessing the system.

Now we will deny access from all IP's other than those we have specifically allowed. Open /etc/hosts.deny in a text editor and add the line `ALL:ALL` which will disallow all connections except for those specified in hosts.allow (local host, intranet, and work, in our example).

OTHER SYSTEM CHANGES:

Cleaning Banners:

We will now make several changes to the system to provide further security, starting with cleaning our banners. Banners can provide a wealth of information to crackers, such as operating system version, service versions, and more. For example, when logging into the system via local console the following is displayed:

```
Welcome to Linux 2.4.5 (tty1):
Test login:
```

There is no reason to volunteer our kernel version (Linux 2.4.5) to snooping eyes, so we will remove it by editing /etc/issue (which is read by /etc/rc.d/rc.local). Currently the contents of /etc/issue will display "`Welcome to \s \r (\l)`". We'd only like to keep the terminal number displayed, so let's delete the existing line and add a new line in its place:

```
AUTHORIZED USE ONLY (\l)
```

While we are at it, we will also modify the Message of the Day (MOTD), which is displayed upon successful login. For legal reasons, as mentioned above, it is a good idea to make it very clear in the MOTD that the system administrator has full access to edit, view, delete, and monitor anything on the system. Open the /etc/motd in a text editor, remove the "Welcome to Linux" line and insert the following:

```
* * * * * * * * * ** * * * * * * * * * * * * * * * *
UNAUTHORIZED USE OF THIS SYSTEM IS PROHIBITED BY LAW. THE
ADMINISTRATOR RETAINS THE RIGHT TO MONITOR, EDIT, DELETE,
OR VIEW ANY AND ALL DATA STORED ON OR PASSING THROUGH
THIS SYSTEM. VIOLATERS WILL BE PROSECUTED TO THE FULLEST
EXTENT OF THE LAW.
* * * * * * * * * ** * * * * * * * * * * * * * * * *
```

Logout and back in to make sure the changes have taken effect. You will also now receive the MOTD when connecting via SSH, which you can test by issuing the command:

```
# ssh localhost
```

Other than confirming that you received the MOTD, this is a good way to test various other changes we made to SSH. Try ssh'ing as root to confirm that it is not allowed. Watch the password timeout length to confirm that it is not excessively long.

Patching:

We will now apply several security patches to address specific known vulnerabilities. An excellent source for finding known vulnerabilities based on distribution can be found at http://www.linuxsecurity.com/advisories/index.html. In the Slackware section we can easily find several patches to apply. It should be noted that poorly written patches can present possible vulnerabilities of their own, and in fact can undo some of the hardening that we have already performed. After applying changes, run back over the previous System Hardening steps, to ensure that everything is still working as it was before patching.

--zlib data corruption vulnerability (http://www.linuxsecurity.com/advisories/slackware_advisory-1973.html)

1. Download the patch from ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/zlib.tgz
2. As root: `# upgradepkg zlib.tgz`

--OpenSSH unauthorized access vulnerability (http://www.linuxsecurity.com/advisories/slackware_advisory-1944.html)

1. Download the patch from ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/openssh.tgz
2. As root:

   Stop sshd:

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

   Upgrade to new package:

   ```
   # upgradepkg openssh.tgz
   ```

   Restart sshd:

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

--multiple packages (http://www.linuxsecurity.com/advisories/slackware_advisory-1843.html)

1. Download the at package and sudo patch (we didn't install xchat, so we can disregard it):
   ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/at.tgz
   ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/sudo.tgz
2. As root:

   Install new at package on top of existing bin package:

   ```
   # installpkg at.tgz
   ```

   Upgrade xchat and sudo

   ```
   # upgradepkg sudo.tgz
   ```

--pine update fixes insecure URL handling (http://www.linuxsecurity.com/advisories/slackware_advisory-1801.html)

1. Download pine patch from:
ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/pine.tgz
2. As root:

```
# upgradepkg pine.tgz
```

--glibc glob overflow patches (http://www.linuxsecurity.com/advisories/slackware_advisory-1800.html)
1. Download the patches from:
ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/glibc.tgz
ftp://ftp.slackware.com/pub/slackware/slackware-8.0/patches/packages/glibcso.tgz
2. As root:

```
# upgradepkg glibcso.tgz glibc.tgz
```

There may be more patches necessary depending on how the reader has configured her system. It is recommended that the reader perform independent research of necessary patches, and install them as needed. A few resources for patch information and security advisories:

--The aforementioned http://www.linuxsecurity.com and http://www.linuxsecurity.com/advisories/slackware.html (Slackware-specific advisories)
--Packet Storm: http://packetstormsecurity.nl/
--The Bugtraq Mailing list: http://www.securityfocus.com/popups/forums/bugtraq/intro.shtml
--Slackware Security Mailing List: http://slackware.com/lists/

SYSTEM LOGGING:

Syslog.conf:

Next we will take a look at our system logging, which is an area where Linux really shines, especially in comparison to other operating systems. Logging is the process of recording system events for later perusal, giving the administrator the ability to locate intrusion attempts, and provide evidence of intrusions to law enforcement.

System and kernel messages are controlled by the syslogd daemon as well as the klogd daemon, both of which use the configuration file found in /etc/syslog.conf. Open /etc/syslog.conf in a text editor so that we can configure what we want logged, and where we want the logs to be recorded.

Inside /etc/syslog.conf rules are defined by two fields, the selector field which tells the system what to log, and the action field which tells the system where to write the logs. Info about the various selector types and priorities can be found in the syslog.conf man page. An example rule would be:

```
authpriv.notice      /var/log/messages
```

Authpriv is the message type which deals with security or authorization messages. Notice is the priority of the message. Priorities range from warning, the lowest priority, to alert, indicating a serious problem. Notice is the second lowest priority, meaning that any authpriv alerts of a priority of warning or greater will be logged. Finally, we specify where these messages should be logged, /var/log/messages in this case. The syslog.conf man page gives a complete listing of message types and priorities, and a definition of each. It should also be noted that the action field doesn't necessarily have to be a file on the local system; you could have the messages sent to the console (/dev/console), a remote machine (@192.168.10.50), or even specific users.

After opening syslog.conf you'll see that by default all system messages of priority info and notice are logged to /usr/adm/messages; all debug messages are logged to /usr/adm/debug; and all messages of priority err (error) and higher are logged to /usr/adm/syslog. You'll also notice that messages of type warn (warning) are not logged because, as the comments tell us, warning messages can slow the system to a crawl if you are running a news server.

First, lets replace all instances of /usr/adm/* with /var/log/*. For example, after *.debug we have /usr/adm/debug, which we will change to /var/log/debug. Slackware is a bit odd in that it still logs to /usr/adm by default whilst nearly every other distribution has moved to /var/log for logging. You'll recall we made a separate one gigabyte partition for /var. By using the /var/log directory to save log files, we will limit our maximum log file sizes to one gigabyte. Were we to leave logging set to /usr/adm, we could be susceptible

to a primitive Denial of Service wherein the attacker could attempt to spawn enough system messages to fill our /usr partition. If you plan on using /usr/adm rather than /var/log for log storage, you should create a separate partition for /usr/adm.

As stated previously, warn priority messages can cause problems when running a news (INN) server. However, we are not running a news server and more importantly, we want as much information as possible logged. We will enable logging of warn messages by un-commenting the *.warn line and ensuring that it points to /var/log/syslog.

Finally, we will add lines to enable logging of all kernel messages and all security messages. Kernel messages are denoted by the type kern; security messages by the type authpriv. We will add the following lines:

```
kern.*          /var/log/kernel
authpriv.*      /var/log/authpriv
```

Our finished syslog.conf will look like this:

```
# /etc/syslog.conf
# For info about the format of this file, see "man
# syslog.conf" (the BSD man page), and
#/usr/doc/sysklogd/README.linux.
#

*.=info;*.=notice               /var/log/messages
*.=debug                        /var/log/debug

# We don't log messages of level 'warn'. Why? Because if
# you're running a news site (with INN), each and every
# article processed generates a warning and a disk access.
# This slows news processing to a crawl.
# If you want to log warnings, you'll need to uncomment
# this line:
*.warn                          /var/log/syslog
*.err                           /var/log/syslog
kern.*                          /var/log/kernel
authpriv.*                      /var/log/authpriv
#
# This might work instead to log on a remote host:
# *                       @hostname
```

Now we must create all the syslog files we specified above, and give them proper (root only) permissions by issuing the following commands:

```
# touch /var/log/kernel /var/log/authpriv &&
>chmod 600 /var/log/messages /var/log/debug /var/log/syslog
/var/log/kernel /var/log/authpriv /var/log/sulog
```

This creates our two new files and sets the permissions of all our log files (including sulog, which logs su activity). After making these changes, we will reboot the system by issuing the `reboot` command as su'ed root. Rebooting should populate the log files with some information, allowing us to confirm that logging is working properly by issuing `cat /var/log/kernel|less` and verifying that kernel messages were passed to the log file. From the reboot alone, I now have over 350 lines of kernel logs. Below are the first five lines, to give you an idea of what you should see.

```
March 15 04:07:02 test syslogd 1.4.1: restart.
March 15 04:07:03 test kernel: klogd 1.4.1, log source =
/proc/kmsg started.
March 15 04:07:03 test kernel: Inspecting /boot/System.map
March 15 04:07:03 test kernel: Loaded 17020 symbols from
/boot/System.map.
March 15 04:07:03 test kernel: Symbols match kernel version
2.4.5
```

Maintaining Logs:

As you can see from the example above, log files can grow quickly and result in log files of massive size. It will be necessary not only to regularly review the log files, but also to manage them by compressing them and if at all possible, archiving them onto removable media such as a Zip Disk or Tape Backup. If disk space is an issue and you can't spare a gigabyte for logging you may want to consider writing a small bash script to compress and rotate the log files and create a cronjob to run the script on a daily or weekly basis.

Installing and Configuring Tripwire:

Finally, we will install Tripwire, which checks the integrity of important files by generating one or more message digests or "fingerprints" for later comparison. For example, you could generate and archive a fingerprint of your /etc/shadow file. A month later you can use Tripwire to generate another fingerprint and compare the two. If the values differ even though you haven't added or changed any user accounts, you'll know that the file has been altered in some way, possibly by an intruder. Tripwire is freely available from http://www.tripwire.org. Download the tarball and enter the following commands:

```
# tar xzvf tripwire-2.3-47.bin.tar.gz
# cd tripwire-2.3
```

Now open the install.cfg file in your text editor and look over the contents, making changes as necessary. I found it necessary to change my text editor path from /bin/vi to /usr/bin/vi for example. Also, towards the bottom of the file there are two options for mail methods: sendmail and SMTP. Since we are not running the sendmail service (and don't even have it installed) it is necessary to comment out the sendmail lines and uncomment the smtp lines. Now save changes and begin the installation.

```
# ./install.sh
```

After reading and accepting the terms of service, you are prompted for a site and local keyfile passphrase. Remembering the previous discussion on proper password selection, choose and enter the passphrases. You'll be asked for your passphrase a couple more times, and then will receive:

```
The installation succeeded.
```

Now it is necessary to examine, and possibly edit the Tripwire configuration file and the Tripwire policy file, found in /etc/tripwire/twcfg.txt and /etc/tripwire/twpol.txt respectively. The configuration file contains essentially the same information we saw in the install.cfg file, sans the commented out lines. The twpol.txt file on the other hand, is an example policy file made for a full installation of RedHat 7.0, and will therefore require some careful study. Although rather time-intensive, it's a good idea to at least correct the application paths and comment out references to programs not being used or not installed. After making the appropriate changes, we can configure and run tripwire:

```
# cd /usr/sbin
# ./twadmin –create-cfgfile –site-keyfile
/etc/tripwire/site.key /etc/tripwire/twcfg.txt
```

After you enter your passphrase at the prompt, twadmin will format the configuration file and exit. Now, we'll issue a similar command to update the policy file:

```
# ./twadmin –create-polfile /etc/tripwire/twpol.txt
```

Once again we are prompted for our passphrase before receiving a message that the file was written. Now we will generate our Tripwire database:

```
# ./tripwire –init
```

Unless you very carefully edited the paths in your twpol.txt file, you will likely see some errors during the database generation step, such as the following:

```
### Warning: File system error.
```

```
### Filename: /root/.amandahosts
### No such file or directory
### Continuing…
```

It really is a good idea to fix all of the policy rules that generate errors, as they will continue to appear each and every time you check file integrity with Tripwire.

Once the database has been generated we will have a complete operating system snapshot. From here on out, we can tell Tripwire to test file integrity by issuing the `./tripwire –check` command, which will scan all important files on the system and report the results as well as create a new Tripwire database in /var/lib/tripwire/report/. A sample output is included in Appendix B.

Keep in mind that if you haven't corrected all of the paths in your twpol.txt, you will likely get up to several hundred path errors. This can make the Tripwire report difficult to read. If this is the case, you might want to send the output to a text file to review later. This can be accomplished by adding a redirect to the command:

```
# ./tripwire –check >> twresults
```

For Tripwire to be effective, it will be necessary to regularly check file integrity. You may want to add a Tripwire check to your crontab so that it runs automatically at a specified time. If at all possible, it's a good idea to copy your Tripwire database to removable media by burning it to CD, copying it to a ZIP disk, or making a tape backup. If the system were compromised, a resourceful hacker could alter or delete the database, rendering Tripwire useless. Ideally, the media would then be secured in a locked carrier and kept off site for added security. Unfortunately, my hardware budget doesn't allow such a backup scheme. A more cost-efficient option would be using scp, the secure copy command, to send the database to a remote system; however this is only marginally effective if you cannot guarantee the security of the system where you are storing the file.

CHECKING CONFIGURATION:

We now have a relatively secure Slackware 8 workstation. All unneeded services have been disabled, we are generating fairly comprehensive logs for review, and Tripwire is configured to alert us to file system changes. Now is the time to simulate a few common attacks and scans to ensure that our host is secure in not just theory but practice.

Testing SSH:

First, we will attempt to SSH into the system from a remote system not included in our /etc/hosts.allow file to ensure that TCP Wrapper is properly disallowing connections from non-approved hosts.

```
# ssh dnall@192.168.20.12

ssh_exchange_identification: Connection closed by remote
host
```

This is responding as it should, denying connections from all hosts other than those specified.

Now we will attempt to login to the system from a legitimate host:

```
# ssh dnall@192.168.20.12
Sent username "dnall"
dnall@192.168.20.12's password:
   * * * * * * * * * ** * * * * * * * * * * * * * * * *
   UNAUTHORIZED USAGE OF THIS SYSTEM IS PROHIBITED BY LAW.
   THE ADMINISTRATOR RETAINS THE RIGHT TO MONITOR, EDIT,
   DELETE, OR VIEW ANY AND ALL DATA STORED ON OR PASSING
   THROUGH THIS SYSTEM. VIOLATERS WILL BE PROSECUTED TO THE
   FULLEST EXTENT OF THE LAW
   * * * * * * * * * ** * * * * * * * * * * * * * * * *
dnall@test:~$
```

As you can see, this returns our modified MOTD and verifies that SSHd is working properly.

Scanning and auditing:

Many auditing tools are available to Linux administrators, most of which are also employed by malicious hackers. In this section we will scan our system with nmap and nessus in an effort to locate open ports and possible vulnerabilities respectively. We will run these tools from local host which will provide output similar to what we would

receive were the system directly connected to the Internet. As I stated earlier, this system will reside behind a separate hardware firewall which will add an extra layer of security. If you do not have a separate firewall or if you will need the system directly connected to the Internet, it is recommended that you explore using ipfw, iptables, or ipchains to provide an added layer of security.

NMAP:

Nmap (Network Mapper) is an excellent auditing tool created by Fyodor and available for download from http://www.insecure.org/nmap/. The current version is 2.54-Beta34, and the source tarball can be directly downloaded from http://download.insecure.org/nmap/dist/nmap-2.54BETA34.tgz
After downloading the file, we will issue the following commands as su'ed root to unpack and install it.

```
# tar xzvf nmap-2.54BETA34.tgz
# cd nmap-2.54BETA34
# ./configure
# make
```

Towards the end of the make process, you'll receive an error that the Xwindows front-end will be unavailable as GTK is not installed. This is fine; we'll be using nmap from the command line anyway.

We will run nmap on our local host by issuing the following command:

```
# ./nmap 127.0.0.1
```

Here is the output:

```
Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Interesting ports on localhost (127.0.0.1):
(The 1554 ports scanned but not shown below are in state:
closed)
Port       State      Service
22/tcp     open       ssh
6000/tcp   open       X11

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

Port 22, our SSH port is open, which is as it should be, but if you have Xwindows running in another console you will also notice that it is listening on port 6000; an obvious security hole. By default X will bind itself to a port when it is launched,

providing an opportunity for malicious hackers to snoop Xwindows session data. Let's disable the port binding by exiting Xwindows and opening /usr/X11R6/bin/startx in our text editor. One of the first uncommented lines reads

```
Defaultserverargs=""
```

Which we will change to

```
Defaultserverargs="-nolisten tcp"
```

After disabling X port binding, lets restart X and run nmap again, just to make sure that SSH is the only service running:

```
# ./nmap 127.0.0.1
Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Interesting ports on localhost (127.0.0.1):
(The 1555 ports scanned but not shown below are in state:
closed)
Port       State      Service
22/tcp     open       ssh
Nmap run completed -- 1 IP address (1 host up) scanned in 1
second
```

Xwindows is no longer binding and listening on a tcp port and the only means of access to the system from a remote location is SSH, which is as it should be.

Nessus:

Nessus, available at http://www.nessus.org, is a vulnerability scanner. Nessus uses plug-ins to update its database of vulnerabilities to scan for. Getting Nessus installed and working properly can be a bit of a chore, but the end result is well worth the effort.

Begin by downloading the Nessus libraries from ftp://ftp.cis.fed.gov/pub/nessus/nessus-1.2.0/src/nessus-libraries-1.2.0.tar.gz (other mirror locations are listed on the Nessus web site). After downloading, issue the following commands as root:

```
# tar xzvf nessus-libraries-1.2.0.tar.gz
# cd nessus-libraries
# ./configure
# make
# make install
```

Now, we will download libnasl from ftp://ftp.cis.fed.gov/pub/nessus/nessus-1.2.0/src/libnasl-1.2.0.tar.gz and install it like so:

```
# tar xzvf libnasl-1.2.0.tar.gz
```

```
# cd libnasl
# ./configure
# make
# make install
```

Now we will install the core Nessus files. This will be a bit tricky as Nessus, by default, uses the GTK toolkit to provide an Xwindows interface, which we opted to not install. By default Nessus uses the GTK toolkit, which we did not install, to provide an Xwindows interface. Luckily, there is a configure option (--disable-gtk) which will allow us to compile Nessus as command line only. Download the core file from ftp://ftp.cis.fed.gov/pub/nessus/nessus-1.2.0/src/nessus-core-1.2.0.tar.gz and issue the following commands:

```
# tar xzvf nessus-core-1.2.0.tar.gz
# cd nessus-core
# ./configure --disable-gtk
# make
# make install
```

Finally, we will install the plug-in modules, found at ftp://ftp.cis.fed.gov/pub/nessus/nessus-1.2.0/src/nessus-plugins-1.2.0.tar.gz, by issuing the following commands:

```
# tar xzvf nessus-plugins-1.2.0.tar.gz
# cd nessus-plugins
# ./configure
# make
# make install
```

Now that Nessus is installed, we must perform a few more steps before we can actually start the scan. First, we must create a user:

```
# cd /sbin &&
> ./ldconfig
# cd /usr/local/sbin &&
> ./nessus-adduser
```

You will be prompted for a login first (use your regular account name, dnall in this case), then given a choice of authentication methods. Select “pass” and enter a login password. Finally you’ll be given an option to specify which hosts the newly created user may scan. Press CTRL+D to select an empty rule set, and confirm your selections.

Now that we have created a user, we must start the Nessus daemon by issuing the ./nessusd command as su’ed root from the /usr/local/sbin directory. Before we can do this

however, we must generate an SSL certificate by issuing the ./nessus-mkcert command and following the prompts. Because we will not be leaving the Nessus daemon running for extended periods of time, we can use the default choices. Once the daemon is running, switch to another terminal and log in to your user account. First we must make a target file, which specifies the hosts that we want Nessus to scan.

```
$ touch nessustarget &&
> echo "192.168.20.12" >> nessustarget
$ cat nessustarget
192.168.20.12
```

Now we can run Nessus by issuing the following command, which is explained in the man page, as well as the Nessus FAQ (http://nessus.org/doc/faq.html#Q.RUNNING.COMMANDLINE):

```
$ nessus -output-type=html_pie -config-file=.nessusrc -
batch-mode 127.0.0.1 1241 [username] [password]
nessustarget nessusresults
```

The SSL certificate information will be displayed and you will be asked to accept. The system will appear to hang for several minutes as the scan runs, after which you will be returned to the bash prompt. Log out your user account and kill nessusd by switching back to the su root terminal and pressing CTRL+C. You should now have a /nessusresults folder which you can view in a web browser. As you can see, Nessus' output is very sleek, generating html files and gif graphs.

Our Nessus scan found very little information. It reports only one "warning" and three "information" items:

```
List of open ports:

-ssh (22/tcp) (Security warnings found)
-general/udp (Security notes found)

Warning found on port ssh (22/tcp)
     The remote SSH daemon supports connections made using
     the version 1.33 and/or 1.5 of the SSH protocol.

     These protocols are not completely cryptographically
     safe so they should not be used.

     Solution:
     If you use OpenSSH, set the option 'Protocol' to '2'
```

```
     If you use SSH.com's set the option
     'Ssh1Compatibility' to 'no'

     Risk factor: Low

Information found on port ssh (22/tcp)
     Remote SSH version : SSH-1.99-OpenSSH_3.1p1

Information found on port ssh (22/tcp)
     The remote SSH daemon supports the following versions
of the SSH protocol:
     -1.33
     -1.5
     -1.99
     -2.0

Information found on port general/udp
     For your information, here is the traceroute to
192.168.20.12:
     192.168.20.12
```

The only matter of concern is the SSH warning regarding protocol versions. Because I may be connecting from locations where I am forced to use older versions of the SSH protocol I will leave this as-is. If the reader is confident that they only need version 2.0 of the SSH protocol she is encouraged to disable the older protocol versions.

The fact that Nessus reported so few alerts is not a slight against it, but rather a testament to our system's security. To generate a simple basis of comparison I also ran Nessus against a stock WindowsXP system on my network and received over twenty high risk alerts and scores more of medium and low risk.

As you can see, Nessus is an invaluable tool which scans for many different vulnerabilities and presents the output in a very readable format

ONGOING MAINTENENCE:

With our present configuration we have a very secure system. Unfortunately, security is not a static field and for the system to remain secure it is necessary to be constantly vigilant not just of our system's security, but also of security trends in general. We will now highlight some of the steps necessary to keep the system secure.

--Log Files

Our system will constantly be logging events but these logs are useless if they are not regularly reviewed for abnormal actions. It is the administrator's duty to watch for possible security violations in the log files. If necessary, you may want to write a few scripts to automatically compress, rotate, and archive your logs. There are many programs available which will assist you in maintaining and auditing your log files if you become overwhelmed, such as Psionic Logcheck, logrotate, and SWATCH.

--Patches

As new security holes are found in software, the administrator must apply appropriate patches in a timely manner. Keep in mind that poorly written patches can present new vulnerabilities of their own, presenting the need for a fairly comprehensive system audit any time a patch is installed. The same holds true for any new software that may be installed on the system.

--Tripwire

Tripwire is an invaluable tool but it will do us no good if we do not regularly compare the current fingerprint values with those in the database. You may want to create a cronjob which runs the Tripwire check or you may wish to perform the checks manually. Either way, you should be vigilant in checking and maintaining your Tripwire information.

--Scanning

Now that we have nmap and Nessus running on the system, it is important to occasionally run scans to ensure that no new possible exploits exist on the system. Keep Nessus up to date with new plug-ins as they are released and scan regularly. Anytime you install new software or apply patches is an excellent opportunity to scan for security holes. As with Tripwire, you may want to create a cronjob to periodically scan the system.

--Security Trends

It is also important to keep abreast of security trends by reading up on new alerts, advisories, and exploits. The Slackware mailing lists, the BugTraq mailing lists, and numerous websites and newsgroups are dedicated to spreading news regarding information security. It is recommended that that you consider joining some of these communities and reading the information they provide. More information and links to some of these sites can be found in the previous "Patching" section.

CONCLUSION:

We now have a working Slackware install that provides a high amount of security yet still retains its functionality as a development and day-to-day use workstation. With only a hardened SSHd running we can be fairly confident that we are immune to the vast majority of exploits.

As you can see from the steps above there is no simple procedure for *nix security. Rather, a multi-tiered approach must be used to provide true "Defense in Depth". By being mindful of log entries, new vulnerabilities, and physical security, we can keep our system up to date and secure.

BIBLIOGRAPHY:

Anonymous (SAMS Publishing). Maximum Linux Security. Indianapolis. Sams Publishing. 1999

Scambray, Joel. McClure, Stuart. Kurtz, George. Hacking Exposed Second Edition. Berkeley. Osborne/McGraw-Hill. 2001.

Parker, Tim. Linux Unleashed Third Edition. Sams Publishing. 1998.

Network Working Group. "Request for Comments: 1244". July 1990. URL: http://www.faqs.org/rfcs/rfc1244.html (15March2002)

DiPaola, Danny. "How to create a multiple partition system" URL: http://www.linuxnewbie.org/nhf/intel/installation/partition/multi_partition.html (15March2002)

SANS Institute. Securing Linux Step-By-Step. SANS Publishing. 2000.

APPENDIX A:
Software selected for installation.

Series A:
I245 – Linux 2.4.5 no SCSI (YOU NEED 1 KERNEL)
Bash1 – GNU bash-1.14.7 shell
Gpm – Cut and paste text with your mouse
Kbd – Change keyboard mappings and console fonts
Infozip – zip/unzip archive utilities
Zoneinfo – Configures your time zone

Series AP:
Diff – GNU diffutils
Ghostscr – GNU Ghostscript version 5.50a
Groff – GNU groff document formatting system
Jed – JED programmer's editor
Ksh93 – KornShell language and interactive shell
Man – Primary tool for reading online documentation
Manpages – man pages (online docs – requires groff)
Mc – The Midnight Commander file manager
Mp3 – Command-line MP3 players
Oggutils – Ogg Vorbis encoder, player, and libraries
Quota – User disk quota utilities
Rpm – Unsupported package tool
Screen – GNU screen ANSI/vt100 virtual terminal emulator
Seejpeg – An SVGAlib image viewer
Sox – Sound utilities
Sudo – Allow special users limited root access
Texinfo – GNU texinfo documentation system
Vim – Improved vi clone

Series D:
Autoconf – Gnu source autoconfig system
Automake – GNU makefile generator
Bin86 – 8086 assembler/loader
Binutils – GNU C compiler utilities
Bison - GNU Bison parser generator
Byacc – Berkely Yacc
Cvs – Concurrent Versions System
Egcs – GNU egcs-1.1.2 C compiler (for the kernel only)
Flex – Fast lexical analyzer generator
Gcc – GNU gcc-2.95.3 C/C++ compiler
Gcc_objc – GNU Objective-C compiler for gcc-2.95.3
Gdb – The GNU debugger

Gdbm – The GNU gdbm database library
Gettext – GNU internationalization support package
Glibc – Libraries for developing ELF binaries
Glocale – Internationalization support for glibc
Gmake – GNU make
Jpeg6 – JPEG image library and tools
Libgr – Various graphics libraries and tools
Libpng – Portable Network Graphics library
Libtiff – Tag Inage File format library
Libtool – GNU Libtool library support script
Linuxinc – Linux kernel include files
M4 – GNU m4 macro processor
Ncurses – CRT screen handling process
Perl – Larry Wall's system language
Pmake – Parallel make from BSD
Python – An interpreted object-oriented language
Readline – Input library with editing
Slang – S-Lang interpreted language and library
Strace – Traces program execution
Svgalib – Super-VGA Graphics Library
Termcap – GNU terminal control library
Zlib – general purpose data compression library

Series E:

Emacsbin – The base GNU Emacs 20.7 system
Emacmisc – Miscellaneous files for Emacs 20.7
Emacinfo – Info (documentation) files for Emacs

Series F:

Manyfaqs – Lots of Linux documentation
Howto – HOWTOs from the Linux Doc Project
Mini - Linux mini-HOWTOs on a variety of tasks

Series KDE:
(* denotes required package)

htdig - * Search engine required for KDE help system
kadmin – KDE system admin utilities
kdebase - * The K Desktop Environment (base package)
kdegames – A KDE games collection
kdelibs - * Libraries for KDE
kdepim - * Personal Information Management tools

kdetoys – A few desktop toys for KDE
kdeutils - * Utilities for KDE
kgraphic – Graphic programs for KDE
kmedia – Multimedia programs for KDE
knetwork – Networking programs for KDE
koffice – KDE office suite
ksupport - * Support files for KDE
qt2 - * Library for C++ GUI development

Series N:

Bitchx – BitchX Internet Relay Chat (IRC) client
Ftchmail – fetchmail – get mail from POP/IMAP/ETRN servers
Ipchains – Firewall configuration utility for 2.2
Iptables – Firewall configuration utility for 2.4
Mailx – The mailx mailer
Metamail – Metamail multimedia mail extensions
Ncftp – NcFTP file transfer utilities
Netpipes – Network pipe utilities
Netwatch – Ncurses network monitor
Lynx – Text-based World Wide Web browser
Openssh – OpenSSH Secure Shell
Openssl – OpenSSL Secure Sockets Layer toolkit
Pine – Pine menu-driven mail program
Procmail – Mail delivery/filtering utility
Tcpdump – Tool for dumping network packets
Tcpip1 – TCP/IP networking programs
Tin – The 'tin' news reader
Trn – A threaded news reader
Wget – WWW/FTP retrieval tool

Series T:
(* denotes required package)

Tetex - * teTex base package
Tex_bin - * teTex binaries

Series TCL:

Tcl – The TCL script language
Tk – The TK toolkit for TCL
Tclx – Extended Tcl
Tix – Tix widget library for TK
Expect – A tool for automating interactive applications

Series X:

Lesstif – A Motif clone
Xf86doc – Documentation for XFree86 4.1.0
Xf86html – HTML documentation for XFree86 4.1.0
Xf86prog – Libraries and headers for X programming
xfntscal – Scalable Speedo/Type1 fonts

APPENDIX B:
Sample Tripwire Check Output:

```
Performing integrity check…
Wrote report file: /var/lib/tripwire/report/test-20020502-182815.twr

Tripwire® 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Thu May 2 18:28:15
Database last updated on:     Never

================================================
Report Summary:
================================================

Host name:                    test
Host IP address:              192.168.20.12
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/test.twd
Command line used:            ./tripwire -check

================================================
Rule Summary:
================================================

------------------------------------------------
Section: Unix File System
------------------------------------------------

Rule Name               Severity Level    Added Removed     Modified
---------               --------------    ----- -------     --------
Invariant Directories   66                0     0           0
Temporary directories   33                0     0           0
*Tripwire Data Files    100               1     0           0
Critical devices        100               0     0           0
*User binaries          66                1     0           1
Tripwire binaries       100               0     0           0
Networking Programs     100               0     0           0
Shell related programs  100               0     0           0
Critical Utility sym-links
                        100               0     0           0
System boot changes     100               0     0           0
Security Control        100               0     0           0
Login Scripts           100               0     0           0
OS Utilities            100               0     0           0
Root config files       100               0     0           0

Total objects scanned:  14006
Total violations found: 3
```

```
=================================================
Object Summary:
=================================================

-------------------------------------------------
# Section: Unix File System
-------------------------------------------------

-------------------------------------------------
Rule Name: User binarires (/usr/sbin)
Severity Level:   66
-------------------------------------------------

Added:
"/usr/sbin/twresults"

Modified:
"/usr/sbin"

-------------------------------------------------
Rule Name: Tripwire Data Files (/var/lib/tripwire)
Severity Level:   100
-------------------------------------------------

Added:
"/var/lib/tripwire/test.twd"

=================================================
Error Report:
=================================================

-------------------------------------------------
Section: Unix File System
-------------------------------------------------

1.    File system error.
      Filename: /usr/Kerberos/bin/rsh
      No such file or directory
2.    File system error.
      Filename: /sbin/mkraid
      No such file or directory
3.    File system error.
      Filename: /sbin/ctrlaltdel
      No such file or directory
4.    File system error.
      Filename: /sbin/dhcpd
      No such file or directory
5.    File system error.
      Filename: /etc/httpd.conf
      No such file or directory

-------------------------------------------------
*** End of Report ***
```

```
Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a
registered trademark of Tripwire, Inc. This software somes with
ABSOLUTELY NO WARRANTY; for details use –version. This is free software
which may be redistributed or modified only under certain conditions;
see COPYING for details.
All rights reserved.
Integrity check complete.
```

Last Updated: May 26th, 2017

Upcoming Training

SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**