A Survival Guide for Linux Security

A consensus document by security professionals from 46 commercial, educational, and government institutions.

SECURING LINUX

STEP-BY-STEP

VERSION 1.0

THE SANS INSTITUTE

One of the great sources of productivity and effectiveness in the community of computer professionals is the willingness of active practitioners to take time from their busy lives to share some of the lessons they have learned and the techniques they have perfected. Much of the sharing takes place through online news groups, through web postings, and through presentations at technical meetings, and those who are able to take the time to scan the newsgroups, surf the web, and attend the meetings often gain invaluable information from those interactions.

SANS' Step by Step series raises information sharing to a new level in which experts share techniques they have found to be effective. The SANS Institute integrates the techniques into a step-by-step plan and then subjects the plan, in detail, to the close scrutiny of other experts. The process continues until consensus is reached. This is a difficult undertaking. A large number of people spend a great deal of time making sure the information is both useful and correct.

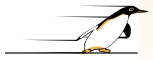
INTRODUCTION

From a small, collaborative effort headed by a University of Helsinki student named Linus Torvalds, Linux has grown into a global phenomenon spurring new industries for distribution, training, and support. It is now the only operating system other than Windows NT that is gaining market share in corporate Information Technology infrastructures. Distributors are actively marketing easier to install, easier to use Linux systems and making real inroads onto the desktops of home, corporate, government, and educational users. By some estimates there are nearly 8 million Linux users worldwide.

Linux is an "Open Source" operating system. The source code for the kernel and system utilities is available for download, inspection, and modification. This is a double-edged sword: system developers and ordinary users alike have access to the source code so bugs are found and fixed more quickly; but system crackers have access to the code as well, and they can use this knowledge to develop exploits more rapidly and reliably. This does not make Linux less secure than its proprietary competition. On the contrary, bugs are discovered faster in an open environment, and patches and updates are issued for Linux system software very quickly. Unfortunately, most users install Linux from CD-ROM media that quite often contains vulnerable programs by the time the ink dries on the label. Another unfortunate aspect of installing commercial Linux distributions is that, for ease of use, these Linux systems are configured with most, if not all, network services running immediately after the computer is booted up, and without any access controls in place. For example, for years all Linux distributions have shipped with TCP wrappers in place, but the `/etc/hosts.allow` and `/etc/hosts.deny` files are empty, meaning that anyone on the Internet can connect to TCP wrapped services.

This guide is intended for the novice home user and the experienced systems administrator alike. It covers the installation and operation of Linux in two basic modes of operation: as a workstation and as a server. It does not cover configuring Linux for some of the other special-purpose functions that it performs so well, such as routers, firewalls, parallel processing, and so forth. The examples and instructions are based on the Red Hat version 6.0 release. Red Hat was chosen because it has the largest share of the Linux market, and version 6.0 was chosen because it includes the latest stable release of the Linux kernel, system libraries, utilities, etc. However, the concepts, advice, and procedures in this guide should translate rather easily to other distributions. You may have to explore your system a little to find configuration files that are in different directories, and to determine which versions of the software packages have been installed, but the exploration itself can be a good instructional tool.

This guide takes you, the reader, through the installation process then splits into separate steps for securing a workstation setup and a server setup. The guide discusses basic packet firewalls in terms of protecting services on a single local computer. Finally, the guide discusses a few useful tools for monitoring and testing the security of your system. We try to follow the principle of "defense in depth." No one step is a silver bullet against system attacks, but taken as a whole, they build multiple layers of defense that make life just that much harder for "script kiddies" and dedicated computer criminals.



INTRODUCTION

We try to explain, as much as possible, the options and ramifications of securing a Linux box. However, this guide is not a text on computer security. Whenever possible, pointers to other documents and resources are included in the text or in Appendix A. We encourage you to study these other resources before setting up your Linux box, and to keep abreast of the latest information from the distributor, the SANS Institute, and other cited security references.

By convention, commands executed by the root user are preceded with the command-line prompt "`[root]#`". In the body of the text, system commands and file names are in `Courier fixed-space font`. Command sequences and file contents are separate from the text, also in Courier fixed-space font. References to manual pages are in the traditional style, page (section), e.g. `inetd.conf(5)`. To read the manual page in Red Hat Linux, and most other distributions, execute the command:

```
[root]# man 5 inetd.conf
```

IMPORTANT:
Updates will be issued whenever a change in these steps is required, and new versions will be published periodically. Please email info@sans.org with the subject <Linux Security> to subscribe to the monthly Network Security Digest containing news of new threats and solutions and announcements of updates. There is no charge.

This edition was guided and edited by:
Lee E. Brotzman, Allied Technology Group, Inc. and David A. Ranch, Trinity Designs

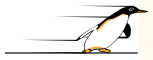
The SANS Institute enthusiastically applauds the work of these professionals and their willingness to share the lessons they have learned and the techniques they use.

Tyler J. Allison, AboveNet Communications, Inc.
Ofir Arkin, LinuxPowered.com
Scott Barker, MostlyLinux, Inc.
Mario Biron, Geonetix Technologies Inc.
Daniel T. Brown, Air Force Research Laboratory, Rome Research Site, Information Assurance Office
David Brumley, Stanford University Network Security Team
Richard Caasi, Science Applications International Corporation
Ian C. Campbell, State University of New York at Albany
John Coleman, Yale University Library Systems
Andrew Cormack, JANET-CERT
Patrick Darden, Athens Regional Medical Center
Clement Dupuis, UniGlobal, Montreal Canada
John F. Feist, Space and Naval Warfare Center
Robin Felix, R. L. Phillips Group
William James Hudson, Robert Mann Packaging, Inc.
Det. Ted Ipsen, Seattle Police Department
Roy Kidder, Corecomm Communications
Alexander Kourakos, Biz Net Technologies
Chet Kress, APAC Customer Services
Loren E. Heal, University of Illinois at Urbana-Champaign
John Lampe, EDMT Technologies
Jonathan Lasser, University of Maryland, Baltimore County (UMBC)
Bill Lavalette, Network Disaster Recovery Systems
Manuel Lopez, Universidad Autonoma de Baja California
Chandrashekhar Marathe, Lucent Technologies India PRC, Bangalore
Rob Marchand, Array Systems Computing
Mike Marney, Robins Air Force Base
John E. Meister, Jr., Intermec Technologies Corporation
Shane B. Milburn, Science Systems and Applications, Inc.
P. Larry Nelson, University of Illinois at Urbana-Champaign
Stephen Northcutt, The SANS Institute
Davi Ottenheimer, M & I Data Services
Jari Pirhonen, AtBusiness Communications Ltd.
Jesse I. Pollard, II, Logicon Information Systems & Services
Patrick O.C. Ramseier, Pilot Network Services, Inc.
Dave Remien, SCIENTECH, Inc.
Andy Poutt, Concept Computing
David Saunders, University of Virginia
Kurt Seifried, SecurityPortal.com and Seifried.org
J. J. Shardlow, Tertio, Ltd.
Andres J Silva III, Collective Technologies
Derek Simmel, Software Engineering Institute, Carnegie Mellon University
Len Smith, University of Michigan, College of LSA
Aurobindo Sundaram, Schlumberger IT
Robert Thomas, U.S. Census Bureau
Bob Todd, Advanced Research Corporation

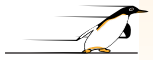
CONTENTS

CONTENTS

CONTENTS

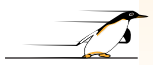


CONTENTS



CONTENTS

CONTENTS



CONTENTS

STEP 1.1 DETERMINE THE SECURITY NEEDS

System security on any operating system needs to be thought out before hand. Security-conscious home computer users and corporations alike need to determine the following before installing:

1. What are you trying to protect? Confidential data (medical records, credit card numbers, etc.), access to essential services (DNS, mail, etc.), computing resources (user shell accounts, software development tools, etc.), and so forth.
2. To what extent should security be implemented before the costs outweigh the value of the original data and/or services being protected?

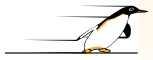
You will need to determine what this machine's main purpose will be. Once determined, try to stick to your original plan. The two primary purposes covered in this guide are:

1. Workstation: X-Windows, user applications like Web browsers, e-mail, word processing, spreadsheets, databases, etc. Workstations don't need to offer services like TELNET, FTP, HTTP, etc. Thus, these services should be shutdown or removed altogether.
2. Server: No X-windows, running network services such as NFS, Samba, HTTP, FTP, SMTP, DNS, shell accounts, firewalls, etc.

Servers should be configured to run as few services as possible. This will offer the administrator better security, fault tolerance, and the ability to scale performance to the services that need it. However, the additional hardware and administrative cost must be considered.

Step 1.1.1 Define security policies

This step is intended more for security professionals and corporate Information Technology managers, but it will be useful to private users concerned about security as well. Corporate and government organizations need to develop, document, and enforce a security and appropriate use policy. The policy should be applied to both internal and external users. Users should learn about the policy through written communication and should sign a document agreeing to the policy. The main reason for a well-documented policy and implementation is that when a security breach or misuse of your systems occur, you have binding legal documents to pursue the people responsible. Without policies and corresponding documentation, enforcement of the policy may be challenged legally.



STEP 1.2 PHYSICALLY SECURE THE COMPUTER

Depending on your security needs, you may have to physically secure the box. Physical security for home users isn't much of an issue, but if you're an ISP that co-locates hardware at another facility, you should think about locking the server(s) up.

- If possible, put the server in a well-ventilated and locked room. Know who has the key to that room.
- Use computer cases that have drive bays and keyboards that can be locked. This ensures that people can't insert floppies and CDs or try to hack away at the console's keyboard.

STEP 1.3 BIOS SECURITY: PASSWORD PROTECTION, LIMITING REBOOTS

Most modern computer BIOSes allow the restriction of bootable devices and password protection for BIOS settings. Different systems use different keystrokes to enter into the BIOS setup but the common ones are the "Del" key for AMI BIOS, "Ctrl + Esc" for Phoenix BIOS, and the "F10" key for Compaq BIOS.

Step 1.3.1. Disable "AUTO" settings

Disable the "AUTO" BIOS settings for options like hard disks, PnP IRQ settings, etc. IDE hard drives configured for "AUTO" configuration can be recognized incorrectly by Linux. System administrators may see file system problems from incorrect BIOS AUTO settings and try to fix the issues with `e2fsck`. This can result in a corrupt file system simply because the IDE "AUTO" setting changed the hard drive translation from the Logical Block Addressing (LBA) mode to the "LARGE" sector layout. Manually configuring the hard drive parameters and other settings ensures that your machine is more reliable and boots faster.

Step 1.3.2. Disable booting from removable media

Enable booting only from the primary hard drive (C:). Do not allow booting from floppy, ZIP, CD-ROM, or any other form of removable media. For medium- and high-security needs, you may want to remove the floppy drive entirely. It should also be noted that some BIOSes allow the floppy drive to be set to READ ONLY, although this will not in and of itself prevent booting the system from a floppy.

Step 1.3.3. Set a BIOS password

Set a BIOS password to ensure that a rogue user cannot change the system's CMOS settings. Use a password with both upper and lower case letters and numbers. See Step 2.5 below for suggestions on selecting strong passwords. There are BIOS password cracking programs available on the Internet so do not use the same password for the BIOS setup to protect LILO (Step 2.6.3) or the root or other user accounts (Step 2.5.2) in the Linux operating system.

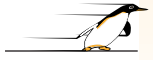
Step 1.3.4. SCSI BIOS setups

Some SCSI controllers have their own BIOS. For example, Adaptec controllers use the Control-A keyboard sequence to enter the SCSI BIOS program. Be sure that you disable the ability to boot off other SCSI media such as CD-ROMs. As of the latest version of Adaptec's SCSI BIOS, there is no way to password protect the SCSI BIOS setup. So if a user can reboot the computer, go into the SCSI BIOS, and enable CD-ROM booting. The only thing keeping them out of your system is a locked CD-ROM cage.

Step 1.3.5. Document BIOS settings

Document the BIOS settings for hardware and system support. In case of a system failure or the administrator is absent, documentation of the hard drive geometry, any non-default system settings, etc., will record the basic initial settings for this system. Place this documentation along with the other hardware manuals in a secure place.

Once your security needs are understood and the security policy documented, it's time to install the Linux operating system. Linux offers a wide array of distributions to choose from and each one has its specific pros and cons. Though it is beyond the scope of this document, we highly recommend you give your choice of Linux distribution some thought as it will impact future administration and ease of use. For this guide, we chose Red Hat 6.0 primarily because of Red Hat's market share and our goal to cover as many users as possible. If you are going to use another distribution or an earlier version of Red Hat, you should be aware of the differences in system administration and security-related configuration. For example, older versions of Red Hat do not use shadow passwords by default, system files may be in different directories, Slackware does not use the SYSV init style startup, etc. Whichever distribution you choose, upgrade to the latest version to get all the benefits of better security, functionality, and performance.



STEP 2
Install Linux

STEP 2.1 DISCONNECT THE MACHINE FROM THE NETWORK

Most Linux distributions bring up their network interfaces during installation even though there is minimal system security in place. In light of this, the best security practice is to disconnect the network card before starting the installation. If you are installing from a network server, make sure that the network you are using is secure while the installation is running. If you are installing from a public FTP site on the Internet, be careful, since your system might be compromised before you ever realize it.

STEP 2.2 SELECT INSTALLATION CLASS: WORKSTATION, SERVER, OR CUSTOM

The Red Hat installation program allows for selecting from one of these three choices. The "Workstation" and "Server" selections do automatic partitioning of the system's hard disks based upon percentages of the overall hard disk size. In addition, these two settings install many software packages that will probably never be used by you or your users. Use the "Custom" installation option for maximum control over the installation process, regardless of whether you are setting up a workstation or server.

STEP 2.3 DEFINE PARTITIONS

Partitioning is a fairly religious debate among UNIX administrators. Why? The theory is that the more partitions a system has, the more reliable it will be. For example, if a partition becomes corrupt or a denial of service attack fills a partition with log messages, the problem is isolated to only that one partition.

It is generally agreed that Linux Workstations only need three partitions: the root partition, "/"; the user partition, "/home"; and a swap partition for virtual memory. Partition schemes for servers depend on the type of service. The general rule for determining the size of a swap partition is to allocate two times the amount of physical memory. For example, a computer with 64 MB of RAM would allocate at least 128 MB of swap space. Systems with limited RAM should allocate relatively more swap space, systems with a large amount of RAM should allocate relatively less. Use common sense. A system with 1GB of RAM doesn't need 2GB of swap. A system with 32 MB of RAM or less should be upgraded with more memory first. At today's prices for physical memory, adding RAM is the most cost-effective upgrade for your system. Administrators of Linux machines running the older 2.0 version of the kernel should note that this version allows for only 128 MB swap partitions. For more swap space, multiple partitions are required. The latest releases of most Linux distributions, including Red Hat 6.0, use the newer 2.2 version of the kernel, which does not have this restriction.

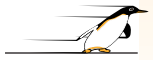
Step 2.3.1 Define Workstation partitions

Typical workstation installations with the X-Window system, window managers, development tools (if necessary), home office applications (if necessary), Web browsers, and core system utilities can consume up to 700-800 MB of disk space. A minimum root partition of 1 GB will leave room for system logs, spooled email messages, and other housekeeping data. Consider even more space for the root partition if the space is available and the requirement for space on the user's (/home) partition is not that great. For workstations that will make extensive use of logging, consider a separate, relatively large partition for /var.

As an example: a mid-range PC with 64 MB of RAM and a 4 GB hard drive could be partitioned like so:

```
/        1800 MB
swap      200 MB
/home    2000 MB
```

Note that if your workstation is going to "dual-boot" Linux and another operating system, the Linux root partition needs to be within the first 1024 cylinders of the boot disk. This is because LILO, the Linux Loader, is limited to using the BIOS to access the disk and the BIOS can not read past 1024 cylinders. See the Installation HOWTO cited in Appendix A for more information. Of course, your mileage may vary but give this topic some serious thought.



Step 2.3.2 Define Server partitions

Disk partitioning for servers is a very different animal. In general, **all** servers benefit from a separate, relatively large partition for `/var`, for logging, configuration, and temporary space. Additional factors that need to be considered depend on the purpose for the server:

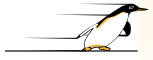
NNTP: News servers require very large amounts of spool space, so a majority of disk space should be reserved for NNTP use only. Even better, put the news spool on its own drive(s) for increased disk performance. Red Hat puts NNTP spool files in `/var/spool/news`.

HTTP: Web servers require disk space for the site's main Web pages as well as users' individual Web pages (if allowed). Web servers should leave plenty of space for HTTP log files which can consume quite a bit of space depending on how logging is configured, how long the logs are retained, etc. Red Hat puts the main Web pages in `/home/httpd` and typically, individual user Web pages go in the user's own home directory under `/home` in the `public_html` directory. Red Hat puts HTTP log files in `/var/log/httpd`.

FTP: FTP servers can require anything from marginal disk space to very large disk space depending on how they are used. Note that if you are going to allow incoming FTP services, there should be enough disk space, or even a dedicated disk, to allow for large sets of files to be uploaded. The main FTP directory for Red Hat is located in `/home/ftp` though users can "cd" into their home directory to get or put files. Red Hat puts FTP log files in `/var/log/xferlog`.

NFS/Samba: File servers typically require a lot of disk space unless the data is directly mounted on CDROM changers, other external jukeboxes or RAID disks. These systems should be able to scale with the needs of the users. Samba typically uses the user's home directory in `/home/*` to put and get files but NFS can be configured many different ways.

E-mail: Mail servers can require large amounts of spool space for storing incoming and outgoing messages, file attachments, etc. Red Hat puts incoming mail in `/var/spool/mail` in files uniquely named for each mail user account. Outgoing mail is stored in `/var/spool/mqueue`.



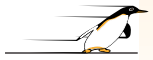
Syslog: A centralized logging host collects and stores syslog messages from other computers in the organization. As you can imagine, logging hosts need large amounts of disk space reserved for the `/var/log` directory. Good practice is to place `/var/log` on a large, fast SCSI device. Logging hosts should not run any other services at all, and should be very secure, so the amount of disk space required for the system software is smaller than for other servers.

Shell: Unix servers that provide shell accounts should have separate partitions for `/tmp` and `/var/log`. This will reduce the impact of classic denial of service attacks that fill the `/tmp` directory with files or create a process that quickly fills the system logs until the root partition is full. Once the root partition is full or the system cannot write any more log files, the system will usually crash.

We recommend that servers have separate partitions for `/var` (or even separate partitions for `/var/log` and `/var/spool`), `/tmp`, the root partition "/", and the user partition `/home`, at a minimum. Some administrators might also consider creating partitions for `/usr` which contains the majority of the Linux Operating System, and `/usr/local` for optional user-installed applications and tools. Some distributions store large system packages like KDE, Netscape and DBMS systems in the `/opt` directory tree, so you might consider a separate partition for that, too. The relative sizes of these partitions are solely at the discretion of the administrator depending on the purpose of the server and your own personal experience.

As an example: a high-end PC with 256MB of RAM and (2) 9GB hard drives Primarily used for NNTP news services could be partitioned like so:

```
First hard disk     /                500MB
First hard disk     /usr            1000MB
First hard disk     /usr/local      1500MB
First hard disk     /tmp            2000MB
First hard disk     /var/spool/news 4000MB

Second hard disk    swap             500MB
Second hard disk    /home           5500MB
Second hard disk    /var            3000MB
```



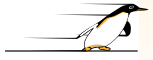
Step 2.3.3. Document the partition scheme

Once the system has been partitioned for your specific environment, document the partition layout. Then, if a partition table is somehow corrupted or deleted, recovering the table will be easier. To document the various tables, run fdisk on every one of your physical disks (hda, sda, etc) and enter "p" to print the table. Copy the results onto paper and store in a safe place for later reference. For example:

```
[root]# fdisk /dev/hda

Using /dev/hda as default device!

Command (m for help): p

disk /dev/hda: 255 heads, 63 sectors, 2193 cylinders
Units = cylinders of 16065 * 512 bytes

Device     Boot   Start     End      Blocks    Id    System
/dev/hda1    *        1    1913    15366141    83    Linux
/dev/hda2          1914    1946      265072+   82    Linux swap
/dev/hda3          1947    2193     1984027+   83    Linux

Command (m for help): q
```

STEP 2.4 SELECT PACKAGES TO INSTALL

Like setting disk partitions, the packages chosen for installation depend on the use of the system. The best way to get through this section is to know your users and how the system will be used. Determine what will and won't be done on the system and choose the software packages accordingly. Carefully go through each section of the installer and hit "F1" whenever you don't know what a given package is or if you aren't sure if you need it or not. When in doubt, leave it out. It's easy enough to add the package later as the need arises. If you don't select a package that Red Hat needs for some other tool, it will warn you about any software dependencies before the installation process actually begins.



Step 2.4.1. Workstation packages

Workstation users usually can fit into one of three categories:

1. Business. The system is used for business purposes like word processing, spreadsheets, E-mail, and WWW browsing. For these systems, there is no need to install services like NFS or HTTP servers, or compilers like GCC and EGCS.
2. Software development. For systems like this, package selection depends on the type of development. A Web developer may need a Web server, PHP, and Perl. Software developers may need GCC or EGCS, G++, Python, Perl, etc., and documentation tools like TeX and Ghostscript.
3. General Purpose. This category probably fits the needs of most home users. Business tools, development tools, multimedia players, games, graphical utilities, are all loaded on the system. The general-purpose user doesn't want to be limited to the fact that they don't have a Web server or GCC installed on their machine. Though this kind of installation is the most convenient for the end user, it makes the maintenance and security of the Linux machine more difficult.

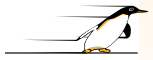
Step 2.4.2. Server packages

Servers are configured for utility, performance, and security. If your budget permits, we recommend you put information services such as WWW and FTP, file and print services such as NFS and Samba, DNS services, email services, and user shell account services on different servers. For example, an information server would only install the base system, FTP and WWW services, Perl, PHP, and nothing else. If you need to have both information services and file services running on the same machine, so be it, but try to minimize the installation of any other unnecessary software.

Step 2.4.3. Let the installation proceed

Once you have selected all the packages you want installed on this system, the Red Hat installer will figure out the dependencies between the packages you selected. It will notify you of any additional packages that you need to install to properly support the selected packages. At this point, you can allow the installer to include the dependant packages or remove the original selection from the installation list entirely.

Finally the installer will load the selected packages and prompt you for any additional configuration settings, such as network cards, mice, video cards, etc. Covering these steps is beyond the scope of this guide but it is provided in the documentation supplied with the shrink-wrap versions of the distributions or on your Linux distribution's web site.



Step 2.5.1. Shadow Passwords with MD5 hashing

One of the final installation options that Red Hat prompts the user with is whether to use "shadow" passwords with or without MD5 cryptographic hashes. Traditionally, users' UNIX passwords were encrypted with the "crypt" algorithm and stored in the world-readable file, `/etc/passwd`. This makes it easy to perform dictionary attacks using tools such as Crack and its cousins. Shadow passwords are stored in `/etc/shadow`, readable only by `root`, which makes it much harder to obtain the encrypted passwords for a dictionary attack. Use the MD5 cryptographic hash to add another layer of security to the shadow password system. It is very important to keep `/etc/shadow` secure and away from prying eyes.

Step 2.5.2. Set passwords for root and all user accounts

One of the last steps before the system reboots is setting the root password. Linux passwords are case sensitive and the installer will recommend that a root password be a combination of UPPER and lower case letters, digits, and special characters including:

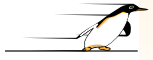
```
[ '~!@#$%^&*()-_=+{[]}\|'";:,<.>/?  ]
```

The password should **not** be based on any personal information such as name, company, Social Security number, birthday, license plate number, etc. A determined attacker can find out a surprising amount of personal information about users, especially in the case of an insider attack.

Never use a common word that would be found in any dictionary in any language, like the English word "ridge". Using digits to substitute for letters, for example "r1dg3", does not help, since common password cracking tools take this trick into account.

So what is a good password? Be unique and do combinations; think of a phrase and make an acronym (don't use an acronym from your everyday work, though). For example, a good password can be constructed from the names of the computer science algorithms for B+ and Patricia trees, "`B+paTs.!!`". Be creative but use something you can remember!

It is an excellent practice to teach your fellow system administrators and users alike how to choose strong passwords. With strong passwords that are changed on a frequent basis, systems are much more difficult to break into.



Step 2.6.1. Create a boot diskette

One step that many Linux users skip is the creation of an emergency boot diskette. The Red Hat installer creates a bootable diskette with a Linux kernel with all of the specific hardware and configuration support for your computer. Though creating a boot disk isn't a required step, eventually, a system emergency will come up you'll be happy you created it. Make sure the boot diskette is properly labeled and stored in a secure place like a tape cabinet in a computer room or locked desk drawer.

Although we strongly suggest you create the emergency diskette **now**, if you decide to do it later, in Red Hat Linux the script `/sbin/mkbootdisk` will create the disk for you. See the man page `mkbootdisk(8)` for more information.

Step 2.6.2. Tighten up settings in `/etc/inittab`

The file `/etc/inittab` defines the boot behavior of the SYSV `init` process, process number 1 (see Step 3.3 for additional information about SYSV init startup scripts). After the first reboot, edit `/etc/inittab` to disable rebooting from the console with the "Control + Alt + Del" key sequence, and to require entering the root password to enter single user mode.

To disable the "Control + Alt + Del" key sequence, change the line in `/etc/inittab` file that reads:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

to read:

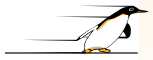
```
#ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

To require entering the root password when you enter single user mode, add the following line to `/etc/inittab` after the entry for `si::sysint...`:

```
~~:S:wait:/sbin/sulogin
```

To make these changes take effect immediately, type in the following command:

```
[root]# init q
```



Step 2.6.3. Password protect LILO boots

The Linux Loader (LILO) is the primary mechanism for booting Linux. If the physical security of the Linux machine can not be assured, password protect the LILO prompt in addition to requiring the root password to enter single user mode. Note that password protection of the LILO prompt may interfere with the automatic restart of a system after a power failure or system crash. Administrators of mission-critical servers in controlled computer room environments may not want this feature.

To password protect the LILO prompt, edit `/etc/lilo.conf` and add the following after the `prompt` line:

```
password = your-password
restricted
```

Replace "`your-password`" a strong password as shown in Step 2.5.2. Save `/etc/lilo.conf` and change the permissions of that file since the LILO password is saved in clear text:

```
[root]# chmod 600 /etc/lilo.conf
```

Make the changes take effect:

```
[root]# /sbin/lilo
```

With these changes in place, the system will boot just has as it has before, but if you give LILO any command line options at the LILO prompt, such as `single` to boot in single-user mode, it will ask for a password before proceeding.

STEP 2.7 SET SYSTEM ACCESS SECURITY POLICIES

Most Linux distributions allow for configuring which users are allowed to login on the system, at which times, on which date, and through which services (TELNET, FTP, etc). Most Linux distributions are configured correctly, if somewhat loosely, out of the box, but it is important to confirm all of these settings.

Step 2.7.1. Check that remote root logins are disabled for TELNET

The `/etc/securetty` file contains a list of all TTY interfaces that allow root logins. In Red Hat Linux, and most distributions, this file by default contains **only** the console TTYs (`tty1, tty2, ... tty8`, inclusive). If remote users require root privileges on this system, they can simply login to their own account and use the "`su -`" command to become root.

Step 2.7.2. Check that remote root logins are disabled for FTP

The `/etc/ftpusers` file contains a list of all user accounts that are not allowed to login via FTP. Make sure that the root account and all additional system daemon accounts are listed in this file. You can also enter account names for other user accounts that you do not want to access the machine by FTP.

Step 2.7.3. Configure the system accounts that can/cannot log into the system

The file `/etc/security/access.conf` contains login parameters for all accounts on the system. You can configure which users can login through which service in a manner similar to TCP wrappers, as described in Section 3.2. To disable all console logins except for the root account and the administrator, put the following in `/etc/security/access.conf`:

```
-:ALL EXCEPT root admin :console
```

Step 2.7.4. Configure the system groups that can/cannot use specific resources

The file `/etc/security/group.conf` defines which users can access files with specific group permissions.

The file `/etc/security/limits.conf` defines system resource limits (CPU time, disk space, etc) for specific users or groups.

The file `/etc/security/times.conf` defines the times of day that specific users are allowed to access the system. Ordinary users can be restricted to logging in during normal business hours through specific network services.

STEP 2.8 CONFIGURE LOGGING

Red Hat and most other distributions use the `sysklogd` package for generating system logs. This package contains two log daemons, `syslogd` and `klogd`. The kernel log daemon, `klogd`, takes care of logging kernel messages and is configured through `/etc/syslog.conf`, just like `syslogd`, and the two work in conjunction with each other seamlessly. Therefore, when we refer to `syslogd`, we are also talking about `klogd`. See the man page `sysklogd(8)` for more information

The syslog daemon parses and separates log messages based on a system of "Facility" and "Level." The stock syslog daemon has some security problems, most notably there is no built-in way to tell if the logs have been tampered with. There are several alternative log daemons that add features such as digital signatures for log files to detect tampering, and encrypted network communications for secure remote logging. See Appendix A for a list of some of these alternative log daemons.

Step 2.8.1. Optimize SYSLOG settings

The default logging for Red Hat Linux is good but it can be made better. To improve it, edit the syslog configuration file, `/etc/syslog.conf`, and add the following lines:

```
*.warn;*.err     /var/log/syslog
kern.*           /var/log/kernel
```

Note: the separators between the syslog facility directives in the first column and the syslog files in the second column ***must*** be TABs. If they are SPACES, the syslog daemon will not load properly.

Step 2.8.2. Configure real-time logging to VTYs

The real-time display of system logs on VTY 7 and 8 (the Alt-F7 and Alt-F8 screens) is very useful. To do this, append the following to `/etc/syslog.conf`:

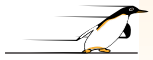
```
*.info;mail.none;authpriv.none     /dev/tty7
authpriv.*              /dev/tty7
*.warn;*.err            /dev/tty7
kern.*                  /dev/tty7
mail.*                  /dev/tty8
```

Once `/etc/syslog.conf` has been updated, you will need to create the new syslog files and fix their permissions:

```
[root]# touch /var/log/syslog /var/log/kernel
[root]# chmod 700 /var/log/syslog /var/log/kernel
```

To get syslog to immediately use the new configuration file, issue the following command:

```
[root]# killall -HUP syslogd
```



Step 2.8.3. Configure log rotation

Red Hat Linux, and some other distributions, uses the `logrotate` tool to keep log files to manageable sizes and retain them for a specified period of time. See the manual page `logrotate(8)` for more information about the command and the configuration file.

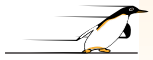
To tune the default Red Hat log rotation configuration and rotate the newly created `syslog` and `kernel` logs in Step 2.8.1, edit `/etc/logrotate.d/syslog` and add the following:

```
/var/log/kernel {
    compress
    postrotate
        /usr/bin/killall -9 klogd
        /usr/sbin/klogd &
    endscript
}

/var/log/syslog {
    compress
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
```

Next, edit `/etc/logrotate.conf` to modify the log rotation behavior to meet your specific needs. In the following example, `logrotate` will rotate logs weekly, keep the last four weeks of logs before it deletes them, compress the rotated logs, and **not** rotate the utmp and wtmp files:

```
# see "man logrotate" for details
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4

# send errors to root
errors root
# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
compress

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d

## no packages own lastlog or wtmp — we'll rotate them here
#/var/log/wtmp {
#    compress
#    monthly
#    rotate 1
#}

#/var/log/lastlog {
#    compress
#    monthly
#    rotate 1
#}
```

Once these changes are done, they will be automatically executed from the crontab file `/etc/cron.daily/logrotate` once a day (the default setting of Red Hat 6.0).



Step 2.8.4. Configure remote logging

In an organization of any significant size, using a centralized logging host has a lot of benefits. From a security point of view, remote logging is important because one of the first things that an attacker will do after gaining root access to a machine is erase or modify the system logs to cover his tracks. If copies of those log messages are stored on another, highly secured system, administrators may still be able to reconstruct what happened. If the logs are lost, important forensic evidence from the break-in is lost with them.

To send syslog messages to a central logging host, edit `/etc/syslog.conf` and add lines for the facilities and logging levels dictated by the security policy of your organization. For example, to send all warning and error messages, and all authentication facility messages to the central (logging host), called loghost add the following:

```
*.warn;*.err            @loghost
authpriv.*;auth.*       @loghost
```

Step 2.8.5. Synchronize system clock with log server

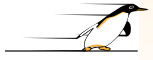
Hardware clocks in PC-class computers are notoriously inaccurate. When many workstations are feeding log messages to a central server, it is important that their clocks are synchronized, so that the time stamps on the log messages are accurate. In the event of a network-wide scan or attack, synchronized log file time stamps will make it easier to build a timeline of the attack. The Network Time Protocol was developed to accurately synchronize the system clocks of a network of computers. An easy way to set the system clock once each hour is to set up a cron job to run `ntpdate` once an hour. Put the following commands into the file `/etc/cron.hourly/set-ntp`:

```
/usr/sbin/ntpdate -bu -t 3 loghost.example.org
/sbin/hwclock --systohc
```

The “b” option sets the time now, rather than staggering it, “u” says to use an unprivileged port, and “t 3” set sets the timeout to 3 seconds, rather than the default 1 second. The second line sets the hardware clock CMOS value to the current system time. Once the file has been saved, set the permissions:

```
[root]# chmod 700 /etc/cron.hourly/set-ntp
```

Of course, this assumes that loghost.example.org is running the `xntpd` NTP daemon. The xntp3 RPM package includes extensive documentation in HTML format in the `/usr/doc/xntp3*` directory. Also see http://www.eecis.udel.edu/~ntp/ for the latest information about NTP and a list of public primary and secondary time servers.



Securing workstations is primarily a process of eliminating network services, and severely limiting access to the remainder. A typical workstation operates on the client end of client-server network communications. There is almost never a need to provide a service. On the remote chance that a service is needed on a workstation, it should be offered only to a small, select group of other workstations or servers on the local area network. In Step 3, we concentrate on removing services, then offer some advice on how to increase the security of selected services if they are absolutely necessary for the operation of the workstation.

STEP 3.1 DISABLE INTERNET DAEMON SERVICES

The Internet daemon, `inetd`, controls access to network services that only start up when needed, such as telnet and ftp. A typical workstation has no need to provide the services in the Internet daemon configuration file, `/etc/inetd.conf`. If you need to access the workstation remotely over the network, consider installing Secure Shell (SSH; see Step 4.2 below). See the manual page `inetd.conf(8)` for a complete description of the format of entries in `/etc/inetd.conf`.

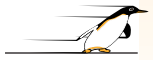
Step 3.1.1. Edit `/etc/inetd.conf` and comment out all services

See Appendix B for a copy of the default `/etc/inetd.conf` installed with Red Hat Version 6.0. The services that are open are those that do not have a hash symbol "#" in column 1. Edit these lines and comment them out by putting the comment character, "#", in the first column. Save the edited file and make the `inetd` process reload the configuration file by sending it the hangup signal:

```
[root]# killall -HUP inetd
```

If you don't comment out any other services, at least make sure that the shell and login services are disabled. The Berkeley "r" programs `rsh` and `rlogin` have a long history of abuse and security problems. Even with the services disabled, you should check to make sure that no user account has a `.rhosts` file in its home directory and that there is no `/etc/hosts.equiv` file. These are the files that the "r" programs use to assign "trust" to users on other computers. The following short shell script will check every home directory and tell you which ones have `.rhosts` files in them.

```
#!/bin/sh
for i in `cut -d: -f6 /etc/passwd`; do
     if [ -e $i/.rhosts ]; then
          echo "SECURITY CHECK found .rhosts in $i"
     fi
done
```



Step 3.1.2. Turn off `inetd` if there are no services

If none of the services in `/etc/inetd.conf` are needed at all, the `inetd` process itself can be turned off:

```
[root]# /etc/rc.d/init.d/inet stop
[root]# /sbin/chkconfig inet off
```

Administrators for distributions that do not have `chkconfig` can simply remove the init scripts directly:

```
[root]# /bin/rm /etc/rc.d/rc[345].d/*inet
```

If, for some reason, one of the services controlled by `inetd` is needed later, it can be turned on temporarily by running the SYSV init script:

```
[root]# /etc/rc.d/init.d/inet start
```

STEP 3.2 USE TCP WRAPPERS TO CONTROL ACCESS TO REMAINING INETD SERVICES

If you absolutely have to have one or more of the services in `/etc/inetd.conf`, use TCP wrappers to restrict access to a limited number of hosts. The TCP wrapper daemon, `/usr/sbin/tcpd`, is a small program that is invoked by `inetd` instead of the normal daemon program. It checks the source address of the request against its access control lists in `/etc/hosts.allow` and `/etc/hosts.deny`. If the lists say that the requester is allowed to connect, the wrapper daemon passes the connection to the appropriate service and everything proceeds as usual. If the lists say to deny access to the service, the connection is dropped. Either way, the wrapper daemon puts an entry in the system logs to document the event.

Wrappers are installed by default with Red Hat and all other Linux distributions. The TCP wrapper daemon is invoked by default in `/etc/inetd.conf` for every external service, with the exception of the authentication daemon, `identd`, and the `linuxconf` administration tool. See the manual page for `hosts_access(5)` for details on the syntax of the `/etc/hosts.allow` and `/etc/hosts.deny` files.

Step 3.2.1. Set the default access rule to deny all

Edit `/etc/hosts.deny` so that the **only** uncommented lines read:

```
ALL: ALL
ypserv: ALL
```

The second line is necessary only if you have installed the Network Information System (NIS) ypserv package.

Step 3.2.2. Allow access to only specific hosts for specific services

For example, to allow ftp access from `host.example.org`, edit `/etc/hosts.allow` and add the line:

```
in.ftpd: host.example.org
```

After entering the access control statement in `/etc/hosts.allow`, edit `/etc/inetd.conf` and remove the comment character from the line for the ftp service:

```
ftp    stream    tcp    nowait    root    /usr/sbin/tcpd    in.ftpd -l -a
```

and restart `inetd`:

```
[root]# killall -HUP inetd
```

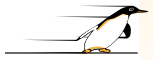
Whenever possible, avoid allowing access to entire domains or subnets, e.g. "`.example.org`" or "`192.168.1.`". Use only the specific hostnames or IP addresses of the machines that are allowed access.

Step 3.2.3. Check the syntax of the access lists with tcpdchk

The TCP wrapper package includes a simple program for checking the syntax of the `/etc/inetd.conf`, `/etc/hosts.allow` and `/etc/hosts.deny` files to make sure that there are no errors and that the files are consistent. Assuming `/etc/hosts.allow` and `/etc/hosts.deny` appear as they do above, here is a sample run of `tcpdchk`:

```
[root]# tcpdchk -v
Using network configuration file: /etc/inetd.conf

>>> Rule   /etc/hosts.allow line 6:
daemons:   in.ftpd
clients:   host.example.org
warning:   /etc/hosts.allow, line 6: host.example.org: host not found
access:    granted

>>> Rule   /etc/hosts.deny line 9:
daemons:   all
clients:   all
access:    denied
```

Note the warning about `host.example.org` saying that it was not found in a name lookup. This tells you that you may have mistyped or otherwise erroneously entered the host name (in this case the hostname is fictitious). Note also that `tcpdchk` will issue warnings about other services that are **not** listed in `/etc/inetd.conf`, but are compiled with TCP wrappers support, such as `portmap` for RPC services (see Step 4.6.1) and `sshd` for Secure Shell (see Step 4.2.3). These warnings can be ignored.



Step 3.2.4. Set up banners for TCP wrapped services

An important option for use with TCP wrappers is that of "banners." Most government agencies, and many commercial organizations, require banners on system access points, such as TELNET and FTP ports. Login banners warning that a system is off limits to unauthorized personnel are often required in order to successfully prosecute a person that breaks into the computer. Some TCP wrapped services can be set up to automatically display a standard login banner. A `Makefile` has been supplied with the TCP wrappers documentation that makes maintenance of banners easier. To create an initial set of banners, execute the following:

```
[root]# mkdir /etc/banners
[root]# cp /usr/doc/tcp_wrappers-7.6/Banners.Makefile /etc/banners/Makefile
[root]# cd /etc/banners
[root]# echo "This is only the prototype banner. Replace with the real banner." >prototype
[root]# make
```

By default, the `Makefile` creates banner files for the `in.telnetd`, `in.ftpd`, and `in.rlogind` services. Read the comments in the `Makefile` for information about banners for other services. The actual wording for the banner is usually a matter of local security policy. The Computer Incident Advisory Capability (CIAC) group has issued an information bulletin about banners that contains sample wording. See Appendix A for the URL.

To activate the banners, edit `/etc/hosts.allow`. For all lines that allow access for the banner services, add the "`banners /etc/banners`" option, followed by the "`allow`" option. Add another option line at the very bottom of `/etc/hosts.allow` setting up banners for all denied services. For example, if `telnet` sessions are allowed from `work.example.org`:

```
in.telnetd: work.example.org : banners /etc/banners : allow
all : all : banners /etc/banners : deny
```

There are a rich set of options for TCP wrappers, including adjusting syslog facilities and levels, adjusting the run-time characteristics of the daemons, "booby traps," and more. See the manual page `hosts_options(5)` and Appendix A for more information.

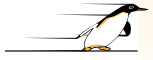
STEP 3.3 DISABLE RUN-TIME NETWORK SERVICES

The Internet daemon handles transient, "one-time" network connections, but there are other network daemons that start at boot-time and normally run until system shutdown, such as `httpd` for the Web, `named` for DNS services, and `smbd` for Samba SMB networking services. They are started at boot-time by the `init` process — process number 1. Red Hat Linux, and most other distributions except Slackware, have a SYSV-style `init`, which uses the concept of runlevels and run-time control scripts to manage the startup and shutdown of services and processes. See Step 2.6.2 for a discussion of the configuration file for `init`, `/etc/inittab`.

See the manual page `init(8)` for more information about `init` and a description of the six normal runlevels. There are two runlevels for normal operation: runlevel 3 has a character-mode console for login; runlevel 5 starts the X Window server and uses the X Display Manager (xdm) login. The scripts that control the startup and shutdown of processes for runlevel 3 are found in the directory `/etc/rc.d/rc3.d`, and runlevel 5 in `/etc/rc.d/rc5.d`. To be precise, the scripts in these directories are symbolic links to general-purpose scripts used for every runlevel in `/etc/rc.d/init.d`.

The names of the scripts are prepended with numbers indicating the order in which they should be executed, for example `S10network` is executed before `S55named`, which is executed before `S85httpd`, and so on. The "S" tells `init` that the script is for startup, a "K" tells `init` that the script is for shutdown (or "Kill"). "S" scripts are passed a single argument, `start`, the "K" scripts are passed the argument `stop`. At any time, root can start or stop services by executing the scripts in `/etc/rc.d/init.d`, for example, to shutdown the Web server:

```
[root]# /etc/rc.d/init.d/httpd stop
```

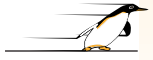


Step 3.3.1. Determine which network services are running

Use `/bin/netstat` to list the ports that are "listening" for connections:

```
[root]# netstat -at
Active Internet connections (servers and established)
Proto  Recv-Q  Send-Q Local Address            Foreign Address       State
tcp         0       0 *:netbios-ssn           *:*                   LISTEN
tcp         0       0 *:www                   *:*                   LISTEN
tcp         0       0 *:smtp                  *:*                   LISTEN
tcp         0       0 *:1040                  *:*                   LISTEN
tcp         0       0 *:702                   *:*                   LISTEN
tcp         0       0 *:697                   *:*                   LISTEN
tcp         0       0 *:692                   *:*                   LISTEN
tcp         0       0 *:673                   *:*                   LISTEN
tcp         0       0 *:printer               *:*                   LISTEN
tcp         0       0 192.168.1.3:domain      *:*                   LISTEN
tcp         0       0 localhost:domain        *:*                   LISTEN
tcp         0       0 *:sunrpc                *:*                   LISTEN
```

Under "Local Address" the port numbers are reported by name if the service is found in the file `/etc/services`, otherwise only the port number is given. As we can see, the default installation still has a lot of services running even after commenting out all the services in `/etc/inetd.conf`.



Unfortunately, `netstat` does not say which process is listening on which port. There is another tool called `lsof`, for "List Open Files" which lists all the open files, including network sockets, for any process. Depending on the installation process followed in Step 2 above, `lsof` may not be installed. For Red Hat Linux the package is on the installation CD-ROM, or the source code can found on the primary archives listed in Appendix A section. `Lsof` has a lot of options and a lot of flexibility. See the manual page `lsof(8)` for more information. To list all open network sockets:

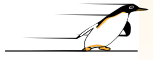
```
[root]# lsof -i +M
COMMAND    PID  USER     FD   TYPE    DEVICE      NODENAME
portmap    3264 root     3u   inet    4546        UDP *:sunrpc[portmapper]
portmap    3264 root     4u   inet    4547        TCP *:sunrpc[portmapper] (LISTEN)
named      3405 root     4u   inet    4724        UDP *:1031
named      3405 root     20u  inet    4720        UDP localhost:domain
named      3405 root     21u  inet    4721        TCP localhost:domain (LISTEN)
named      3405 root     22u  inet    4722        UDP 192.168.1.3:domain
named      3405 root     23u  inet    4723        TCP 192.168.1.3:domain (LISTEN)
lpd        3432 root     5u   inet    4757        TCP *:printer (LISTEN)
rpc.statd  3462 root     3u   inet    4790        UDP *:671[status]
rpc.statd  3462 root     4u   inet    4793        TCP *:673[status] (LISTEN)
rpc.rquot  3472 root     3u   inet    4813        UDP *:681[rquotad]
rpc.mount  3482 root     3u   inet    4833        UDP *:690[mountd]
rpc.mount  3482 root     4u   inet    4836        TCP *:692[mountd] (LISTEN)
rpc.mount  3482 root     5u   inet    4841        UDP *:695[mountd]
rpc.mount  3482 root     6u   inet    4844        TCP *:697[mountd] (LISTEN)
rpc.mount  3482 root     8u   inet    4849        UDP *:700[mountd]
rpc.mount  3482 root     10u  inet    4852        TCP *:702[mountd] (LISTEN)
sendmail   3522 root     4u   inet    4924        TCP *:smtp (LISTEN)
httpd      3549 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3558 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3559 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3560 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3561 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3564 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3565 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3566 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3567 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3568 root     15u  inet    4976        TCP *:www (LISTEN)
httpd      3569 root     15u  inet    4976        TCP *:www (LISTEN)
smbd       3591 root     5u   inet    5024        TCP *:netbios-ssn (LISTEN)
nmbd       3601 root     5u   inet    5046        UDP *:netbios-ns
nmbd       3601 root     6u   inet    5048        UDP *:netbios-dgm
nmbd       3601 root     8u   inet    5052        UDP localhost:netbios-ns
nmbd       3601 root     10u  inet    5054        UDP localhost:netbios-dgm
```

To reduce this list to just the names of the daemons themselves:

```
[root]# lsof -i -Fc | grep '^c' | cut -b2-20 | sort -u
httpd
lpd
named
nmbd
portmap
rpc.mountd
rpc.rquotad
rpc.statd
sendmail
smbd
```

Step 3.3.2. Eliminate unnecessary services

The trick is to know which init script controls which daemon. Most of the daemons are controlled by scripts of the same name, the others can be found by using grep to search for the name in `/etc/rc.d/init.d/*`. To eliminate the service, first shut it down with its init script, then remove the script from the runlevel directory. For Red Hat Linux, the easiest way to remove the links from the runlevel directories is with the `/sbin/chkconfig` program, or you can simply delete them. By default, `chkconfig` adds or removes links to scripts in levels 3, 4, and 5. It does not delete the actual script in `/etc/rc.d/init.d`. If you are manipulating the runtime scripts manually, remember to always remove the startup script from both the runlevel 3 and runlevel 5 directories, so that changing the default runlevel will not unintentionally open network services you thought were turned off.



The following will shutdown all services listed in Step 3.3.1 above:

```
[root]# /etc/rc.d/init.d/httpd stop
[root]# /sbin/chkconfig httpd off
[root]# /etc/rc.d/init.d/lpd stop
[root]# /sbin/chkconfig lpd off
[root]# /etc/rc.d/init.d/named stop
[root]# /sbin/chkconfig named off
[root]# /etc/rc.d/init.d/netfs stop
[root]# /sbin/chkconfig netfs off
[root]# /etc/rc.d/init.d/smb stop
[root]# /sbin/chkconfig smb off
[root]# /etc/rc.d/init.d/nfs stop
[root]# /sbin/chkconfig nfs off
[root]# /etc/rc.d/init.d/portmap stop
[root]# /sbin/chkconfig portmap off
[root]# /etc/rc.d/init.d/sendmail stop
[root]# /sbin/chkconfig sendmail off
```

To remove the startup scripts manually, replace `chkconfig` with `rm` like so:

```
[root]# /bin/rm /etc/rc.d/rc[345].d/*sendmail
```

Once the service is turned off, consider removing the software package entirely. This frees up disk space and enhances security since local users can not abuse a service that isn't there.

Step 3.3.3. Check for any remaining services

Return to Step 3.3.1 and repeat with the `netstat` and `lsof` commands, checking for any remaining network services, and disable them until all network services are turned off.

STEP 3.4 GET THE LATEST VERSIONS OF SOFTWARE

In the remaining steps below, we discuss increasing the security of selected services. You should only enable the absolute minimum set of services necessary to do your work. Before enabling any services, make sure you have the latest version of the software. All the Linux vendors do a good job of keeping up with security-related bug fixes and update their FTP sites regularly with the latest software versions.

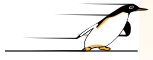
Step 3.4.1. Find security-related updates

Updated software, or "errata" in Red Hat's parlance, can be found on their web site at `http://www.redhat.com/corp/support/errata`, see Appendix A for the locations of updates for other distributions. Red Hat has recently revamped the errata page for Red Hat 6.0, with security-related updates in one column and other package updates in another. At a minimum, you should review and retrieve all of the security-related updates.

Step 3.4.2. Download updates

FTP URLs to the updated packages are included on the errata page itself, so downloading the packages can be done straight through your Web browser. A good practice is to create a separate directory, for example `/usr/local/updates`, and place the updated packages there. Updated packages can also be downloaded en masse from `ftp://updates.redhat.com`. Select the directory for the proper version number and architecture, e.g. `6.0/i386`, and download all the files into `/usr/local/updates`. (This can get tedious week after week, see Step 3.4.4 for information about automating the process).

RPM has the ability to install packages directly over the network by supplying a URL for the package name; this eliminates the need for a separate download directory. For workstations on slow and/or unreliable network connections, like modem lines, it is best to download the package and install from a local copy, rather than having the network link drop or hang halfway into an update.



Step 3.4.3. Install updates

Instructions for installing the updates with rpm are included in the errata pages. Generally, the package can be updated like so:

```
[root]# rpm -Uvh <package-name>
```

where `<package-name>` is the fully-qualified file name for the RPM package, and the options are: "U" for upgrade, "v" for verbose, "h" to print hash marks while it installs ("v" and "h" are for a nicer display, only "U" is required).

For example to update Samba:

```
[root]# rpm -Uvh samba-2.0.5a-1.i386.rpm
```

To update directly over the network, give a URL for the package name:

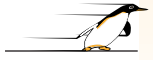
```
[root]# rpm -Uvh ftp://updates.redhat.com/6.0/i386/samba-2.0.5a-1.i386.rpm
```

If you have downloaded all of the latest updates packages into /usr/local/updates, you can "freshen" the packages that are already installed on your system with the command:

```
[root]# rpm -F /usr/local/updates/*
```

This will upgrade only those packages in `/usr/local/updates` that have already been installed, but with an earlier version number.

Note that to update the kernel, use `-i` for "install" rather than `-U` for "upgrade". This will leave the previous version of the kernel intact (the "U" option removes the previous kernel), so if the new kernel does not work for some reason, the machine can still be booted with the old kernel. Also, if you are running a modular kernel, especially on a SCSI-based system, you may need to rerun `mkinitrd` before booting the updated kernel. See the manual page for `mkinitrd(8)` for more information.



Step 3.4.4. Automate the process

It is important to update the packages before using the workstation for day-to-day work, but it is just as critical to stay up to date as time goes on. New bug fixes and enhancements are generated all the time. Trying to keep your local workstation synchronized with the updates can be tedious and time consuming.

Happily, there are several third-party tools that automate the process of checking an FTP site for new packages, and even installing the updates without human intervention. You can find these tools on the Metalab FTP site in the `/utils/package/` subdirectory, on Freshmeat, and many other popular Linux FTP sites. See Appendix A for specific URLs. We will discuss only one program here, but there are several others.

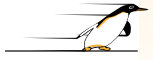
Step 3.4.4.1. Use AutoRPM to automate updates

See Appendix A for the URL of the AutoRPM home page. To quote from the AutoRPM man page:

```
AutoRPM is a program that can do any combination of the
following: mirror RPMs from an FTP site, keep installed
RPMs consistent with an FTP site or local directory, and
keep installed RPMs in a cluster or network of systems
consistent.  It is highly flexible.
```

To run AutoRPM you must first install the "`libnet`" Perl library, available from any Perl CPAN site. If you are unfamiliar with installing Perl CPAN libraries, the author of AutoRPM has thoughtfully provided an RPM version available from his Web site.

In our testing of AutoRPM for this guide, we found that the program is indeed very flexible, but a little tricky to install. The tar-ball could use a Makefile to make installation easier. The default configuration seems to work quite well for Red Hat Linux, and after a little browsing through the man pages, changing the configuration to find updates for other distributions was not difficult.



To install AutoRPM, download the compressed tar file and extract the program:

```
[root]# cd /usr/local/src
[root]# tar xzf <download-dir>/autorpm*.tar.gz
[root]# cd autorpm*
```

where `<download-dir>` is where the tar file was downloaded to. The wildcards in the commands simply avoid typing in the actual version number (which probably has changed by the time this guide went to press).

Create the configuration directory and install the configuration files, programs and man pages:

```
[root]# mkdir --mode 700 /etc/autorpm.d
[root]# cp autorpm.conf autorpm.d/* /etc/autorpm.d
[root]# mkdir --mode 700 /etc/autorpm.d/pools
[root]# cp pools/* /etc/autorpm.d/pools
[root]# install --owner=root --group=bin --mode=0700 autorpm.pl /usr/sbin/autorpm
[root]# cp autorpm.8 /usr/local/man/man8
[root]# cp autorpm.conf.5 /usr/local/man/man5
```

Read the manual pages for `autorpm(8)` and `autorpm.conf(5)`, but the default settings should work for Red Hat Linux.

Run AutoRPM manually in interactive mode to check for updated `RPMs`:

```
[root]# autorpm --interactive
```

AutoRPM will download the package names from the FTP sites listed in `/etc/autorpm.d/pools/redhat-updates` and compare them to the set of currently installed packages. The default list of FTP sites is quite long. It is good practice to edit the list and pare it down to only those sites you know and trust, such as `ftp://updates.redhat.com` and perhaps a local mirror. Next, an interactive dialog screen allows you to choose to update now, update later, or view RPM information such as files, dependencies, scripts, and signatures. The manual page has all the details.

Before you try to run AutoRPM to automatically download and install updates, edit `/etc/autorpm.d/redhat-updates` and add the command "`PGP_Require Yes`" in the action block labeled "`action (updated)`". This will force a PGP signature check of the RPM package before it is installed. You will need to install PGP or GPG and download the public keys for Red Hat before this will work. See the "PGP SIGNATURES" section of the `rpm(8)` man page for more information.

AutoRPM also comes with a cron job that can be placed in `/etc/cron.daily` or `/etc/cron.weekly`, whichever you prefer:

```
[root]# cp autorpm.cron /etc/cron.weekly/autorpm
```

The cron job is set up to mail reports of what needs to be done to the address listed in `/etc/autorpm.d/autorpm.conf` in the variable `ReportDest` (root, by default).

Configuring AutoRPM to work for other distributions can be as simple as changing the list of FTP sites in the `/etc/autorpm.d/pools/redhat-updates` file, however you may want to change the name to match your distribution. If so, you will need to edit `/etc/autorpm.d/autorpm.conf` and change the entry for `Config_File` to reflect the new name.

Step 3.4.5. Subscribe to security-related mailing lists

There are a lot of mailing lists that distribute information about computer security, newly discovered vulnerabilities and exploits, bug fixes, and updates. Even with an automated process for finding updated RPMs, it is important to stay informed about security-related problems for which updated software may not be available yet. At least you will know to turn a vulnerable service off until a fix is available.

See Appendix A for a list of security-related mailing lists and instructions for subscribing. For a typical workstation user, a high-volume mailing list like Bugtraq could result in information overload. At a minimum, subscribe to the "announce" list for your Linux distribution.

STEP 3.5 CACHING-ONLY DOMAIN NAME SERVICE (DNS)

Red Hat 6.0 includes the latest version of the Berkeley Internet Name Domain (BIND) software, version 8.2. This version includes several security-related bug fixes and additional security enhancements. Regardless, a Linux workstation has no need to run its own name server, even in a caching-only mode. Permanently disable and remove the software.

Step 3.5.1. Disable and remove DNS server software

Stop the `named` daemon, if it hasn't been done in step 3.3.2 above, and remove the software from the workstation:

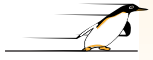
```
[root]# /etc/rc.d/init.d/named stop
[root]# rpm -e caching-nameserver
[root]# rpm -e bind
```

Step 3.5.2. Set primary and secondary name servers

If this was not done during the installation process, edit `/etc/resolv.conf` and add the IP addresses of your primary and, if available, secondary nameservers. Each of the name servers is denoted on a line by itself that looks like:

```
nameserver www.xxx.yyy.zzz
```

Logically enough, the IP for the primary nameserver is listed first, and the secondary nameserver second. Since the caching Domain Name Service has been removed, make sure that there is no `nameserver` line that points to your local IP number or 127.0.0.1.



STEP 3.6 ELECTRONIC MAIL

A typical workstation does not need to run a Mail Transfer Agent (MTA), like `sendmail`, in daemon mode. Most Internet service providers, university, government, and commercial organizations provide mail servers.

Step 3.6.1. Turn off sendmail daemon mode

Edit `/etc/sysconfig/sendmail` to read:

```
DAEMON=no
QUEUE=15m
```

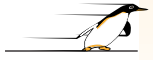
The QUEUE option controls how often sendmail will "wake up" to process the outgoing mail queue, which is set at 15 minutes in the above example. After saving the edited `/etc/sysconfig/sendmail`, restart the `sendmail` daemon:

```
[root]# /etc/rc.d/init.d/sendmail restart
```

Step 3.6.2. Define SMTP server for mail clients

Red Hat, like most Linux distributions, comes with a large number of Mail User Agents (MUAs), including mail (aka Mail), elm, mutt, pine, nmh, exmh, and Netscape. Each of these programs can be configured to send mail through an external MTA. Depending on the requirements for the organization, the external MTA will handle relaying the mail to its final destination and address translation.

The mail program (`/bin/mail`, aka `/usr/bin/Mail`), elm, mutt, and pine will all invoke `sendmail` stand-alone to deliver the mail to the given e-mail address, thus their configuration does not include an entry for an SMTP server. The assumption here, though, is that your organization has an SMTP server that handles relaying all out-bound electronic mail.



▲ **Step 3.6.2.1. Set out-bound SMTP server for sendmail**

You direct `sendmail` to relay all out-bound mail to the SMTP server with the `SMART_HOST` feature. There are two ways to do this, using the `m4` macros, or editing `/etc/sendmail.cf` directly. Using `m4` macros for anything other than the simplest modifications to `/etc/sendmail.cf` is the preferred method, and is discussed in Step 4.4 below. For now, the simplest way to set the out-bound SMTP server for sendmail is to edit `/etc/sendmail.cf` directly. Look for a line that starts with `DS` (this should be near the top of the file, just after a heading that says "`local info`"). Edit the line to read:

```
DSsmtp.my.domain
```

where `smtp.my.domain` is the fully qualified domain name of the SMTP server. To cover all bases for locally delivered mail, too, search for the lines that start with the two-letter combinations "`DR`", "`DH`", and "`DM`" edit them accordingly:

```
DRsmtp.my.domain
DHsmtp.my.domain
DMmy.domain
```

This will handle unqualified addresses (those without any @domain portion), local addresses (user@host without any other domain information), and sets the masquerade address, respectively. Save `/etc/sendmail.cf` and restart sendmail.

▲ **Step 3.6.2.1. Set out-bound SMTP server for other mail clients**

As mentioned above, Pine will default to using sendmail on the localhost for sending mail. You can set the SMTP server directly in Pine from the main menu by selecting "S" for setup, then "C" for config, moving down to the field labeled `smtp-server` and entering the hostname of the SMTP server.

The "new MH message system," nmh, and its X Window GUI front-end, exmh, can be configured to use an external SMTP server by editing the file `/etc/nmh/mts.conf`. Edit the `servers:` option, replacing the default "localhost" with the hostname of your SMTP server.

The SMTP server is set in Netscape Messenger by selecting the `Edit->Preferences...` dialog, selecting `Mail & Newsgroups->Mail Servers`, and placing the hostname of the SMTP server in the window under `Outgoing Mail Server`.

STEP 3.7 NFS CLIENT-SIDE SECURITY

A typical workstation does not need to export directories to be NFS-mounted by another computer. The Network File System has inherent security problems because the Remote Procedure Call (RPC) mechanism it relies on uses IP number and UID for authentication, opening NFS to IP spoofing attacks.

Step 3.7.1. Turn off NFS exports and remove NFS daemons

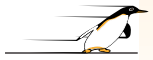
If there is no alternative to exporting NFS mount points, see Step 4.6 below for information on limiting access and permissions, otherwise turn off NFS and remove the NFS software completely:

```
[root]# /etc/rc.d/init.d/nfs stop
Shutting down NFS services:                [  OK  ]
Shutting down NFS mountd:                  [  OK  ]
Shutting down NFS daemon:                  [  OK  ]
Shutting down NFS quotas:                  [  OK  ]
Shutting down NFS statd:                   [  OK  ]
[root]# rpm -e knfsd
```

Step 3.7.2. Configure local NFS mounts

Even with the NFS daemons deleted from the workstation, you can still mount NFS directories from a central server as long as the NFS file system is compiled into the kernel. In fact for a workstation environment, this is the normal use of NFS. For example, say you are a member of a software development team and the source code for your project is maintained on a central server called `project` in the directory `/usr/src/devel`. You wish to mount this directory on your workstation on the directory `/mnt/devel`. Since all you need to do is copy source files in and out, mount the directory with the following options:

`rw`	Mount read-write
`hard`	Do a “hard” mount, i.e. hang if the server does not respond
`noexec`	Do not allow executing binaries
`nosuid`	Do not honor set-uid or set-gid bits (redundant with noexec)
`nodev`	Do not interpret character or block special files



For example:

```
[root]# mkdir /mnt/devel
[root]# mount -t nfs -o rw,hard,noexec,nosuid,nodev project:/usr/src/devel /mnt/devel
```

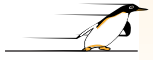
Note that read and write permissions on the NFS-mounted directory are defined by the UID and GID of the owner on the server (project), not the client (your workstation), so the UIDs and GIDs need to match up across both machines.

To have this directory mounted every time your workstation is booted, put the mount point in `/etc/fstab` like so:

```
/mnt/devel project:/usr/src/devel nfs rw,hard,noexec,nosuid,nodev,bg,auto 0 0
```

Adding `auto` allows for the init scripts to mount the NFS directory at boot-time (i.e. with `mount -a`). Note, however, that specifying a `hard` mount means that the workstation will hang if the NFS server is not up and running. Therefore, we have added `bg`, telling `/bin/mount` to do the mount in the background and allow us to proceed in the event the server is not immediately available. You may also wish to add another option, `intr`, to allow the mount process to be interrupted in case of a hang. If the NFS server or the local network have reliability problems, consider using a `soft` mount, as opposed to a `hard` mount. The advantage is that the workstation will not hang when the NFS server is down or unavailable; the disadvantage is that you may lose data on the NFS server if it is down and the client times out.

In the case that the NFS mount is expressly for the purpose of sharing executable programs (like `/usr/local/bin` on the NFS server, for example), you will need to change `noexec` to `exec`, but think hard before changing `nosuid` to `suid` to allow set-uid and set-gid permissions. Note that the `nosuid` and `noexec` options can still be defeated by programs like `suidperl` that copy a script as a normal data file, then execute it with SUID privileges. See the manual page for `mount(8)` and `fstab(5)` for more information.



STEP 3.8 LIMIT WORLD WIDE WEB SERVICES TO THE LOCAL HOST

Like most distributions, Red Hat 6.0 installs a fully functional Apache Web server that, by default, allows connections from any computer on the Internet. For a typical workstation setup, this is not necessary. If you are a Web developer, though, having a fully functional localized Web server is extremely useful for testing pages, CGI scripts, applets, etc. before they are deployed on an operational Web site.

Step 3.8.1. Turn off HTTP and remove the server software

If there is no need for a local web server, simply remove it from the workstation:

```
[root]# /etc/rc.d/init.d/httpd stop
Shutting down http:                        [   OK   ]
[root]# rpm -e apache
[root]# rpm -e apache-devel
```

Step 3.8.2. Limit HTTP access to localhost only

The Apache HTTP daemon can be bound to a specific network interface. If the interface is `localhost`, IP 127.0.0.1, then any attempt to connect to port 80 from an outside network results in a connection refused error. From the outside, it looks like there isn't any Web server on the machine at all.

Edit `/etc/httpd/conf/httpd.conf`, and look for a section near the bottom with the option "`Listen`" — it should come just before the section for `VirtualHost`. Add the following line:

```
Listen 127.0.0.1:80
```

Save `/etc/httpd/httpd.conf` and restart the Web server:

```
[root]# /etc/rc.d/init.d/httpd restart
Shutting down http:                        [   OK   ]
Starting httpd:                            [   OK   ]
[root]# netstat -a | grep www
tcp         0       0 localhost:www         *:*                 LISTEN
```

The `netstat` command shows us that the Web daemon is listening only on the loopback device, not the regular network device. To test that access is denied, from another computer, try:

```
$ telnet workstation 80
Trying 192.168.1.3...
telnet: Unable to connect to remote host: Connection refused
```

The Web server is now closed off to everyone except users on the workstation itself.

STEP 3.9 REMOVE ANONYMOUS FTP SERVICE

Even though you probably turned off the FTP service in Step 3.1.1 above, or limited access to it with TCP wrappers in Step 3.2.1, you should still eliminate the ability to do anonymous FTP. First edit `/etc/ftpaccess` and change the line that reads:

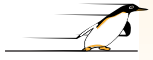
```
class    all    real,guest,anonymous    *
```

to read:

```
class    all    real   *
```

Finally, remove the home directory used for anonymous FTP:

```
[root]# rpm -e anonftp
```



STEP 4.1 SERVERS: SEE STEPS 3.1, 3.2, 3.3, AND 3.4 FOR DISABLING ALL UNNECESSARY SERVICES, SETTING WRAPPERS, AND UPDATING SOFTWARE

Like a workstation, a server should run only those few network services necessary to fulfill its purpose. An information server may be set up to run HTTP and FTP, but should not also bear the burden of operating large mailing lists or file sharing to a number of clients through Samba or NFS. While the remainder of this section provides steps for securing a number of services, this by no means implies that a single machine should provide all of them at the same time.

A typical server operates in a "lights-out" environment in a computer room. It should have as few user accounts as possible, and remote access to the server should be limited to Secure Shell only, to avoid the possibility of sniffing passwords on the network, and to provide strong authentication and encryption of the login session.

Consider whether any software development tools are necessary. A common mode of attack by crackers and "script kiddies" is to break into an ordinary user account, then download and compile exploit programs and root kits to compromise root and hide their tracks. If there is no C compiler or other development tools, it makes the task of the cracker that much harder. Development of tools and programs that have to run on the server, such as CGI scripts for a Web server, should be done on workstations, and tested thoroughly before installation on the operational server.

While you are removing the packages for the compilers and development libraries, you should also browse through the list of packages on the system with `rpm -qa` to see what other superfluous packages can be removed. In addition to the security benefits, removing unneeded packages frees up disk space that can be used for additional logging and temporary space. If you are not sure what a package is for, try `rpm -qi <package-name>` to get more information. In fact, it can be very useful to execute:

```
[root]# rpm -qail >/root/package.list
[root]# chmod 400 /root/package.list
```

This will dump a list of all installed packages, the package descriptions and a list of all the files in each package.

STEP 4.2 INSTALL SECURE SHELL FOR REMOTE ACCESS

Secure Shell (SSH) was originally developed at the Helsinki University of Technology. It provides strong authentication using RSA public key cryptographic algorithms and automatic and transparent encryption of network communications for login and file copying operations. Source code for version 1 of the SSH protocol is freely available for non-commercial use. The license for protocol version 2 source code is much more limited, and any use at a commercial or government organization is forbidden without a commercial license. See `http://www.ssh.fi/sshprotocols2/download.html` for links to the License Agreement for SSH2.

Step 4.2.1. Download, compile, and install SSH

SSH version 1 is available for download from `ftp://ftp.cs.hut.fi/pub/ssh/`, and if this site is too busy, there is a list of mirror sites on the SSH download page cited above. As of the writing of this guide `ssh-1.2.27.tar.gz` was the latest version, and we will use this version number in our examples below. Check the FTP site for later versions, since problems and bugs are corrected in the version 1 source code occasionally.

```
[root]# wget ftp://ftp.cs.hut.fi/pub/ssh/ssh-1.2.27.tar.gz
[root]# tar xzf ssh-1.2.27.tar.gz
[root]# cd ssh-1.2.27
[root]# ./configure --with-libwrap --without-rsh --disable-suid-ssh --disable-scp-stats
[root]# make
[root]# make install
```

The options to the `configure` script do the following: compile the SSH daemon with support for TCP wrappers; disables all support for using rsh as a fall-back option; disables the set-uid bit on the SSH client; and turns off the printing of on-screen statistics by the secure copy program, `scp` (to emulate the behavior of `rcp`).

Step 4.2.2. Start the SSH daemon

The SSH daemon, `sshd`, should be started at boot time and remain running as long as the server is up. A simple means for doing this is to add the line:

```
/usr/local/sbin/sshd
```

to `/etc/rc.d/rc.local`, where it will be executed after all other boot-time programs are started.

See Appendix C for a Red Hat-centric SYSV-init script using the `daemon` and `killproc` functions. Copy this script into `/etc/rc.d/init.d/sshd`, and link it into the runlevel directories:

```
[root]# for i in /etc/rc.d/rc[345].d; do (cd $i; ln -s ../init.d/sshd S50sshd); done
```

or, alternatively:

```
[root]# chkconfig sshd on
```

To start the daemon now, without rebooting:

```
[root]# /etc/rc.d/init.d/sshd start
```

Step 4.2.3. Set up `/etc/hosts.allow` for SSH access

Compiling SSH with the `–with-libwrap` configure option allows `sshd` to perform TCP-wrapper-style access control, inspecting the `/etc/hosts.allow` and `/etc/hosts.deny` files for each connection request. The syntax and options are exactly the same as any TCP-wrapped service, and the service name is `sshd`. For example:

```
# Allow SSH access to anyone in our domain or the goodguys.org domain
sshd: LOCAL .goodguys.org
```

Step 4.2.4. Generate SSH keys

SSH performs user authentication and the exchange of session encryption keys using RSA cryptographic public keys. Each user that is allowed to access the server must generate a public/private key pair on the workstation that they will use to access the server, and the public key must be placed on the server in the account that they wish to login to.

On your remote workstation, install SSH as per the instructions above, then in your regular user account do this:

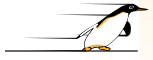
```
[user]$ ssh-keygen
```

The key generator will compute the public and private key pair and prompt for the file to save them in, defaulting to `.ssh/identity` in your home directory. Accept the default location. Then you will be asked to enter a passphrase.

About passphrases: you are not required to enter one, and if you don't, SSH will not prompt you for a passphrase when it uses the key (like when you start a login session, for example). This is especially handy if you plan to automate `ssh` or `scp` (SSH secure copy) in scripts. The downside is that if your workstation and your account are compromised, the attacker will be able to access the server without entering a passphrase.

For ordinary shell and file copy access to the server, it is best to enter a passphrase, keeping in mind all the rules for selecting good passwords. If there is a need for automated, non-interactive access to the server using SSH, consider using `ssh-agent` and `ssh-add`. The agent is a process that holds private keys for use by `ssh` and `scp`. The `ssh-add` program communicates with `ssh-agent` process and sends it the key, prompting for the passphrase if necessary. After your key is stored by `ssh-agent`, you will not need to enter a passphrase again for the remainder of the login session. See the manual pages for `ssh-agent(1)` and `ssh-add(1)` for more information.

When `ssh-keygen` is done, there will be two files in the `.ssh` directory, `identity` and `identity.pub`. The former is your binary private key, the latter is an ASCII file with your public key. The public key has to be stored on the server before you can use SSH to login.



The safest way to transport the public key to the server is "sneaker-net", i.e., copy the key to a floppy and carry it to the server. Transmitting the public key over the network is generally safe, though. Even if the public key is sniffed off the network, it is useless without the private key and it is extremely difficult, if not impossible, to derive one key from the other. **Never** send the private key over the network, however, and make sure that the `.ssh/identity` file has 0600 permissions and is owned by you.

Once the key is copied to the server, put it in the home directory of the account you wish to access (we'll use the root account in our example) and assume the public key is in the file `identity.pub` in the home directory.

```
[root]# cd ~root
[root]# mkdir .ssh
[root]# cat identity.pub >>.ssh/authorized_keys
[root]# chmod 600 .ssh/authorized_keys
```

The `authorized_keys` file holds the public keys for every user that is granted access. If you copy more than one key using the method above, be careful with the redirection; make sure you append (>>) to the file, **do not** overwrite it (>).

Step 4.2.5. Use ssh and scp for remote access

With the SSH daemon running on the server, and the public key of a workstation user account in `/root/.ssh/authorized_keys`, you may now use SSH to access the server securely from your workstation.

The `ssh` command works just like `rsh`. To open a login shell:

```
[user@workstation]$ ssh -l root server
```

To execute a command:

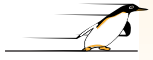
```
[user@workstation]$ ssh -l root server ls /home/httpd/html
```

The `scp` command works like `rcp`. To copy a file to the server:

```
[user@workstation]$ scp This.File root@server:That_File
```

To copy lots of files from the server:

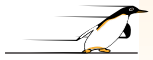
```
[user@workstation]$ scp root@server:All.* backup_dir
```



Note that if you entered a passphrase during the ssh-keygen stage, you will be prompted for that passphrase for each of these commands.

If `ssh` or `scp` do not work correctly, use the `-v` switch to see more detail about the interactions between the client and server:

```
[user@workstation]$ ssh -v -l root server
SSH Version 1.2.27 [i686-unknown-linux], protocol version 1.5.
Standard version.  Does not use RSAREF.
workstation: Reading configuration data /etc/ssh_config
workstation: ssh_connect: getuid 100 geteuid 0 anon 0
workstation: Connecting to server [192.168.1.3] port 22.
workstation: Allocated local port 1022.
workstation: Connection established.
workstation: Remote protocol version 1.5, remote software version 1.2.27
workstation: Waiting for server public key.
workstation: Received server public key (768 bits) and host key (1024 bits).
workstation: Host 'server' is known and matches the host key.
workstation: Initializing random; seed file /home/user/.ssh/random_seed
workstation: Encryption type: idea
workstation: Sent encrypted session key.
workstation: Installing crc compensation attack detector.
workstation: Received encrypted confirmation.
workstation: Trying rhosts or /etc/hosts.equiv with RSA host authentication.
workstation: Server refused our rhosts authentication or host key.
workstation: No agent.
workstation: Trying RSA authentication with key 'user@workstation'
workstation: Received RSA challenge from server.
workstation: Sending response to host key RSA challenge.
workstation: Remote: RSA authentication accepted.
workstation: RSA authentication accepted by server.
workstation: Requesting pty.
workstation: Requesting X11 forwarding with authentication spoofing.
workstation: Requesting shell.
workstation: Entering interactive session.
Last login: Sun Jul 25 11:08:51 1999 from workstation
You have new mail.
[root@server /root]#
```



Step 4.2.6. Replace "r" programs with SSH

Once SSH is installed and working, there is no need to have the "r" clients (`rlogin`, `rsh`, and `rcp`) or the server daemons (`in.rlogind`, `in.rshd`, or `in.rexecd`) on the workstation or the server at all. Even if they have been commented out of `/etc/inetd.conf`, removing the programs is the safest course.

```
[root]# rpm -e rsh
```

Once the "r" programs are gone, SSH programs can be put in their place:

```
[root]# ln -s /usr/local/bin/ssh /usr/bin/rsh
[root]# ln -s /usr/local/bin/slogin /usr/bin/rlogin
[root]# ln -s /usr/local/bin/scp /usr/bin/rcp
```

STEP 4.3 DOMAIN NAME SERVICE AND BIND VERSION 8

Red Hat 6.0 ships with BIND version 8.2, the latest version available as of this writing. Administrators of earlier releases of Red Hat and other distributions should check the version of BIND they are running. If you are running BIND Version 4, upgrade to BIND version 8.2. If you are running 8.1.1 or earlier you must upgrade to BIND Version 8.1.2 or later, because the earlier versions have a known remote root exploit. Check the updates for your distribution, or download the latest source from `http://www.isc.org`.

Step 4.3.1. Restrict zone transfers

Configure the primary master name server to perform zone transfers only to its slave servers, use the `allow-transfer` option for the zone in `/etc/named.conf`, and specify the IP address of only the slave servers for the zone:

```
zone "notransfer.com" {
        type master;
        file "db.notransfer.com";
        allow-transfer { 192.168.1.1; 192.168.2.1; };
};
```

The secondary, or slave, servers should not transfer zone information to any other name servers. In their copy of `/etc/named.conf`, put this:

```
zone "notransfer.com" {
        type slave;
        masters { 192.168.0.1; };
        allow-transfer { none; };
};
```

Step 4.3.2. Restrict queries

You can limit the hosts that are allowed to query a name server with the `allow-query` option. You can limit the interfaces that queries will be allowed on with the `listen-on` option. These options can be applied globally in the options list in `/etc/named.conf`:

```
options {
        allow-query { 192.168.0/24; };
        listen-on { 192.168.0.1; 127.0.0.1; };
};
```

Or the restriction can be for particular zones:

```
zone "notransfer.com" {
        type slave;
        masters { 192.168.0.1; };
        allow-transfer { none; };
        allow-query { 127/8; 192.168.0/24; };
};
```

This is useful for protecting zone information on a bastion host in a firewall environment. It allows queries only from the primary master nameserver on the internal network while a shadow primary master nameserver on the external network handles queries from the Internet and serves up a much more restricted view of the firewall network. See “DNS and BIND,” 3rd Edition, Ablitz and Liu, O’Reilly and Associates, 1998 for more information (this book is indispensable for any nameserver administrator).

Step 4.3.3. Run `named` in a chroot jail

BIND Versions 8.1.2 and 8.2 have the ability to run as a user other than root, which helps, but still leaves the system open if the nameserver is compromised. For better security, the service can be set up to run in its own chroot directory tree. Setting up a proper chroot jail must be done with care. Not only do the zone files and other configuration files need to be in the chroot directory tree, but libraries, devices, and executables necessary for operation of `named` have to be there, too. In the following example, the username (dns), UID, GID, and home directory were chosen arbitrarily. Use whatever unique values make sense for your site.

▲ **Step 4.3.3.1. Create the new user and group**

```
[root]#    groupadd -g 150 dns
[root]#    useradd -u 150 -g 150 -M dns
```

▲ **Step 4.3.3.2. Prepare the chroot directory**

```
[root]#    mkdir -m 0700 /home/dns
[root]#    cd /home/dns
[root]#    mkdir -p etc lib dev usr/sbin var/named var/run
[root]#    mknod -m 666 dev/null c 1 3
```

▲ **Step 4.3.3.3. Copy configuration files and programs**

```
[root]#    cp /etc/named.conf etc
[root]#    cp -R /var/named/* var/named
[root]#    chown -R dns.dns var/named var/run
[root]#    cp /usr/sbin/{named,named-xfer} usr/sbin
```

▲ **Step 4.3.3.4. Copy shared libraries**

First, figure out which libraries are needed:

```
[root]#    ldd /usr/sbin/named
           libc.so.6 => /lib/libc.so.6 (0x4001c000)
           /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
[root]#    cp /lib/libc.so.6 lib
[root]#    cp /lib/ld-linux.so.2 lib
```

▲ **Step 4.3.3.5. Set `syslogd` to listen to named logging**

`named` communicates with `syslog` through `/dev/log`, but this isn't possible in the chroot jail. The `syslog` daemon must be told to create a new socket for `named` to write to. Edit `/etc/rc.d/init.d/syslog`, and rewrite the line that starts the `syslog` daemon to read:

```
daemon syslog -a /home/dns/dev/log
```

And restart the daemon:

```
[root]# /etc/rc.d/init.d/named stop
[root]# /etc/rc.d/init.d/named start
```

▲ **Step 4.3.3.6. Edit the named init script**

Edit /etc/rc.d/init.d/named to start the daemon with the new UID/GID in the chroot jail:

```
start)
        # Start daemons.
        echo -n "Starting named: "
        daemon named -u dns -g dns -t /home/dns
        echo
        touch /var/lock/subsys/named
        ;;
```

And restart the nameserver:

```
[root]# /etc/rc.d/init.d/named stop
[root]# /etc/rc.d/init.d/named start
```

To be sure it worked, check /var/log/messages, named should have written entries that read something like:

```
Jul 25 01:08:00 server named[1782]: chrooted to /home/dns
Jul 25 01:08:00 server named[1782]: group = dns
Jul 25 01:08:00 server named[1782]: user = dns
```

Use nslookup to test that the nameserver is operating as expected.

▲ **Step 4.3.3.7. Specify a new control channel for ndc**

If you need to run the nameserver daemon control program, ndc, you will have to specify a new control channel within the chroot jail:

```
[root]# ndc -c /home/dns/var/run/ndc
```

A shell alias or a short script with the new control channel can be used to save keystrokes.

As mentioned in Step 3.6 above, `sendmail` is the Mail Transfer Agent (MTA) supplied with Red Hat 6.0 and most other distributions. Red Hat 6.0 ships with `sendmail` version 8.9.3, the latest version available as of this writing. Administrators of earlier releases of Red Hat or other distributions should upgrade to the version 8.9.3 of `sendmail`. The 8.9.x versions of `sendmail` disallow electronic mail relay by default. Mail relay is a favorite tool of the purveyors of "spam" bulk mail to use your server to forward unsolicited mail.

The configuration and operation of `sendmail` is quite complicated. All `sendmail` administrators should have a copy of "sendmail", Costales and Allman, O'Reilly & Associates, 1997 (the Bat Book) on their bookshelf. Modifying `/etc/sendmail.cf` by hand can be difficult and fraught with danger for the beginner. The best way to make changes is to use the `m4` macro processor system, package `sendmail-cf*.rpm` on the Red Hat distribution media. Full details on using `m4` can be found in the Bat Book.

Step 4.4.1. Turn off SMTP `vrfy` and `expn` commands in `/etc/sendmail.cf`

The `vrfy` SMTP command allows a remote user to verify the E-mail address of a local user on the server. The `expn` SMTP command expands aliases and mailing list `:include:` aliases. Local addresses and aliases should be considered confidential, or at least sensitive, information. To turn off the commands, edit `/etc/sendmail.cf` and search for the line with `PrivacyOptions`. Change it to read:

```
O PrivacyOptions=goaway
```

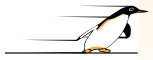
The `goaway` option is shorthand for `authwarnings,noexpn,novrfy,needmailhelo,needexpnhelo, needvrfyhelo`; complete descriptions of each of these privacy options can be found in the Bat Book.

To set this option with `m4` macros, add the following to the build macro (for Red Hat Linux, this is file `/usr/lib/sendmail-cf/cf/redhat.mc`) and rebuild `/etc/sendmail.cf`:

```
define(`confPRIVACY_FLAGS', `goaway')
```

A sample procedure for this follows:

```
[root]# cp /etc/sendmail.cf /etc/sendmail.cf.orig
[root]# cd /usr/lib/sendmail-cf/cf
[root]# echo "define(\`confPRIVACY_FLAGS', \`goaway')" >>redhat.mc
[root]# m4 redhat.mc >/etc/sendmail.cf
[root]# /etc/rc.d/init.d/sendmail stop
[root]# /etc/rd.c/init.d/sendmail start
```



Step 4.4.2. Define hosts allowed to relay mail

Sendmail V8.9 has built-in checks to stop mail relays. Using unwitting mail servers to relay unsolicited mail is a favorite practice of spammers. An organizational mail server has to allow mail relay for mail clients inside the organization, though. Relaying is controlled through an access database, which is in the file `/etc/mail/access` in Red Hat Linux. Other distributions may control relaying through the file `/etc/mail/relay-domains`. Both methods are discussed below.

Step 4.4.2.1. Check that the access database is active

To make sure `sendmail` has the ability to use the access database check for the name of the database in `/etc/sendmail.cf`:

```
[root]# grep Kaccess /etc/sendmail.cf
Kaccess hash -o /etc/mail/access
```

If the line isn't found, the only easy way to add this feature is with the m4 macro `FEATURE(`access_db').` See the sample procedure in Step 4.4.1 above.

If your distribution uses the `relay-domains` file, instead, check to make sure that `sendmail` is using the file:

```
[root]# grep relay-domains /etc/sendmail.cf
FR-o /etc/mail/relay-domains
```

If the line isn't found, edit `/etc/sendmail.cf` and add the line just as it appears above, create the `relay-domains` file and restart the `sendmail` daemon:

```
[root]# touch /etc/mail/relay-domains
[root]# chmod 600 /etc/mail/relay-domains
[root]# /etc/rc.d/init.d/smtp restart
```

▲ Step 4.4.2.2. Set access for domains allowed to relay

The access database has a simple "key value" format: the key is a fully qualified hostname, subdomain, domain, or network; the value is an action, REJECT, DISCARD, OK, RELAY, or an arbitrary message. Only hosts or domains with the RELAY action are allowed to use the mail server as a mail relay. For example, if your domain is example.org, and your server is also handling mail for all hosts on the 128.184.x.x network, edit `/etc/mail/access` to read:

```
example.org                RELAY
128.184                    RELAY
```

For distributions that use `relay-domains`, instead, the file `/etc/mail/relay-domains` should be added to include just those host and domain names that are allowed to relay mail through the server, and no others. Remember to restart the sendmail daemon after modifying the database.

Much more information about relaying and anti-spam configuration control can be found at `http://www.sendmail.org/m4/readme.html`.

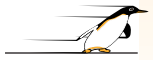
■ Step 4.4.3. Set domain name masquerading

Many organizations prefer to have uniform mail addresses for all personnel, such as `Joe.Schmo@example.org`. Some mail clients can be set so that the uniform mail address is used on all out-bound mail, but managing all the mail clients in even a small group can be tedious. Sendmail can be configured to rewrite the headers of all out-bound mail so that they masquerade as the central mail server. To define the masquerade address, edit `/etc/sendmail.cf` and look for `DM` in columns 1-2. Append the masquerade name to this line:

```
DMexample.org
```

Or, to be complete, use the `m4` macros and add the following:

```
MASQUERADE_AS(`example.org')
FEATURE(masquerade_entire_domain)
FEATURE(allmasquerade)
FEATURE(masquerade_envelope)
```



Step 4.4.4. Install an alternative MTA

Several other mail transfer agents are available to replace sendmail. Two popular aternatives are Qmail (www.qmail.org) by David Bernstein, and Postfix (www.postfix.org) by Wietse Venema. Both of these MTAs were designed and written from the ground up with security and performance in mind. It is beyond the scope of this guide to give details on installing and configuring either of these alternatives, but a wealth of information is available on the Internet.

Step 4.4.5. Secure the POP and IMAP daemons

For mail servers that collect all incoming mail for an organization, a common means to deliver the mail to clients is for them to retrieve the mail using the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP). POP is the older and simpler of the two protocols, providing basic commands for authentication, retrieval and deletion of mail messages from the mail server. IMAP is more flexible and supports creating, deleting, and renaming mail folders (mailboxes), searching, selective retrieval of message attributes and more.

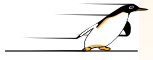
Step 4.4.5.1. Get the latest version of POP and IMAP daemons

Unfortunately, many POP and IMAP daemon implementations have been plagued with vulnerabilities that lead to remote root compromises of mail servers on many platforms. There are several well-known exploit programs available for cracking vulnerable Linux POP and IMAP daemons. Hopefully, most of the problems have been found and fixed, but it is very important to have the absolute latest version of the daemon program installed on the server.

Step 4.4.5.2. Control access to POP and IMAP with TCP wrappers

POP/IMAP is traditionally run out of `inetd`, so access control through TCP wrappers is easy and very important. Limit access to only those hosts that have a legitimate need for the service. For a mail hub that holds mail for the entire `example.org` domain and 128.184 network, and delivers it to clients with POP version 3 or IMAP, put this in `/etc/hosts.allow` (remember from Step 3.2.1 above that `/etc/hosts.deny` has only "`ALL: ALL`" for denial of all services by default):

```
ipop3d: .example.org 128.184.
imapd:  .example.org 128.184.
```



▲ **Step 4.4.5.3. Install an alternative POP or IMAP daemon**

There are several alternative POP daemons available. One of the most popular is Qpopper from Qualcomm, Inc. This version supports all POP3 extensions, APOP, and Kerberos V4. See Appendix for a URL to the Qpopper home page. As of this writing the current stable release is 2.53, and version 3.0 is in beta release. A popular IMAP daemon replacement is Cyrus IMAPD from the Carnegie Mellon Enterprise Electronic Mail Project. See Appendix A for more information on where to locate these packages.

▲ **Step 4.4.5.4. Install an SSL wrapper for secure POP/IMAP connections**

There are several third-party open source programs that wrap TCP services with the Secure Socket Layer (SSL) protocol to provide strong authentication and end-to-end encryption. For a list, see `http://www.openssl.org/related/apps.html`. Two applications that are useful for providing secure POP/IMAP connections are `stunnel` and `sslwrap` (see Appendix A for URLs to these packages). Both packages require the SSLeay or OpenSSL packages. See Step 4.9.5 for information on downloading and compiling OpenSSL.

STEP 4.5 PRINTING SERVICES

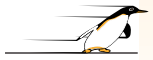
Red Hat Linux ships with the Berkeley line printer system. Over the last few years, a few buffer overflow exploits have been found in the `lpr` and `lprm` commands, both used on the client side of the connection. If you are running a version of Red Hat Linux earlier than 6.0, be sure to update to the latest version of the `lpr` package.

■ **Step 4.5.1. List allowed remote hosts in `/etc/hosts.lpd`**

Put the names of hosts allowed to use this print server in `/etc/hosts.lpd`. You can get the same effect by listing them in `/etc/hosts.equiv`, but that method has serious implications in conjunction with the BSD “r” programs, `rsh`, `rlogin`, etc. (See Step 4.2 for replacing the “r” programs with SSH).

■ **Step 4.5.2. Replace Berkeley lpr/lpd with LPRng**

A popular alternative to the Berkeley lpr/lpd system is LPRng. It is compliant with the RFC1179 requirements for network printing, gives a great deal of flexibility to the administrator for defining permissions for specific actions, and supports authentication with Kerberos and PGP. Caldera OpenLinux and Debian ship with LPRng already, administrators of those distributions can skip the installation step.



▲ **Step 4.5.2.1. Download and install LPRng**

Links to the latest stable version can be found at `http://www.astart.com/LPRng.html`. Download this version, and extract the files from the compressed tar file. Follow the instructions in the file INSTALL in the source directory to compile and install LPRng.

▲ **Step 4.5.2.2. Set remote hosts and/or networks that are allowed access**

The file `/etc/lpd.perms` controls access to specific hosts or networks in addition to controlling specific operations, for instance it is possible to allow or deny specific users from removing jobs from the print queue. The default `/etc/lpd.perms` installed by LPRng is well commented and the package comes with lots of documentation. See the manual page for `lpd.perms(5)` for more information on the permissions configuration file.

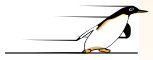
To restrict access to `lpd` to only those hosts in the `example.org` domain, or network `128.184`, place the following in `/etc/lpd.perms`:

```
REJECT SERVICE=X NOT REMOTEIP=128.184.0.0/255.255.0.0
REJECT SERVICE=X NOT REMOTEHOST=*.example.org
```

STEP 4.6 NETWORK FILE SYSTEM

NFS by its very design has some serious security problems. The RPC service depends on simple UID/GID and IP authorization for permissions, all of which are easily spoofed. Sun Microsystems, the developer of the NFS protocol, has enhanced it with Secure RPC that uses cryptographic authentication, but to date there is no Linux implementation of Secure RPC available.

Red Hat 6.0 has moved away from the “user-space” NFS implementation to a “kernel-space” implementation, `knfsd`. Just recently, security problems were discovered in the user-space NFS code. Users of earlier Red Hat versions and other distributions that use the user-space code should upgrade to the latest version. This is not to say that there are no problems with the kernel-space code. The version shipped with Red Hat 6.0 has a bug that makes it impossible for non-Linux systems to mount directories on Linux systems. See `ftp://ftp.kernel.org/pub/linux/devel/gcc/` for the latest version, which corrects this problem.



Step 4.6.1. Set access to RPC services in `/etc/hosts.allow`

RPC services are registered and accessed through the `portmap` daemon. The version of `portmap` supplied with all Linux distributions uses the TCP wrapper library to allow or deny access to RPC services, such as NFS. The only difference from other TCP-wrapped services is that requests from the local host are always authorized, regardless of what is in the `hosts.allow` or `hosts.deny` files, and `portmap` does not do hostname lookups, so `hosts.allow` must specify the hosts by their IP address, or network number/netmask.

For example, to allow NFS access to `client1.example.org` and `client2.example.org`, IP addresses `192.168.1.10` and `192.168.1.11` respectively, and the entire `128.184` network, put this in `/etc/hosts.allow`:

```
portmap: 192.168.1.10 192.168.1.11 128.184.0.0/255.255.0.0
```

Step 4.6.2. Limit exports to specific machines with specific permissions

The file `/etc/exports` controls which directories are exported for NFS mounting and the hosts that are allowed to mount them. The format is:

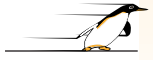
```
exported-dir     [host](options)
```

If you do not provide a host, the directory is exported to any host on the Internet, so always provide a host name, and remember that the hostname's IP address must match that given in `/etc/hosts.allow`.

Unless there is a compelling reason to do otherwise, export the directory read-only with the "`ro`" option. File permissions on the exported directory are determined by the UID/GID of the user on the remote host that mounts the directory. If the remote host is compromised, the attacker can emulate any UID/GID she wishes. Therefore, any file on a read-write exported NFS directory can be created, altered, or deleted. The administrator of an NFS server should not allow NFS access to hosts outside her administrative control except in read-only mode.

If at all possible, avoid allowing a NetWare server to mount any UNIX filesystem, Linux, Solaris, whatever. NetWare NFS services present significant security problems, such as root read-write privilege regardless of the export restrictions.

See the manual page for `exports(5)` for details on the options for `/etc/exports`.



STEP 4.7 SERVER MESSAGE BLOCK (SMB) SAMBA SERVER

The SMB protocol is the core of the Common Internet File System developed by Microsoft for file and printer sharing. The idea behind Samba is to make a UNIX server look exactly like any NT box to its clients in the network neighborhood. Setting up the Samba software itself is relatively simple, but there are a number of nuances to successfully integrating it into the office environment. Samba server administrators should review the extensive documentation that comes with the software, and the book "Samba: Integrating UNIX and Windows," Blair, SSC, Inc. ISBN 1-57831-006-7.

Step 4.7.1. Get the latest version of Samba

As of this writing, updated packages for Red Hat version 6.0 have been issued for Samba to correct security problems. Check the updates for your distribution, and make sure that you have installed Samba version 2.0.5a or later.

Step 4.7.2. Limit access to specific hosts

Edit the file `/etc/smb.conf` in the `[global]` section to set up the list of hosts that are allowed access to the Samba server and the interfaces that the Samba server will listen on:

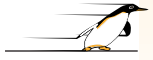
```
hosts allow = .example.org 128.184.
interfaces 192.168.0.1/24 127.0.0.1/32
bind interfaces only = true
```

Step 4.7.3. Use encrypted passwords

Before setting up Samba to use encrypted passwords, read `/usr/doc/samba-2.0.5a/docs/textdocs/ENCRYPTION.txt`.

Red Hat Linux 6.0 defaults to looking for Samba passwords in `/etc/smbpasswd`. Other distributions may use `/etc/samba.d/smbpasswd`, or a different path entirely. The path is set in the configuration file `/etc/smb.conf`. You can create a template `/etc/smbpasswd` with the following command:

```
[root]# mksmbpasswd.sh < /etc/passwd >/etc/smbpasswd
[root]# chmod 600 /etc/smbpasswd
```



Edit the template to remove entries for system accounts like bin, daemon, and ftp. The administrator should set default passwords for each of the accounts. If you wish to have your users set their own password, you will need to edit `/etc/smbpasswd` and put the string "NO PASSWORD" in the first eleven characters in the password field, leaving the remaining 21 "X" characters. Then enable null passwords with the line:

```
null passwords = true
```

in the `[global]` section of `/etc/smb.conf`, and restart the smb server. If you decide to use this method, give the users only a short period of time to reset their passwords. If any entries in `/etc/smbpasswd` contain the string "NO PASSWORD" after the deadline, then set passwords for the users yourself.

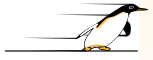
Step 4.7.4. Remove "guest" or anonymous shares

The default `/etc/smb.conf` that comes with Red Hat 6.0 only enables user-level shares of the home directories for each user on the local host. Other distributions may enable other publicly-readable guest shares. Before enabling Samba, carefully inspect the shares defined in `/etc/smb.conf` and disable any that are not absolutely necessary. For the remaining shares allow write access only when absolutely necessary. Consider setting write permissions for only those users that need the permission, not for any user connected to the service.

Step 4.7.5. Set default file creation masks

The default file creation mask makes files that are world-readable. Edit `/etc/smb.conf` and search for the following lines and change the masks to clear the "other" permission bits:

```
create mask = 0770
directory mask = 0750
```



STEP 4.8 CENTRAL SYSLOG HOST

As discussed in Step 2.8.3, a centralized logging host is important for organizations with multiple machines. They provide an additional line of defense in the preservation of information and evidence about system anomolies and break-ins. Because of this important security function, logging hosts should be as secure as possible. No other services, except perhaps SSH for remote administration, should be running, or even installed, on the machine. Only the absolute minimum set of system utilities should be installed. The host should have a large, fast disk dedicated to the `/var/log` directory for the collection of the syslog messages. A fast, high-capacity backup device is also advisable. Log rotation (see Step 2.8.4) should be turned off, or set to a long interval, and all log files should be retained for a long period of time.

As was mentioned in Step 2.8 above, there are several alternative syslog daemon implementations available that are more secure than the stock `sysklogd` provided on the Red Hat installation CD-ROM. See Appendix A under Step 2 for references to some of the alternatives.

Step 4.8.1. Configure syslogd to accept remote log messages

The default behavior of the syslog daemon in Red Hat Linux is not to accept remote log messages. This is contrary to the behavior of most BSD-style log daemons. To turn this feature on, edit `/etc/rc.d/init.d/syslog`, and add the "-r" option to the line that starts the syslog daemon:

```
start)   ...
  daemon syslogd -r
```

Then, restart the syslog service:

```
[root]# /etc/rc.d/init.d/syslog restart
```

Step 4.8.2. Configure log rotation

As discussed in Step 2.8.4, the `logrotate` program is designed to rotate, preserve, and delete log files after a certain period of time, or when the files reach a certain size. For a loghost, log rotation should be turned off (by deleting `/etc/cron.daily/logrotate`), or the logrotate configuration file should be edited to preserve the log files for a much longer period of time. For example, if your organization's security policy states that the logs must be kept for a year, edit `/etc/logrotate.conf` and change the first few lines to read:

```
# rotate log files monthly
monthly

# keep a years worth of backlogs
rotate 12
```

Red Hat Linux, and most other distributions, has a package called "`anonftp`" that sets up an anonymous FTP directory with the proper permissions for secure operation. To make sure, check in `/home/ftp`; the `bin` and `etc` directories should be owned by UID/GID root, mode 111 (execute-only). The `pub` directory should be owned by UID root, GID ftp, mode 02555 (set-GID, read-and-execute-only).

Step 4.9.1. Limit access with TCP wrappers

The FTP daemon is invoked through `inetd` and protected by TCP wrappers. If the FTP server is only meant to provide data to a limited set of machines, like your local domain or network, put the restriction in `/etc/hosts.allow`:

```
in.ftpd: .example.org 128.184.
```

A general anonymous FTP server will be accessible to the world:

```
in.ftpd: ALL
```

Note that TCP wrappers will log all connections, so that you can monitor them in the log file `/var/log/secure`. The files that are transferred are logged in `/var/log/xferlog`.

Step 4.9.2. Limit permitted operations in `/etc/ftpaccess`

The WU-FTP daemon supplied with Red Hat Linux allows fine-tuned control through the `/etc/ftpaccess` configuration file. See the manual page for `ftpaccess(5)` for more information. In this file, you can define special classes of users based on where they are connecting from, the number of simultaneous users, limit the operations allowed by classes of users, and much more.

If a writable directory is required (see below), anonymous users can be precluded from modifying the contents, regardless of the directory permissions, by putting the following into `/etc/ftpaccess`:

```
chmod          no  guest,anonymous
delete         no  guest,anonymous
overwrite      no  guest,anonymous
rename         no  guest,anonymous
```

Step 4.9.3. Protect incoming directory

In general, it is never a good idea to allow write access to an anonymous FTP directory, but sometimes it must be done. Traditionally this is called the "incoming" directory. The Washington University FTP daemon has a number of control features that will help to keep the incoming directory from turning into an illegal "warez" site.

First, create the incoming directory with write, but not read, access:

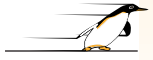
```
[root]# mkdir -m 333 /home/ftp/incoming
```

Then, edit `/etc/ftpaccess` and add these lines:

```
path-filter   anonymous   /etc/pathmsg  ^[-A-Za-z0-9._]*$ ^\.     ^-
upload        /home/ftp   /incoming     yes    root   ftp    0600  nodirs
noretrieve    /home/ftp/incoming/
```

The first line restricts upload file names to letters, numbers, hyphen, period, and underscore, and it restricts file names from starting with a period or hyphen (so the anonymous user can not create a file called "`...`" for instance). The second line says that files uploaded to the incoming directory are allowed, that files will have UID root, GID ftp, mode 0600, and that the user is not allowed to create subdirectories. The last line denies downloads from the incoming directory entirely, so once a file is written there, another anonymous user can't get it. Again, read the manual page for `ftpaccess(5)` for more information about these and other control mechanisms.

Regardless of the protections, the incoming directory should be reviewed daily, and all files stored there moved to another directory out of the anonymous directory tree. Write a cron job to check the directory each night, perform the move and notify the administrator about any files found there.



STEP 4.10 HYPERTEXT TRANSFER PROTOCOL (HTTP) SERVER

All major Linux distributions come with the Apache HTTP server software. Apache is designed for flexibility and has a wealth of features. Most security-related settings are in the main configuration files found in `/etc/httpd/conf` in Red Hat Linux. The file `httpd.conf` sets up basic operating parameters for the HTTP daemon; `access.conf` sets up basic access rules; and `srm.conf` sets up basic server configuration parameters like the document root, aliased directories, CGI script directories, and icons for indexes.

See the Apache manual included with Red Hat 6.0, `http://localhost/manual/index.html`, especially the Security Tips, `http://localhost/manual/misc/security_tips.html`, and the description of how sections work, `http://localhost/manual/sections.html`. The Apache manual is also available from the Apache Web site at `http://www.apache.org/docs/`.

Step 4.10.1. Set basic access to default deny

In `access.conf`, set the root directory access and options to:

```
<Directory />
Options None
AllowOverride None
order deny,allow
deny from all
</Directory>
```

So by default, access to all directories and files on the server is denied.

Step 4.10.2. Selectively open access to specific directories

In the remainder of `access.conf`, specify only the access and options absolutely necessary. For an Intranet server, access should be allowed only to IP addresses on the internal network:

```
<Directory /home/httpd/html>
Options None
AllowOverride None
order deny,allow
deny from all
allow from 192.168.
</Directory>
```

Step 4.10.3. Selectively allow options on specific directories

Consider the implications of allowing any of the following Options before you set them in the `<Directory>` directive:

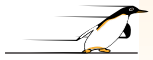
`ExecCGI`	Should only be allowed for CGI directories
`FollowSymLinks`	If users have write access to the HTML directories, they can set symbolic links to areas that contain sensitive data
`Includes`	Server side includes can be used to bypass default file access restrictions
`IncludesNOEXEC`	Safer version of Includes that disables the `#exec` statement and `#include` of CGI scripts
`Indexes`	The daemon will print a directory listing for any directory without an index file (index.html). This may expose the names of data files ordinarily hidden

Step 4.10.4. Selectively use `.htaccess` to override access control

Changing the `AllowOverride` directive from `None` to `All` tells Apache to look for a file named `.htaccess` in the directory holding the requested file, and to interpret the contents to override any of the default access and options directives for that directory. The advantage is that access and options can be set dynamically without editing the configuration file and restarting the server. The disadvantage is that security and options settings are no longer specified in a single place, but scattered around the directory structure. This can make it more difficult to audit the security policy for the Web site. In addition, there is a performance hit for every access, since the server has to look for, open and interpret the contents of the file for every request.

Step 4.10.5. Use password protection for sensitive data

Apache allows for protecting directories and/or files with username/password authentication. This provides a moderate level of security, although the username and password are passed over the network in the clear and are subject to being sniffed. See the locally installed Apache manual, `http://localhost/manual/mod/directives.html` for more information on the `AuthType`, `AuthUserFile`, and other authentication directives.



Step 4.10.6. Use SSL for secure HTTP communications

The Secure Sockets Layer (SSL) protocol provides strong authentication and end-to-end encryption for TCP sockets. It is most often used in conjunction with secure Web applications. Open source software implementations of SSL exist as compile-time extensions to the Apache web server.

Note: Administrators of commercial Web sites in the U.S. and other countries may be restricted by patent law and licensing from using SSL's RSA encryption software. Operators of commercial Web sites should seriously consider purchasing a commercial SSL-capable Web server. See Appendix A for some pointers to popular secure Web server vendors.

Note also: Commercial Web sites should have digital certificates from recognized Certificate Authorities. The two primary sources for commercial certificates are Thawte Consulting (`http://www.thawte.com`) and Verisign, Inc. (`http://www.verisign.com`).

Step 4.10.6.1. Download OpenSSL and `mod_ssl`

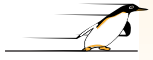
Extract the source code for the Apache server from the SRPM, or download the latest version from `ftp://ftp.apache.org`. The version numbers in the example below were the latest as of the writing of this guide. Check the FTP sites for the latest versions of each package.

```
[root]# cd /usr/local/src
[root]# wget ftp://ftp.apache.org/dist/apache_1.3.6.tar.gz
[root]# wget ftp://ftp.modssl.org/source/mod_ssl-2.3.10-1.3.6.tar.gz
[root]# wget ftp://ftp.openssl.org/source/openssl-0.9.3a.tar.gz
[root]# tar xzf apache_1.3.6.tar.gz
[root]# tar xzf mod_ssl-2.3.10-1.3.6.tar.gz
[root]# tar xzf openssl-0.9.3a.tar.gz
```

Step 4.10.6.2. Build OpenSSL

The Makefile for OpenSSL assumes that the Perl interpreter is in `/usr/bin/perl5`. Either edit the Makefile to change this to `/usr/bin/perl`, or redefine it on the command line. The latter method is illustrated below:

```
[root]# cd openssl-0.9.3a
[root]# make links PERL=/usr/bin/perl
[root]# ./Configure gcc
[root]# make
[root]# cd ..
```



▲ Step 4.10.6.3. Build Apache with `mod_ssl` module

Configuration of Apache to use `mod_ssl` can be done directly from the `mod_ssl` source directory:

```
[root]# cd mod_ssl-2.3.10-1.3.6
[root]# ./configure --with-apache=../apache_1.3.6 \
--with-ssleay=../openssl-0.9.3a --prefix=/usr/local/apache
[root]# cd ../apache_1.3.6
[root]# make
[root]# make certificate TYPE=test
[root]# make install
```

▲ Step 4.10.6.4. Start Apache with `mod_ssl` and test

Before starting the SSL version of Apache, stop the old daemon from running.

```
[root]# /etc/rc.d/init.d/httpd stop
[root]# /usr/local/apache/sbin/apachectl startssl
```

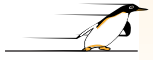
To test the server, from another host, enter:

```
[root]# netscape https://server-name/
```

After `netscape` prompts to accept the test certificate, a screen that says “It Worked!” should be displayed.

▲ Step 4.10.6.5. Read the `mod_ssl` documentation

Read `https://server-name/manual/mod/mod_ssl/` thoroughly. You may especially want to read the FAQ in this manual to learn how to create digital certificates.



Once your system has been configured to increase security, you can optimize the system for performance. Though it is beyond the scope of this document to discuss every aspect of Linux performance, a few optimizations will yield substantial performance gains.

STEP 5.1 KERNELS: THOUGHTS ABOUT CONFIGURATION, RECOMPILING, AND INSTALLING A NEW KERNEL

Like system partitioning, the use of kernel modules is somewhat of a religious debate. Red Hat and most other Linux distributions use modularized kernels to make the installation of Linux simple and recognition of hardware as plug-and-play as possible. Only the most basic operations are compiled into the kernel, and modules are loaded to implement support for specific hardware and software features. Kernel modules allow for the installation and removal of kernel features, such as IP Aliasing, without the need to re-compile the kernel or reboot the system.

On the flip side, the use of kernel modules is just one more thing that can break. Monolithic kernels only support the required hardware and features, yielding better performance. Kernel modules increase the risk of a security compromise through a rogue module. The module vs. monolith choice is yours but there are benefits to both methods.

Regardless of method, Red Hat 6.0 debuted with a stable version of the Linux 2.2.x kernel with great improvements in stability, performance and hardware support. Though it is out of the scope of this document, it is recommended that users read the KERNEL-HOWTO document and consider compiling a new kernel with built-in support for the specific hardware on the system.

Although newer kernels are generally better in terms of performance and reliability, some versions can introduce minor, and sometimes major, bugs. It's important to always keep recent backups and always test a new kernel before using it in an operational environment. Mission critical systems should stay away from the odd-numbered development kernels, 2.1.x, 2.3.x, and run only the latest version of the "stable" even numbered kernels, 2.0.x and now 2.2.x.

Out of the box, Red Hat, and most distributions, does not take full advantage of modern hardware. By minimizing optimizations, Linux will run on more legacy hardware. By tuning a step at a time, reliability and stability are maintained and performance is increased.

Step 5.2.1. TCP/IP Receive Window size

The TCP/IP acknowledge (ACK) receive window or RWIN defaults to a value of 0. For modern day Ethernet and Fast Ethernet networks, this is less than optimal. To fix this, edit `/sbin/ifup` and make the following changes:

Line 110:
```
route add -net ${NETWORK} netmask ${NETMASK} ${DEVICE}
```
to:
```
route add -net ${NETWORK} netmask ${NETMASK} window 16384 ${DEVICE}
```

Line 112:
```
route add -host ${IPADDR} ${DEVICE}
```
to:
```
route add -host ${IPADDR} window 16384 ${DEVICE}
```

Line 117:
```
route add default gw ${GATEWAY} metric 1 ${DEVICE}
```
to:
```
route add default gw ${GATEWAY} metric 1 window 16384 ${DEVICE}
```

Line 125:
```
route add default gw ${GATEWAY} ${DEVICE}
```
to:
```
route add default gw ${GATEWAY} window 16384 ${DEVICE}
```

Line 134:
```
route add default gw $gw ${DEVICE}
```
to:
```
route add default gw $gw window 16384 ${DEVICE}
```

STEP 5.3 PACKET FIREWALLS AND LINUX IP MASQUERADING

A Linux server running a well-configured firewall is one of the most effective ways to protect the local server and any internal networks behind it. Linux 2.2.x kernels have a very stable packet firewall implementation that is administrated through a tool called `ipchains`. This tool gives the administrator the flexibility to define the type of traffic allowed in and out of the firewall. Note however, that kernels prior to version 2.2.11 had a bug in the IP fragmentation code that allowed an attacker to send carefully crafted IP fragments that would bypass `ipchains` rules. We recommend that administrators of enterprise systems install the latest version of the kernel (version 2.2.12 at the time of this writing).

Firewall technology is a book in itself (for example, “Building Internet Firewalls” Chapman & Zwicky, 1995, O’Reilly & Associates, ISBN 1-56592-124-0), and far beyond the scope of this guide. See the Resouces for a URL to the IPCHAINS and IP-MASQ HOWTOs for an in-depth discussion.

Step 5.3.1. Getting more from your external connection with IP Masquerade

A firewall protects standalone servers very well but it can also protect internal computers. A very common example of this configuration would be a person or company connecting their home/office network to the Internet. A wrinkle to this setup is that TCP/IP addresses used on the Internet are becoming a scarce and valued commodity. Thus, getting TCP/IP addresses for all of your internal computers can cost quite a bit of money. Network address translation (NAT) was developed to conserve Internet TCP/IP addresses while still allowing internal computers to access the Internet. For Linux, a form of NAT was developed called IP Masquerade, which is in common use by many Linux users today.

Step 5.3.2. A strong `/etc/rc.d/rc.firewall` ruleset

See Appendix D for the complete listing of a strong packet firewall script for an IP Masqueraded machine connected to the Internet via a Ethernet connection to a DSL or Cable modem, or another persistent network connection. There are extensive embedded comments to explain what the various rules do and how to modify the rules to suit your specific needs.

Step 5.3.3. Double check, install, and test the firewall

Once you have the `/etc/rc.d/rc.firewall` ruleset copied into the machine, it **very** important to tailor it to your specific environment. Make sure that the names of the internal and external interfaces are correct. Make sure that the TCP/IP addressing scheme of your internal network is configured properly.

Next, test the ruleset to be sure that it loads without errors, does not block any specific traffic that you need for your environment, and that it loads upon every reboot of the server.

Step 5.3.3.1. Make the ruleset executable

To be able to run the ruleset, you need to make the `/etc/rc.d/rc.firewall` script executable:

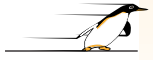
```
[root]# chmod 700 /etc/rc.d/rc.firewall
```

Step 5.3.3.2. Load the ruleset while at the console of the Linux server

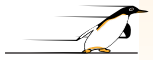
As shown in Step 2.8, monitor the console logs on `VTY7` and `VTY8`. If the Firewall script runs without any visible errors, the system should immediately begin to block traffic that you want. The console logs will show you what **is** and **isn't** blocked.

When you execute `/etc/rc.d/rc.firewall` the output of the script should look something like the following:

```
[root]# /etc/rc.d/rc.firewall
```



```
Loading IPCHAINS Firewall Version 3.12
-------------------------------------------------------------------------------
External IP: 192.168.0.14
External broadcast: 192.168.1.255
Default GW: 192.168.0.1
-------------------------------------------------------------------------------
  - Adding multicast route.
SIOCADDRT: File exists
  - Enabling IP forwarding.
  - Enabling dynamic TCP/IP address hacking.
  - Changing IP masquerading timeouts.
  - Loading masquerading modules.
  - Flushing all old rules and setting all default policies to REJECT
-------------------------------------------------------------------------------
Input Rules:
  - Setting input filters for traffic on the internal LAN.
  - Setting input filters for specific internal hosts.
  - Setting input filters for traffic from the external interface.
  - Setting input filters for public services (all interfaces).
  - Setting input filters for explicit external hosts.
  - Final input catch all rule.
-------------------------------------------------------------------------------
Output Rules:
  - Setting output filters for traffic on the internal LAN.
  - Setting output filters for specific internal hosts.
  - Setting input filters for traffic to the external interface.
  - Setting output filters for public services (all interfaces).
  - Reject specific outputs.
  - Setting output filters for explicit external hosts.
  - Final output catch all rule.
-------------------------------------------------------------------------------
Forwarding Rules:
  - Enable IP Masquerading from the internal LAN.
-------------------------------------------------------------------------------
Firewall implemented.
```



▲ Step 5.3.3.3. Test the firewall ruleset

While at the console of the Linux server, try various network applications like TELNET, FTP, SSH, etc. to computers on the Internet and on the internal LAN. If that test goes well, try a similar set of tests on one of the internal computers. If, for any reason, the machines don't respond properly, check the `/var/log/messages` file or on the console, watch `VTY7` for firewall hits.

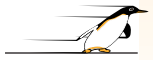
■ Step 5.3.4. Analyze a typical IPCHAINS firewall ruleset hit

So you are starting to see strange IPCHAINS entries in `/var/log/messages`. Realistically, this is happening either because your firewall ruleset is too restrictive for your environment or it is working properly and blocking unwanted traffic. For example, once the firewall ruleset is installed, DHCP no longer works. Here is the Resulting IPCHAINS firewall log:

```
Aug  8 21:00:32 testpc kernel: Packet log: output REJECT eth0 PROTO=17
192.168.0.14:68 192.168.0.1:67 L=604 S=0x00 I=41786 F=0x0000 T=64
```

There is a LOT of information in this one line so let's break it down by section. Refer back to the logged hit as you read this.

- This firewall hit occurred on Aug 8 at 9:00:32pm on the "testpc" computer
- The ruleset hit was on the "output" side of the firewall (other hits can be "input" or "forward")
- The hit was then "REJECTed" (others can say "DENY" or "ACCEPT")
- The firewall hit was on the "eth0" interface
- This hit occurred over the "PROTO=17" protocol or UDP (other common protocol numbers are "1" for ICMP and "6" for TCP. A full list of other protocol numbers can be found in `/etc/protocols`)
- The source the hit came from was IP address 192.168.0.14 on port 67, which is for DHCP client (a full list of port numbers can be found in `/etc/services`)
- The destination address was IP 192.168.0.1 to port 68 which is DHCP Server
- The packet's length was 604 bytes long
- This packet did not have any "Type of Service" (TOS) set (don't worry about TOS, it's a low-level network flag denoting how the packet should be passed through the network, a value of 0 is normal)
- The packet's "IP ID" number was 41786 (again, don't worry about IP ID, it's used to reassemble fragmented packets)
- The packet was not fragmented



- The packet had a TimeToLive (TTL) of 64 (every hop across the Internet will subtract 1 from this number, when TTL=0, the packet is dropped)
- Though not available in some earlier Linux kernel versions, if there is an additional field after TTL, it reflects the IPCHAINS rule number which caused the packet to log

Putting it all together, we see that the OUTPUT rule for "DHCP Clients" was not enabled, resulting in a REJECTed packet. If you uncomment those two lines in the firewall ruleset and run the script again, DHCP will work fine.

Step 5.3.5. Running the firewall ruleset upon every reboot

To make the `/etc/rc.d/rc.firewall` ruleset run when Linux boots, edit `/etc/rc.d/init.d/network`. Look for the case statement. At the end of the "`start)`" case (immediately before the double semicolon "`;;`") insert the following lines:

```
#Load the IPCHAINS ruleset
/etc/rc.d/rc.firewall
```

This will invoke the firewall ruleset immediately after the network interfaces are initialized, minimizing the period of time the interfaces are vulnerable.

There are many security-related tools available for monitoring the computer from the inside (host-based) and the outside (network-based). In this step, we will not present exhaustive details of the installation and operation of each tool. We just want to make you aware of the most common tools available, their general purpose and operational parameters. You are encouraged to inspect the documentation and Web sites mentioned, and experiment with the tools to craft them into an integrated, personalized toolkit.

STEP 6.1 HOST-BASED MONITORING AND INTRUSION DETECTION

Host-based monitoring and intrusion detection (ID) tools generally concentrate on reading, processing and evaluating system log files. This implies that log files contain enough information to detect intrusion attempts in the first place. See Step 2.8.1 above for setting options in `/etc/syslogd.conf` and other logging options for maximizing the information available.

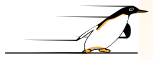
As with any ID system, the greatest problem is "false positives": results that look like intrusions, or at the very least trigger some action, when in fact the event is completely innocuous. All ID and monitoring tools require patience and persistence on the part of the administrator. As opposed to the best network access policy — default deny — the best intrusion detection policy is default allow. In other words, ignore only those events that you are sure are safe to ignore, and let the ID system report everything else. Time and experience will allow you to winnow out events that do not require your attention.

Step 6.1.1. Swatch, the Simple WATCHer

Swatch is a Perl program designed to read syslog files and generate events based on the content of the messages. Version 2.2 is supplied on the Red Hat Linux distribution media. Version 3.0 is in Beta testing at the time of this writing, available from `ftp://ftp.stanford.edu/general/security-tools/swatch/`. This version is a complete rewrite taking greater advantage of the features of Perl version 5, and with a completely revamped configuration file for scanning the syslog files.

The control file for swatch version 2.2 consists of a list of Perl regular expressions followed by "actions." See the manual page `swatch(5)` for a detailed description of the file format. Here is a very simple control file that will list only the lines in the syslog file that have the word "refused" in them:

```
/refused/       echo=bold
/.*/            ignore
```



To check /var/log/messages from the command line, do this:

```
[root]# swatch -c testrc -f /var/log/messages
*** swatch-2.2 (pid:32381) started at Fri Jul 25 15:12:08 EDT 1999
Jul 25 10:32:42 localhost sshd[16508]: refused connect from 192.168.1.1
Jul 25 10:33:31 localhost sshd[16530]: log: Rsa authentication refused
for root: bad modes for /root/.ssh/authorized_keys
Jul 25 20:11:22 localhost sshd[2312]: error: Fwd X11 connection from
127.0.0.1 refused by tcp_wrappers.
```

A good starting point for a control file under Red Hat Linux is /usr/doc/swatch-2.2/config_files/swatchrc.personal. It consists of a list of regular expressions that generate actions, followed by a list of regular expressions to ignore, and the final action is to simply echo everything else. This file can be tuned by adding actions and ignores and reiterating through /var/log/messages until you get results that are manageable.

Swatch can be configured to run through a single log file in a single pass, or it can be run as a daemon that continually monitors new log messages as they are generated. It can be configured with different control files to monitor mail logs, firewall logs, Samba logs, etc. See the manual page for swatch(8) for more information.

Step 6.1.2. Psionic Logcheck

Another of the many log file monitor programs is Logcheck from Psionic Software Systems. See Appendix for URLs to the Psionic Logcheck home page and other logfile monitors in the MetaLab Web site.

Logcheck comes with good documentation. Do a thorough read of the INSTALL text file before installing and running it. Installation is simple. After unpacking the compressed tar file, execute "make linux" in the source directory to copy the Linux-specific configuration files and script to /usr/local/etc/.

Logcheck generates reports that have three sections: active system attacks, security violations, and unusual system events. Active system attack notices are generated when a syslog entry matches any pattern in logcheck.hacking; security violations are generated from matches to patterns in logcheck.violations; and unusual events are generated by any line that does not match a pattern in logcheck.ignore. Read the comments in logcheck.sh for more detailed descriptions of these configuration files.

Like swatch, the administrator needs patience and persistence to weed out records that do not indicate genuine events and add those patterns to `logcheck.ignore`. At first, there are likely to be many false alarms. Investigate each reported instance to determine if, in fact, the reported event requires attention. If it does not and never will, a pattern to match that event can be put into `logcheck.ignore` and you will not see it reported again. It is best to make the pattern as specific as possible, so that unusual variations of the event are reported.

Step 6.1.3. Tripwire

Tripwire was originally developed by Gene Kim and Gene Spafford at the University of Purdue COAST Laboratory. The idea behind Tripwire is that in order to successfully compromise a system, some system files must be changed, added, or deleted in order to place back doors, trojans and exploit scripts. Tripwire generates a database of cryptographic signatures for important system binaries and configuration files and reports changes in any of these files over time.

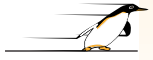
See appendix A for URLs for the source code for the Tripwire "Academic Source Release" version 1.3.1 and version 2.0 for Red Hat Linux. However it must be noted that Tripwire 2.0 is not supported for Red Hat 6.0, only for the earlier versions 5.2 and 5.1. As of the time of this writing, only the ASR 1.3.1 release works for Red Hat 6.0.

Read the documentation that comes with each release carefully for information about compiling, installing, and maintaining Tripwire.

Step 6.1.3.1. Tripwire databases

Before Tripwire, the only way to test the integrity of system files was by checking CRC checksums. Unfortunately, CRC 16- and 32-bit checksums are easily subverted. Tripwire builds a database of cryptographic checksums, such as MD5 and Snefru, that are nearly impossible to spoof and checks them regularly. If the checksum test fails, it is a very strong indication that the file in question has been modified.

The single greatest weakness in Tripwire was that, if an attacker can gain root access to the machine, he can modify or rebuild the Tripwire database to mask his intrusion. The only defense against this attack is to place the database on read-only media, such as a write-protected floppy or CD-R. One of the most significant innovations in Version 2 of Tripwire is cryptographic signing of the database and configuration files, which eliminates the need for placing Tripwire databases on read-only media. Hopefully, support for Red Hat 6.0 will be available by the time this guide goes to press.



▲ Step 6.1.3.2. Running Tripwire

The first time Tripwire is run, it builds a database of checksums and other information (UID, GID, mode, size, etc) for the files and directories listed in the configuration file. Then, each time it runs, it compares the information in the database to the current state of the system. When run interactively from the command line, it gives you the option of updating the database for each changed file it finds. So, if `/etc/group` has changed, and you were the one that changed it, you can update the database so that you will not be notified until the next time it changes (perhaps by an attacker).

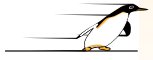
▲ Step 6.1.3.3. Use rpm to verify package files

The database built by the Red Hat Package Manager (`rpm`) when it installs packages includes MD5 checksums, UID, GID, mode, size, etc. You can use `rpm` to verify the contents of the database against the current state of the files on disk and report any files that are different. This is a quick check for files that have changed.The disadvantage of this method, as opposed to the Tripwire method, is that with rpm verification you can not test changes over time, only since the files were installed. There is no provision for updating database entries that have changed with the knowledge and approval of the administrator.

To verify all package files, run:

```
[root]# rpm -Va
S.5....T c /etc/exports
S.5....T c /etc/hosts.allow
S.5....T c /etc/hosts.deny
.M...UG.   /mnt/cdrom
```

The entries for the files in `/etc` tell you that the size, MD5 checksum, and modification time have changed; for `/mnt/cdrom` the mode, user and group have changed.



Step 6.1.4. Psionic PortSentry

There are several tools for detecting port scans, Psionic PortSentry adds a twist: when a scan is detected it can take immediate action to block the scanning host from any further access to the machine. It works well in conjunction with Psionic Logcheck (Step 6.1.2). See Appendix A for a URL to the home page.

PortSentry comes with good documentation. Before compiling and installing, read `README.install`, `README.methods`, and `README.qa` thoroughly. PortSentry compiled "out-of-the-box" for the authors, without any changes necessary. The program and configuration files are installed in `/usr/local/psionic/portsentry` by default.

PortSentry can detect not only simple TCP connect-type scans, but many types of stealth scans, too, such as SYN/half-open, FIN, and NULL scans.

Before running PortSentry, review `portsentry.conf`. All the entries are documented in the configuration file, and in `README.install`. By default, the only action taken against scanning hosts is to place:

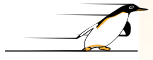
```
ALL: <attacker-IP>
```

at the end of `/etc/hosts.deny`. Since you should already have "`ALL: ALL`" at the end of `/etc/hosts.deny`, this is ineffective. In fact, this method is completely ineffective if any services have been allowed to the attacking host in `/etc/hosts.allow`, since TCP wrappers read `/etc/hosts.allow` for a match before reading `/etc/hosts.deny`.

On the other hand, using the `KILL_ROUTE` option in `portsentry.conf` is highly effective if configured correctly. There are two basic approaches to the `KILL_ROUTE` option: 1) add an entry in the routing table to send responses to the attacking host to a fictitious destination; and 2) add firewall packet filter rules with `ipchains/ipfwadm` to drop all packets from the attacking host. Adding a bogus route to the routing table may still allow some UDP-based attacks to get through (see `README.install` for more information). The preferred method is packet filtering with `ipchains`:

```
KILL_ROUTE="/sbin/ipchains -I input -s $TARGET$ -j DENY -l"
```

Note that adding automatic packet filtering leaves open the possibility of a denial-of-service attack, where the attacker spoofs a scan from another host in order to block that host from legitimate access.



Step 6.2.1. Tiger, the Texas A&M system checker

Tiger is a tool that inspects security-related settings of UNIX computers. Tiger was just recently upgraded (for the first time since 1994) to support checking of Red Hat Linux systems. See Appendix A for a URL to the Tiger home page and the source code.

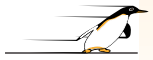
Step 6.2.2. Install and configure Tiger

Installing is simple: unpack the compressed tar file and execute "`make install`". This places the configuration and script files in `/usr/local/tiger`. The `Makefile` does not create the intermediate directories Tiger relies on though, so after installation, execute the following:

```
[root]# mkdir -m 700 -p /var/spool/tiger/bin
[root]# mkdir -m 700 -p /var/spool/tiger/logs
[root]# mkdir -m 700 -p /var/spool/tiger/work
```

Also, if you are running Bash version 2 as the default shell, you will need to edit `/usr/local/tiger/systems/Linux/2/config` and comment out the line that starts with "`GROUPS=`". In Bash 2, `GROUPS` is a read-only variable, and the script will abort if it tries to assign to it.

The checks that Tiger performs are set out in `/usr/local/tiger/tigerrc`. There are several sample `tigerrc` files in the source directory, and they are well commented. The default `tigerrc` pretty much tries to test everything it can. For Red Hat Linux version 6, most checks will work as advertised, except signature checking. You can turn this off by editing `tigerrc` and changing `Tiger_Check_SIGNATURES=Y to Tiger_Check_SIGNATURES=N`. By the same token, you can turn off running Crack on the password file by editing the entry for `Tiger_Run_CRACK`.



Step 6.2.3. Running Tiger

To run Tiger:

```
[root]# cd /usr/local/tiger
[root]# ./tiger
```

A number of messages are printed on standard output while Tiger runs through its checks. When the run is complete, the results are stored in `/var/spool/tiger/logs/security.report.<host>.<domain>.<date>-<time>`. Be prepared to have a lot of messages in the first security report. Reported problems look like:

```
—WARN—  [acc001w] Login ID postgres is disabled, but still has a valid shell
        (/bin/bash).

—FAIL—  [perm007f] /etc/aliases should not have group read.
—FAIL—  [perm007f] /etc/aliases should not have world read.
```

A WARN entry is something Tiger considers undesirable, but not terrible. A FAIL entry is something Tiger thinks you should change. The string in square brackets indicates how to get more information about the specific warning with the `tigexp` program:

```
[root]# ./tigexp acc001w

The listed login ID is disabled in some manner ('*' in passwd field, etc),
but the login shell for the login ID is a valid shell (from /etc/shells or
the system equivalent).  A valid shell can potentially enable the login ID
to continue to be used.  The login shell should be changed to something
that doesn't exist, or to something like /bin/false.
```

Step 6.2.4. Changing Tiger checks

Going through the security report, you will probably find some warnings and failures that you think are perfectly fine. The base information Tiger uses for comparison can be found in `/usr/local/tiger/systems/Linux/2`. For example, the file `file_access_list` contains the suggested ownership and mode bits for a number of system files. If you prefer that `/etc/aliases` is allowed group- and world-read permissions, you can edit this file to set the proper mode bits, and you will no longer get a FAIL message for those permissions.

Once again, patience and persistence are needed here. Follow the suggested changes in the security report and judiciously modify the base information until Tiger's security reports return the most meaningful information for your site.

Step 6.2.5 TARA, an updated version of Tiger

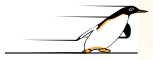
Because Tiger has only recently seen any new development, Advanced Research Corporation has taken the base code and enhanced it to produce the Tiger Analytical Research Assistant (TARA). See Appendix A for a URL to the TARA home page.

Installation, configuration, and operation of TARA is almost exactly the same as the original Tiger, even down to the same directory names. Note that the default `tigerrc` file enables far fewer system checks than Tiger. Review this file before running TARA and enable the checks that you think are relevant.

STEP 6.3 NETWORK-BASED VULNERABILITY ANALYSIS: LOOKING FROM THE OUTSIDE IN

There are many, many tools designed to scan hosts on a network for open ports, general security vulnerabilities, or specific exploitable services. Before any network scanning is done, the most important thing to secure is your own backside... get authorization to perform the scanning, in writing if possible. Most government and commercial organizations have specific policies for who can perform scanning, the scope of the scanning, and what can and can not be scanned for. Unauthorized violation of these policies can lead to sanctions, unemployment, and even legal action.

After you get authorization, post notice that a scan is going to be done and when. Some administrators are of the opinion that letting others know about an impending scan gives users an opportunity to plug holes beforehand, skewing the results of the scan. Well, at least the results are skewed in the right direction, and you will avoid problems with local users' intrusion detection programs unexpectedly raising red flags.



Before the scan, test the scanning software on an isolated network segment to make sure that the scanning tool does not have an adverse effect on the machines it is scanning. Some scanners are very efficient at crashing computers.

After the scan, disseminate information about the vulnerabilities that were found on a need-to-know basis. Results of vulnerability scans should be treated like any other sensitive information in your organization. Don't post the results on the company Intranet Web server. Summarize the results and discuss them with each responsible party privately.

Step 6.3.1. SATAN derivatives: SARA and SAINT

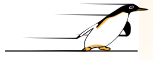
The Security Administrator Tool for Analyzing Networks (SATAN) caused quite a stir when it was released in 1994 by Dan Farmer and Wietse Venema. It had an innovative design for the user interface, using HTML and Web technology to effect a simple point-and-click process for scanning from 1 to thousands of hosts at one time.

The original release scanned for well-known vulnerabilities and exploits. The design, however, allowed for enhancing it with other checks and tests rather easily, and the authors assumed that it would be developed quickly. Two updated versions of SATAN are currently available: the Security Auditor's Research Assistant (SARA); and the Security Administrator's Integrated Network Tool (SAINT).

Both versions are under active development, in fact one of the original authors of SAINT is the currently the developer of SARA, and the systems track each other. Be sure to get the latest version of the program, earlier versions are known to have problems compiling under Red Hat 6.0. To compile and run either program, unpack the compressed tar file and execute "`perl reconfig`" to set up paths to the Perl interpreter and Web browser, and "`make linux`" to compile the scanner and its support programs.

When you invoke the scanner (`sara` or `saint`), it will invoke your Web browser and load the main page. Scanning proceeds from Data Management, to select the database to store results in, to Target Selection, to pick the host(s) to scan, to Report and Analysis. When the scan is over, results can be displayed by Danger Level, Type of Vulnerability, or Vulnerability Count.

There is much more to both systems, of course. Consult the documentation for more information.



Step 6.3.2 Nessus

Nessus (`http://www.nessus.org`) is an open-source security scanner that uses a client/server architecture. The Nessus daemon, `nessusd`, runs the scans and checks for vulnerabilities under the control of the client, which is written with a Gtk GUI interface. Nessus has a library of "plugins" that perform vulnerability tests. As of this writing there were 208 plugins, testing such things as IMAP and rpc.mountd overflows, CGI script bugs, RPC services, WinGate proxy holes, and a series of denial-of-service attacks, just to name a few.

Compiling and installing should be as simple as:

```
[root]# ./configure; make; make install
```

The `nessusd` server must be run first. The first time `nessusd` runs it computes a host key and builds a default users file `/usr/local/nessus/nessus.users`. This file controls who may connect to the server. To add an entry for root, run `nessusd` again as:

```
[root]# nessusd -P root,pass
```

where `pass` is an initial password. Starting the client, `nessus`, you first must connect to the server. Enter "`root`" in the "`Login:`" window and click "`Log in now!`". You will be prompted for the password entered above only the first time you log in, after that a private key is generated and used for all subsequent logins.

Note that Nessus tests for denial of service exploits by default, and our tests show that you are very likely to crash just about every machine you scan if this option is enabled. Before performing a scan, go to the Plugins tab and disable Denial of Service.

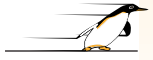
After a scan, Nessus pops up a report window with information for each scanned host. Security problems are highlighted with a red bullet, areas of concern are labeled with a black bullet. In both cases, a brief description is printed. Reports can be saved to disk for later study and comparison.

Step 6.3.3 Nmap port scanner

Nmap scans hosts for open ports. Unlike SARA/SAINT and Nessus, it does not test for specific vulnerabilities. Nmap is freely available from http://www.insecure.org/nmap/. There are a lot of scanning options: TCP connect() scanning, SYN half-open, FIN, NULL, UDP, ICMP, and SYN/FIN with IP fragments to bypass packet filters. It can try to identify the OS type of the remote host by using TCP/IP stack fingerprinting.

Compiling and installing is as easy as running make in the source directory. To scan localhost for all open TCP and UDP ports, for example to double-check your work from Steps 3.1, 3.2, and 3.3, execute:

```
[root]# ./nmap -sT -sU 127.0.0.1
WARNING:  -sU is now UDP scan — for TCP FIN scan use -sF

Starting nmap V. 2.2-BETA4 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on localhost (127.0.0.1):
Port    State   Protocol   Service
21      open    tcp        ftp
22      open    tcp        ssh
23      open    tcp        telnet
25      open    tcp        smtp
80      open    tcp        http
98      open    tcp        linuxconf
111     open    udp        sunrpc
111     open    tcp        sunrpc
113     open    tcp        auth
608     open    udp        sift-uft
610     open    tcp        npmp-local
618     open    udp        unknown
627     open    udp        unknown
629     open    tcp        unknown
632     open    udp        unknown
634     open    tcp        ginad
637     open    udp        unknown
639     open    tcp        unknown
1024    open    udp        unknown
1024    open    tcp        unknown
1026    open    udp        unknown
2049    open    udp        nfs

Nmap run completed — 1 IP address (1 host up) scanned in 5 seconds
```



Of course, if your host has these many ports open, you still have some work to do. To try to guess the operating system of the host at 192.168.1.1, execute:

```
[root]# ./nmap -O 192.168.1.1

Starting nmap V. 2.2-BETA4 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on garth (192.168.1.1):

Port    State   Protocol   Service
21      open    tcp        ftp
23      open    tcp        telnet
25      open    tcp        smtp
111     open    tcp        sunrpc
113     open    tcp        auth
515     open    tcp        printer
6000    open    tcp        X11

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=6118242 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.1.132; 2.2.0-pre1 - 2.2.2

Nmap run completed — 1 IP address (1 host up) scanned in 2 seconds
```

Nmap version 2.2-BETA4 (the latest version as of this writing) comes with a Gtk GUI front-end called `nmapfe`. The GUI makes it easy to select the host(s) or network(s) to scan and the options for scanning. The output from the nmap run is printed in its own window and can be stored in a log file for later comparison and study.

Step 6.3.4 Commercial products

Internet Security Systems (`http://www.iss.net`) markets commercial tools for host-based security assessment (System Scanner™), host- and network-based intrusion detection (RealSecure™), and network vulnerability scanning (Internet Scanner™). ISS products are mature professional tools. Evaluation copies of System Scanner and Internet Scanner are available free for download. The evaluation copies work only on the localhost.

Network Associates, Inc. (`http://www.nai.com`) markets CyberCop Scanner, which performs network vulnerability scanning for hosts, firewalls, hubs and switches. CyberCop is extensible through the Custom Audit Scripting Language (CASL). It has an automatic update feature to keep the software and vulnerability database up to date.

APPENDIX A
Resources and References

The contents of all the Appendices in this guide are available on the World Wide web at http://www.sans.org/linux.htm.

Step 2:

Installation HOWTO:	http://metalab.unc.edu/pub/Linux/docs/HOWTO/Installation-HOWTO
TrinityOS	http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html
Secure-syslog	http://www.core-sdi.com/ssyslog/
Syslog-ng	http://www.balabit.hu/products/syslog-ng.html
Nsyslogd	http://coombs.anu.edu.au/~avalon/nsyslog.html

Step 3:

CIAC login banners information bulletin	http://ciac.llnl.gov/ciac/bulletins/j-043.shtml
Linux Journal article on TCP Wrappers	http://linuxjournal.com:8080/lj-issues/issue40/2180.html
Linux software archives:	
Metalab:	http://metalab.unc.edu/pub/Linux/
Freshmeat:	http://www.freshmeat.net/
AutoRPM home page	http://www.kaybee.org/~kirk/html/linux.html
Linux Security HOWTO	http://metalab.unc.edu/pub/Linux/docs/HOWTO/Security-HOWTO.html
Linux Administrator's Security Guide	http://www.securityportal.com/lasg/
RedHat Security Information	http://www.redhat.com/corp/support/docs/errata.html (Select General Errata for your version, look for "security" in the Web page)

Subscribe to Red Hat mailing list for security and update announcements:

```
$ Mail -s subscribe redhat-announce-list@redhat.com </dev/null
```

Caldera Systems Security Advisories	http://www.calderasystems.com/news/security/index.html
Caldera-announce mailing list	http://www.calderasystems.com/support/forums/caldera-announce.html
SUSE Electronic mail security alerts	http://www.suse.com/MailingLists/index.html
Debian Security Information	http://www.debian.org/security/

Debian Mailing list for security and update announcements:

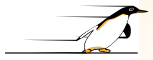
```
$ Mail -s subscribe debian-security-announce-request@lists.debian.org </dev/null
```

Step 4:

SSH	ftp://ftp.cs.hut.fi/pub/ssh
BIND	http://www.isc.org/view.cgi?/products/BIND/index.phtml
named chroot jail	http://www.linux.com/security/articles/july/sec_bind.phtml
Sendmail	http://www.sendmail.org/current-release.html
Qmail	http://www.qmail.org
Postfix	http://www.porcupine.org
Eudora Qpopper	http://www.eudora.com/free/servers.html
Cyrus IMAPD	http://asg.web.cmu.edu/cyrus/imapd/
sendmail relaying	http://www.sendmail.org/tips/relaying.html
`stunnel`	http://mike.daewoo.com.pl/computer/stunnel/
`sslwrap`	http://www.rickk.com/sslwrap/
LPRng	http://www.astart.com/LPRng.html
Samba	http://www.samba.org
Apache	http://www.apache.org
Commercial secure web servers:	
Covalent Raven SSL Module:	http://www.covalent.net/raven/ssl/
Red Hat Linux E-Commerce Server:	http://store.redhat.com/commerce/
Roxen:	http://www.roxen.com/
Stronghold:	http://www.c2.net/products/sh2/index.php3
Zeus:	http://www.zeustech.net/

Step 5:

Linux kernels	http://www.kernel.org
	http://www.kernelnotes.org
IP Masquerade HOWTO	http://ipmasq.cjb.net
IP CHAINS HOWTO	http://www.rustcorp.com/linux/ipchains/

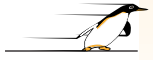


Step 6:

Psionic Logcheck	http://www.psionic.com/abacus/logcheck/
Other syslog filters	http://metalab.unc.edu/pub/Linux/system/admin/log/
Tripwire Security home page	http://www.tripwiresecurity.com
Tripwire ASR 1.3.1	http://www.tripwiresecurity.com/products/ASR1_3.html
Tripwire 2.0	http://www.tripwiresecurity.com/products/2_0Linux.html
TARA home page	http://www.arc.com/tara/
Psionic PortSentry	http://www.psionic.com/abacus/portsentry/
Tiger home page	http://www.net.tamu.edu/network/tools/tiger.html
Tiger source code	http://www.net.tamu.edu/ftp/security/TAMU/
SAINT home page	http://www.wwdsi.com/saint/
SARA home page	http://www.arc.com/sara/
Nmap	http://www.insecure.org/nmap/

Books and articles for installation, administration, advanced security, etc.

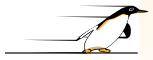
"Building Internet Firewalls," Chapman & Zwicky, 1995, O'Reilly & Associates, ISBN 1-56592-124-0
"DNS and BIND," 3rd Edition, Ablitz and Liu, O'Reilly and Associates, 1998
"sendmail," Costales and Allman, O'Reilly & Associates, 1997
"Samba: Integrating UNIX and Windows," John D. Blair, SSC, Inc. ISBN 1-57831-006-7



APPENDIX B
Stock Red Hat 6.0 /etc/inetd.conf

The following is the default `/etc/inetd.conf` file installed with Red Hat 6.0. Any line not preceded by a comment character, "#", is available — and vulnerable — to the Internet. See Step 3.1 for more information.

```
#
# inetd.conf                    This file describes the services that will be available
#                               through the INETD TCP/IP super server.  To re-configure
#                               the running INETD process, edit this file, then send the
#                               INETD process a SIGHUP signal.
#
# Version:      @(#)/etc/inetd.conf   3.10 05/27/93
#
# Authors:      Original taken from BSD UNIX 4.3/TAHOE.
#                               Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
# Modified for Debian Linux by Ian A. Murdock <imurdock@shell.portal.com>
#
# Modified for RHS Linux by Marc Ewing <marc@redhat.com>
#
# Further modified by Olaf Kirch <okir@caldera.com> for Caldera Open Linux
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
# To re-read this file after changes, just do a 'killall -HUP inetd'
#
# Note: builtin UDP services now silently drop packets from ports < 512.
#echo           stream     tcp       nowait      root        internal
#echo           dgram      udp       wait        root        internal
#discard        stream     tcp       nowait      root        internal
#discard dgram  udp        wait      root        internal
#daytime        stream     tcp       nowait      root        internal
#daytime dgram  udp        wait      root        internal
#chargen        stream     tcp       nowait      root        internal
#chargen dgram  udp        wait      root        internal
#time           stream     tcp       nowait      root        internal
#time           dgram      udp       wait        root        internal
#
# These are standard services.
#
ftp             stream     tcp       nowait      root        /usr/sbin/tcpd in.ftpd -l -a
telnet          stream     tcp       nowait  root            /usr/sbin/tcpd   in.telnetd
#gopher         stream     tcp       nowait  root            /usr/sbin/tcpd   gn
```



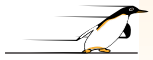
APPENDIX B
Stock Red Hat 6.0
/etc/inetd.conf

```
# do not uncomment smtp unless you *really* know what you are doing.
# smtp is handled by the sendmail daemon now, not smtpd.  It does NOT
# run from here, it is started at boot time from /etc/rc.d/rc#.d.
#smtp           stream     tcp       nowait      root        /usr/bin/smtpd   smtpd
#nntp           stream     tcp       nowait      root        /usr/sbin/tcpd   in.nntpd
#
# Shell, login, exec and talk are BSD protocols.
#
shell           stream     tcp       nowait      root        /usr/sbin/tcpd   in.rshd
login           stream     tcp       nowait      root        /usr/sbin/tcpd   in.rlogind
#exec           stream     tcp       nowait      root        /usr/sbin/tcpd   in.rexecd
#talk           dgram      udp       wait        nobody.tty  /usr/sbin/tcpd   in.talkd
#ntalk          dgram      udp       wait        nobody.tty  /usr/sbin/tcpd   in.ntalkd
#dtalk          stream     tcp       wait        nobody.tty  /usr/sbin/tcpd   in.dtalkd
#
# Pop and imap mail services et al
#
#pop2           stream     tcp       nowait      root        /usr/sbin/tcpd   ipop2d
#pop3           stream     tcp       nowait      root        /usr/sbin/tcpd   ipop3d
#imap           stream     tcp       nowait      root        /usr/sbin/tcpd   imapd
#
# The Internet UUCP service.
#
#uucp           stream     tcp       nowait      uucp        /usr/sbin/tcpd   /usr/sbin/uucico-
l
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#
#tftp           dgram      udp       wait        root        /usr/sbin/tcpd   in.tftpd
#bootps         dgram      udp       wait        root        /usr/sbin/tcpd   bootpd
#
# cfinger is for GNU finger, which is currently not in use in RHS Linux
#
finger          stream     tcp       nowait      nobody      /usr/sbin/tcpd   in.fingerd -u
#cfinger        stream     tcp       nowait      root        /usr/sbin/tcpd   in.cfingerd
#
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
#
#systat         stream     tcp       nowait      nobody      /usr/sbin/tcpd    /bin/ps -auwwx
#netstat        stream     tcp       nowait      nobody      /usr/sbin/tcpd    /bin/netstat —inet
#
# Authentication
#
#auth           stream     tcp       nowait      root        /usr/sbin/tcpd in.identd -i -t30
auth            stream     tcp       nowait      nobody      /usr/local/sbin/oidentd -i
swat            stream     tcp       nowait.     400 root    /usr/sbin/tcpd swat
#
# End of inetd.conf
```

APPENDIX C
Red Hat SYSH init script for the SSH daemon

The following SYSV init script follows the general conventions for Red Hat 6.0 init scripts as outlined in the online document `/usr/doc/initscripts-4.16/sysvinitfiles`.

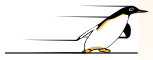
```
##!/bin/bash
#
#               /etc/rc.d/init.d/sshd
#
# sshd                          Start the Secure Shell daemon
#
# chkconfig: 345 50 50
# description:  The Secure Shell daemon, sshd, allows for strong \
#               authentication of, and encrypted communications with, \
#               remote clients using the Secure Shell Protocol
# processname:  sshd
# pidfile:      /var/run/sshd.pid
# config:       /etc/sshd_config

# Source function library.
. /etc/rc.d/init.d/functions

SSHD=/usr/local/sbin/sshd
SSHD_CONFIG=/etc/sshd_config

case "$1" in
   start)
               echo -n "Starting SSH services: "
               if [ -x $SSHD -a -f $SSHD_CONFIG ]
               then
                                    daemon $SSHD
               else
                                    echo_failure
               fi
               echo
               touch /var/lock/subsys/sshd
               ;;
   stop)
               echo -n "Shutting down SSH services: "
               killproc sshd
               echo
               rm -f /var/lock/subsys/sshd
               ;;
   status)
               status sshd
               ;;
   restart)
               $0 stop; $0 start
               ;;
   reload)
               killall -HUP sshd
               ;;
   *)
               echo "Usage: sshd {start|stop|status|reload|restart}"
               exit 1
               ;;
esac
```



APPENDIX D
A strong packet firewall ruleset

```
#!/bin/sh
#------------------------------------
# Author: David A. Ranch
# Based on the TrinityOS IPCHAINS firewall v3.12
# http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html
#------------------------------------
# This configuration assumes the following (DSL / Cablemodem setup):
#
#         1)  The external interface is running on "eth0"
#         2)  The external IP address is dynamically assigned
#         3)  The internal IP Masqueraded network interface is "eth1"
#         4)  The internal network is addressed within the private
#             192.168.1.x TCP/IP addressing scheme per RFC1918
#
#    ****
#    NOTE:    All 2.2.x Linux kernels prior to 2.2.11 have a fragmentation
#    ****     bug that renders all strong IPCHAINS rulesets void.  It
#             is CRITICAL that users upgrade the Linux kernel to 2.2.11+
#             for proper firewall security.
#
#------------------------------------

#**********************************************************************
# Initializing
#**********************************************************************
echo -e "\n\nLoading IPCHAINS Firewall Version 3.12"
echo "------------------------------------"

#------------------------------------
# Variables
#------------------------------------

# The loopback interface and address
LOOPBACKIF="lo"
LOOPBACKIP="127.0.0.1"

# External interface device.
#
# NOTE:  PPP and SLIP users will want to replace this interface
#        with the correct modem interface such as "ppp0" or "sl0"
#
EXTIF="eth0"
echo External Interface: $EXTIF
```

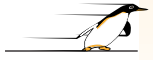
```
# IP address of the external interface
#
# NOTE: Red Hat users of DHCP to get TCP/IP addresses (Cablemodems, DSL, etc)
#       will need to install and use a different DHCP client than the stock
#       client called "pump".  One recommended DHCP client is called "dhcpcd"
#       and can found in Appendix A.
#
#       The stock Red Hat DHCP client doesn't allow the ability to have scripts
#       run when DHCP gets a TCP/IP address.  Specifically, DHCP delves out
#       TCP/IP addresses to its clients for a limited amount of time; this
#       called a "lease".  When a DHCP lease expires, the client will query the
#       DHCP server for a lease renewal.  Though the DHCP client will usually
#       get back its original TCP/IP address, this is NOT always guaranteed.
#       With this understood, if you receive a different TCP/IP address than
#       the IPCHAINS firewall was configured for, the firewall will block ALL
#       network access in and out of the Linux server because that was what it
#       was configured to do.
#
#       As mentioned above, the key to solve this problem is to use a DHCP
#       client program that can re-run the /etc/rc.d/rc.firewall ruleset once a
#       new TCP/IP address is set.  The new ruleset will make the required
#       changes to the rulesets to allow network traffic from and to your new
#       TCP/IP address.
#
#       With the dhcpcd program, it will need to executed with the following
#       command line option to have the firewall ruleset re-run upon every DHCP
#       lease renew:
#
#                                -c /etc/rc.d/rc.firewall
#
# Static TCP/IP addressed users: For EXTIP, EXTBROAD, and EXTGW, simply replace
# the pipelines with your correct TCP/IP address, broadcast address, and
# external gateway, respectively.
#
# eg:   EXTIP="172.22.43.23"
#
EXTIP=`/sbin/ifconfig | grep -A 4 $EXTIF | awk '/inet/ { print $2 } ' | sed -e s/addr://`
echo External IP: $EXTIP
```



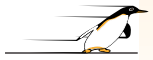
```
# Broadcast address of the external network
EXTBROAD=`/sbin/ifconfig | grep -A 1 $EXTIF | awk '/Bcast/ { print $3 }' | sed -e s/Bcast://`
echo External broadcast: $EXTBROAD

# Gateway for the external network
EXTGW=`/sbin/route -n | grep -A 4 UG | awk '{ print $2}'`
echo Default GW: $EXTGW
echo " -- "

# Internal interface device.
INTIF="eth1"

# IP address on the internal interface
INTIP="192.168.1.1"
echo Internal IP: $INTIP

# IP network address of the internal network
INTLAN="192.168.1.0/24"

# IP Port Forwarded Addresses
#
# IP address of an internal host that should have external traffic forwarded to
# Port forwarding allows external traffic to directly connect to an INTERNAL
# Masq'ed machine. An example need for port forwarding is the need for external
# users to directly contact a WWW server behind the MASQ server.
#
# NOTE:  Port forwarding is well beyond the scope of this documentation to
#        explain the security issues implied in opening up access like this.
#        Please see Appendix A to find the IP-MASQ-HOWTO for a full explanation.
#
# Disabled by default.
#PORTFWIP="192.168.1.20"

# IP Mask for all IP addresses
UNIVERSE="0.0.0.0/0"

# IP Mask for broadcast transmissions
BROADCAST="255.255.255.255"

# Specification of the high unprivileged IP ports.
UNPRIVPORTS="1024:65535"

# Specification of X Window System (TCP) ports.
```

```
XWINDOWS_PORTS="6000:6010"

# The TCP/IP address of your slave DNS servers (if any).
# This is OPTIONAL!
#
# Disabled by default.
#SECONDARYDNS="172.31.44.22"

# The TCP/IP addresses of a specifically allowed EXTERNAL hosts
#
# Disabled by default.
#SECUREHOST="172.20.2.11"
#SECUREHOST2="172.20.2.12"

# TCP/IP addresses of INTENRAL hosts network allowed to directly
#         connect to the Linux server.  All internal hosts are allowed
#         per default.
#
# Disabled by default
#HOST1IP="192.168.1.10"
#HOST2IP="192.168.1.11"

# Logging state.
#
# Uncomment the " " line and comment the "-l" line if you want to
# disable logging of some of more important the IPCHAINS rulesets.  #
# The output of this logging can be found in the /var/log/messages
# file.  It is recommended that you leave this setting enabled.
# If you need to reduce some of the logging, edit the rulesets and
# delete the "$LOGGING" syntax from the ruleset that you aren't
# interested in.
#
# LOGGING=" "
LOGGING="-l"

echo "————————————————————————————————————"
```

```
#————————————————————————————
# Debugging Section
#————————————————————————————
# If you are having problems with the firewall, uncomment the lines
# below and then re-run the firewall to make sure that the firewall
# is not giving any errors, etc.  The output of this debugging
# script will be in a file called /tmp/rc.firewall.dump
#————————————————————————————
#
#echo "  - Debugging."
#echo Loopback IP: $LOOPBACKIP > /tmp/rc.firewall.dump
#echo Loopback interface name: $LOOPBACKIF >> /tmp/rc.firewall.dump
#echo Internal interface name: $INTIF >> /tmp/rc.firewall.dump
#echo Internal interface IP: $INTIP >> /tmp/rc.firewall.dump
#echo Internal LAN address: $INTLAN >> /tmp/rc.firewall.dump
#echo ————————————————————————- >> /tmp/rc.firewall.dump
#echo External interface name: $EXTIF >> /tmp/rc.firewall.dump
#echo External interface IP: $EXTIP >> /tmp/rc.firewall.dump
#echo External interface broadcast IP: $EXTBROAD >> /tmp/rc.firewall.dump
#echo External interface default gateway: $EXTGW >> /tmp/rc.firewall.dump
#echo ————————————————————————- >> /tmp/rc.firewall.dump
#echo External secondary DNS: $SECONDARYDNS >> /tmp/rc.firewall.dump
#echo External secured host: $SECUREHOST >> /tmp/rc.firewall.dump

#————————————————————————————
# General
#————————————————————————————
# Performs general processing such as setting the multicast route
# and DHCP address hacking.
#
# Multicast is a powerful, yet seldom used aspect of TCP/IP for multimedia
# data. Though it isn't used much now (because most ISPs don't enable multicast
# on their networks, it will be very common in a few more years. Check out
# www.mbone.com for more detail.
#
# Adding this feature is OPTIONAL.
#
# Disabled by default.
#echo "  - Adding multicast route."
#/sbin/route add -net 224.0.0.0 netmask 240.0.0.0 dev $EXTIF
```

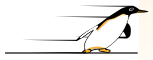


```
# Turn on IP Forwarding in the Linux kernel
#
# There are TWO methods of turning on this feature.  The first method is the
# Red Hat way. Edit the /etc/sysconfig/network file and change the
# "FORWARD_IPV4" line to say:
#
#         FORWARD_IPV4=true
#
# The second method is shown below and can executed at any time while the
# system is running.
#
echo "   - Enabling IP forwarding."
echo "1" > /proc/sys/net/ipv4/ip_forward

# Disable IP spoofing attacks.
#
# This drops traffic addressed for one network though it is being received on a
# different interface.
#
echo "   - Disabling IP Spoofing attacks."
for file in /proc/sys/net/ipv4/conf/*/rp_filter
do
 echo "1" > $file
done

# Comment the following out of you are not using a dynamic address
echo "   - Enabling dynamic TCP/IP address hacking."
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#-------------------------------------
# Masquerading Timeouts
#-------------------------------------
# Set timeout values for masq sessions (seconds).
#
# Item #1 - 2 hrs timeout for TCP session timeouts
# Item #2 - 10 sec timeout for traffic after the TCP/IP "FIN" packet is received
# Item #3 - 60 sec timeout for UDP traffic
#
echo "   - Changing IP masquerading timeouts."
/sbin/ipchains -M -S 7200 10 60
```



APPENDIX D
A strong packet firewall ruleset

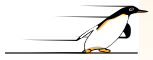
```
#--------------------------------------
# Masq Modules
#--------------------------------------
# Most TCP/IP-enabled applications work fine behind a Linux IP
# Masquerade server.  But, some applications need a special
# module to get their traffic in and out properly.
#
# Note:   Some applications do NOT work though IP Masquerade server at ALL such
#         as any H.323-based program.  Please the IP-MASQ HOWTO for more details.
#
# Note #2:Only uncomment the modules that you REQUIRE to be loaded.
#         The FTP module is loaded by default.
#--------------------------------------
echo "  - Loading masquerading modules."

#/sbin/modprobe ip_masq_cuseeme
/sbin/modprobe ip_masq_ftp
#/sbin/modprobe ip_masq_irc
#/sbin/modprobe ip_masq_quake
#/sbin/modprobe ip_masq_raudio
#/sbin/modprobe ip_masq_vdolive

#--------------------------------------
# Default Policies
#--------------------------------------
# Set all default policies to REJECT and flush all old rules.
#--------------------------------------
echo "  - Flushing all old rules and setting all default policies to REJECT "

# Flush all old rulesets
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward

# Change default policies to REJECT.
#
# We want to only EXPLICTLY allow what traffic is allowed IN and OUT of the
# firewall.  All other traffic will be implicitly blocked.
/sbin/ipchains -P input REJECT
/sbin/ipchains -P output REJECT
/sbin/ipchains -P forward REJECT
```



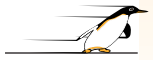
```
#*********************************************************************
# Input Rules
#*********************************************************************
echo "————————————————————————————————————"
echo "Input Rules:"

#————————————————————————————————
# Incoming Traffic on the Internal LAN
#————————————————————————————————
# This section controls the INPUT traffic allowed to flow within the internal
# LAN.  This means that all input traffic on the local network is valid.  If
# you want to change this default setting and only allow certain types of
# traffic within your internal network, you will need to comment this following
# line and configure individual ACCEPT lines for each TCP/IP address you want
# to let through.  A few example ACCEPT lines are provided below for
# demonstration purposes.
#
# Sometimes it is useful to allow TCP connections in one direction but not the
# other.  For example, you might want to allow connections to an external HTTP
# server but not connections from that server.  The naive approach would be to
# block TCP packets coming from the server. However, the better approach is to
# use the -y flag which will block only the packets used to request a
# connection.
#————————————————————————————————
echo "  - Setting input filters for traffic on the internal LAN."

# Local interface, local machines, going anywhere is valid.
#
# Comment this line out if you want to only allow specific traffic on the
# internal network.
/sbin/ipchains -A input -j ACCEPT -i $INTIF -s $INTLAN -d $UNIVERSE

# Loopback interface is valid.
/sbin/ipchains -A input -j ACCEPT -i $LOOPBACKIF -s $UNIVERSE -d $UNIVERSE

# DHCP Server.
#
# If you have configured a DHCP server on the Linux machine to serve IP
# addresses to the internal network, you will need to enable this section.
#
# This is an example of how to let input traffic flow through the local
# LAN if we have rejected all prior requests above.
#
# Disabled by default
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p udp -s $UNIVERSE bootpc -d $BROADCAST/0 bootps
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $UNIVERSE bootpc -d $BROADCAST/0 bootps
```

```
#--------------------------------------
# Explicit Access from Internal LAN Hosts
#--------------------------------------
# This section is provided as an example of how to allow only SPECIFIC hosts on
# the internal LAN to access services on the firewall server.  Many people
# might feel that this is extreme but many system attacks occur from the
# INTERNAL networks.
#
# Examples given allow access via FTP, FTP-DATA, SSH, and TELNET.
#
# In order for this ruleset to work, you must first comment out the line above
# that provides full access to the internal LAN by all internal hosts. You will
# then need to enable the lines below to allow any access at all.
#--------------------------------------
#echo "  - Setting input filters for specific internal hosts."

# First allowed internal host to connect directly to the Linux server
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp-data
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ssh
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP telnet

# Second allowed internal host to connect directly to the Linux server
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp-data
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ssh
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP telnet

#--------------------------------------
# Incoming Traffic from the External Interface
#--------------------------------------
# This ruleset will control specific traffic that is allowed in from
# the external interface.
#--------------------------------------
#
echo "  - Setting input filters for traffic from the external interface."
```

```
# Remote interface, claiming to be local machines, IP spoofing, get lost & log
/sbin/ipchains -A input -j REJECT -i $EXTIF -s $INTLAN -d $UNIVERSE $LOGGING

# DHCP Clients.
#
# If you get a dynamic IP address for your ADSL or Cablemodem connection, you
# will need to enable these lines.
#
# Enabled by default.
/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p udp -s $UNIVERSE bootps -d $BROADCAST/0  bootpc
/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE bootps -d $BROADCAST/0  bootpc

# FTP:  Allow external users to connect to the Linux server ITSELF for
#       PORT-style FTP services.  This will NOT work for PASV FTP transfers.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ftp-data

# HTTP: Allow external users to connect to the Linux server ITSELF for
#       HTTP services.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP http

# ICMP: Allow ICMP packets from all external TCP/IP addresses.
#
# NOTE: Disabling ICMP packets via the firewall ruleset can do far more than
# just stop people from pinging your machine.  Many aspects of TCP/IP and its
# associated applications rely on various ICMP messages.  Without ICMP, both
# your Linux server and internal Masq'ed computers might not work.
#
/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p icmp -s $UNIVERSE -d $EXTIP

# NFS: Reject NFS traffic FROM and TO external machines.
#
# NOTE: NFS is one of the biggest security issues an administrator will face.
# Do NOT enable NFS over the Internet or any non-trusted networks unless you
# know exactly what you are doing.
#
/sbin/ipchains -A input -j REJECT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP 2049
/sbin/ipchains -A input -j REJECT -i $EXTIF -p tcp -s $UNIVERSE 2049 -d $EXTIP
```



```
# NNTP: Allow external computers to connect to the Linux server ITSELF
#       for NNTP (news) services.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP nntp

# NTP:  Allow external computers to connect to the Linux server ITSELF for
#       NTP (time) updates
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ntp

# TELNET: Allow external computers to connect to the Linux server ITSELF for
#         TELNET access.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP telnet

# SSH server:  Allow external computers to connect to the Linux server ITSELF
#              for SSH access.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ssh

#--------------------------------------
# Incoming Traffic on all Interfaces
#--------------------------------------
# This will control input traffic for all interfaces.  This is
# usually used for what could be considered as public services.
#--------------------------------------
echo "  - Setting input filters for public services (all interfaces)."

# AUTH: Allow the authentication protocol, ident, to function on all
#       interfaces but disable it in /etc/inetd.conf.  The reason to
#       allow this traffic in but block it via Inetd is because some
#       legacy TCP/IP stacks don't deal with REJECTed "auth" requests
#       properly.
#
/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $UNIVERSE auth

# BOOTP/DHCP: Reject all stray bootp traffic.
#
# Disabled by default.
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE bootpc
```



```
# DNS:  If you are running an authoritative DNS server, you must open
#       up the DNS ports on all interfaces to allow lookups.  If you are
#       running a caching DNS server, you will need to at least open the DNS
#       ports to internal interfaces.
#
#       It is recommend to secure DNS by restricting zone transfers and split
#       DNS servers as documented in Step 4.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $UNIVERSE domain
#/sbin/ipchains -A input -j ACCEPT -p udp -s $UNIVERSE -d $UNIVERSE domain

# RIP:  Reject all stray RIP traffic.  Many improperly configured
#       networks propagate network routing protocols to the edge of the
#       network.  The follow line will allow you explicitly filter it here
#       without logging to SYSLOG.
#
# Disabled by default.
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE route

# SAMBA:Reject all stray SAMBA traffic. Many networks propagate the
#       chatty SMB network protocols to the edge of the network.  The
#       following line will allow you explicitly filter it here without
#       logging to SYSLOG.
#
# Disabled by default.
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE netbios-ns
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE netbios-dgm
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE netbios-ssn

# SMTP: If this server is an authoritative SMTP email server, you must
#       allow SMTP traffic to all interfaces.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $EXTIP smtp

#--------------------------------------
# Explicit INPUT Access from external LAN Hosts
#--------------------------------------
# This controls external access from specific external hosts (secure hosts).
# This example permits FTP, FTP-DATA, SSH, POP-3 and TELNET traffic from a
# secure host INTO the firewall. In addition to these input rules, we must also
# explicitly allow the traffic from the remote host to get out.  See the rules
# in the output section for more details
#
# Disabled as default.
#--------------------------------------
echo "  - Setting input filters for explicit external hosts."
```

```
# The secure host
#
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP ftp-data
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP ssh
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP pop-3
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP telnet
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST2 -d $EXTIP telnet
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST2 -d $EXTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST2 -d $EXTIP ftp-data

#--------------------------------------
# Port Forwarding
#--------------------------------------
# Port forwarding allows external traffic to directly connect to an INTERNAL
# Masq'ed machine. An example need for port forwarding is the need for external
# users to directly contact a WWW server behind the MASQ server.
#
# NOTE:  Port forwarding is well beyond the scope of this documentation to
#        explain the security issues implied in opening up access like this.
#        Please see Appendix A to read the IP-MASQ-HOWTO for a full explanation.
#
# Do not use ports greater than 1023 for redirection ports.
#
# Disabled by default.
#--------------------------------------
#echo "   * Enabling Port Forwarding onto internal hosts."
#/usr/sbin/ipmasqadm portfw -f
#echo "   * Forwarding SSH traffic on port 26 to $PORTFWIP"
#/usr/sbin/ipmasqadm portfw -a -P tcp -L $EXTIP 26 -R $PORTFWIP 22

# HIGH PORTS:
#
# Enable all high unprivileged ports for all reply TCP/UDP traffic
#
# NOTE:  The use of the "! -y" flag filters TCP traffic that doesn't have the
#        SYN bit set.  In other words, this means that any traffic that is
#        trying to initiate traffic to your server on a HIGH port will be
#        rejected.
#
#        The only HIGH port traffic that will be accepted is either return
#        traffic that the server originally initiated or UDP-based traffic.
#
# NOTE2: Please note that port 20 for ACTIVE FTP sessions should NOT use
#        SYN filtering.  Because of this, we must specifically allow it in.
#
echo "   - Enabling all input REPLY (TCP/UDP) traffic on high ports."
```

```
/sbin/ipchains -A input -j ACCEPT ! -y -p tcp -s $UNIVERSE -d $EXTIP $UNPRIVPORTS
/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE ftp-data -d $EXTIP $UNPRIVPORTS
/sbin/ipchains -A input -j ACCEPT -p udp -s $UNIVERSE -d $EXTIP $UNPRIVPORTS

#————————————————————————
# Catch All INPUT Rule
#————————————————————————
#
echo "  - Final input catch all rule."

# All other incoming is denied and logged.
/sbin/ipchains -A input -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING


#*********************************************************************
# Output Rules
#*********************************************************************
echo "————————————————————————————————"
echo "Output Rules:"

#————————————————————————
# Outgoing Traffic on the Internal LAN
#————————————————————————
# This ruleset provides policies for traffic that is going out on the internal
# LAN.
#
# In this example, all traffic is allowed out.  Therefore there is no
# requirement to implement individual filters.  However, as with the input
# section above, examples are given for demonstrative purposes.  It is also
# noted that the same rules, outlined above, apply regarding the order of the
# filtering rules.
#————————————————————————
echo "  - Setting output filters for traffic on the internal LAN."

# Local interface, any source going to local net is valid.
/sbin/ipchains -A output -j ACCEPT -i $INTIF -s $UNIVERSE -d $INTLAN

# Loopback interface is valid.
/sbin/ipchains -A output -j ACCEPT -i $LOOPBACKIF -s $UNIVERSE -d $UNIVERSE
```

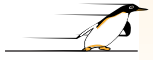
```
# DHCP: If you have configured a DHCP server on this Linux machine, you
#       will need to enable the following ruleset.
#
# Disabled by default.
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p udp -s $INTIP/32 bootps -d $BROADCAST/0 bootpc
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTIP/32 bootps -d $BROADCAST/0 bootpc

# HTTP: The following is an example of how to allow HTTP traffic to an
#       intranet WWW server without allowing access from the external
#       network.
#
# Disabled by default.
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTIP/32 http -d $INTLAN


#--------------------------------------
# Explicit Output from Internal LAN Hosts
#--------------------------------------
# The following rulesets only allow SPECIFIC hosts on the internal LAN to
# access services on this firewall server itself.  Many people might feel that
# this is extreme but many system attacks occur from the INTERNAL network as
# well.
#
# Examples given allow access via FTP, FTP-DATA, SSH, and TELNET.
#
# In order for this ruleset to work, you must first comment out the line above
# that provides full access to the internal LAN by all internal hosts.
#
# Disabled by default.
#--------------------------------------
#echo "  - Setting output filters for specific internal hosts."

# First host
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp-data
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ssh
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP telnet

# Second host
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp-data
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ssh
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP telnet
```

APPENDIX D
A strong packet firewall ruleset

```
#------------------------------------
# Outgoing Traffic on the External Interface
#------------------------------------
# This ruleset will control what traffic can go out on the external interface.
#------------------------------------
echo "  - Setting input filters for traffic to the external interface."

# Reject outgoing traffic to the local net from the remote interface,
# stuffed routing; deny & log
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $UNIVERSE -d $INTLAN $LOGGING

# Reject outgoing traffic from the local net from the external interface,
# stuffed masquerading, deny and log
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $INTLAN -d $UNIVERSE $LOGGING

# DHCP Client: If your Linux server is connected via DSL or a Cablemodem
#              connection and you get dynamic DHCP addresses, you will need to
#              enable the following rulesets.
#
# Enabled by default.
/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE bootpc -d $UNIVERSE bootps
/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p udp -s $UNIVERSE bootpc -d $UNIVERSE bootps

# FTP: Allow FTP traffic (the Linux server is a FTP server)
#
# Disabled by default.
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp -d $UNIVERSE
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp-data -d $UNIVERSE

# HTTP: Allow HTTP traffic (the Linux server is a WWW server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP http -d $UNIVERSE

# NTP: Allow NTP updates (the Linux server is a NTP server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ntp -d $UNIVERSE

# TELNET: Allow telnet traffic (the Linux server is a TELNET server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP telnet -d $UNIVERSE

# SSH server: Allow outgoing SSH traffic (the Linux server is a SSH server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ssh -d $UNIVERSE
```

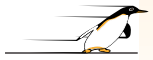
APPENDIX D
A strong packet firewall ruleset

```
#--------------------------------------
# Outgoing Traffic on all Interfaces
#--------------------------------------
# This will control output traffic for all interfaces.  This is
# usually used for what could be considered as public services.  It
# is noted that we provide a few rejection rulesets as examples but
# these are not required due to the overall REJECT statement above.
#--------------------------------------
echo "  - Setting output filters for public services on all interfaces."

# AUTH: Allow authentication tap indent on all interfaces (but disable it
#       in /etc/inetd.conf).
#
/sbin/ipchains -A output -j ACCEPT -p tcp -s $UNIVERSE auth -d $UNIVERSE

# DNS: If you your Linux server is an authoritative DNS server, you must
# enable this ruleset
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP domain -d $UNIVERSE
#/sbin/ipchains -A output -j ACCEPT -p udp -s $EXTIP domain -d $UNIVERSE

# ICMP: Allow ICMP traffic out
#
# NOTE: Disabling ICMP packets via the firewall ruleset can do far
# more than just stop people from pinging your machine.  Many aspects
# of TCP/IP and its associated applications rely on various ICMP
# messages.  Without ICMP, both your Linux server and internal Masq'ed
# computers might not work.
#
/sbin/ipchains -A output -j ACCEPT -p icmp -s $UNIVERSE -d $UNIVERSE

# NNTP: This allows NNTP-based news out.
#
/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP nntp -d $UNIVERSE

# SMTP: If the Linux servers is either an authoritative SMTP server or
# relay, you must allow this ruleset.
#
/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP smtp -d $UNIVERSE
```



```
#--------------------------------------
# Specific Output Rejections
#--------------------------------------
# These rulesets reject specific traffic that you do not want out of
# the system.
#--------------------------------------
echo "  - Reject specific outputs."

# RPC.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE sunrpc $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP sunrpc -d $UNIVERSE $LOGGING

# Mountd.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 635 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP 635 -d $UNIVERSE $LOGGING

# PPTP.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 1723 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 1723 $LOGGING

# Remote Winsock.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 1745 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 1745 $LOGGING

# NFS.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 2049 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP 2049 -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 2049 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP 2049 -d $UNIVERSE $LOGGING

# PcAnywhere.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 5631 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 5631 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 5632 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 5632 $LOGGING

# Xwindows.
#
# NOTE: See variable section above for the example range (6000:6010 by default)
# Xwindows can use far more than just ports 6000-6010.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE $XWINDOWS_PORTS $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE $XWINDOWS_PORTS $LOGGING
```

```
# NetBus.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 12345 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 12346 $LOGGING

# NetBus Pro.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 20034 $LOGGING

# BackOrofice
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE/0 31337 $LOGGING

# Win Crash Trojan.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 5742 $LOGGING


# Socket De Troye.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 30303 $LOGGING

# Unknown Trojan Horse (Master's Paradise [CHR])
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 40421 $LOGGING



#------------------------------------
# Output to Explicit Hosts
#------------------------------------
# This controls output to specific external hosts (secure hosts).  This example
# implementation allows ssh and pop-3 protocols out to the secure host.  In
# addition to these rules, we must also explicitly allow the traffic in from
# the remote host.  See the input rules above to see this take place.
#
# Disabled by default.
#------------------------------------
echo "  - Setting output filters for explicit external hosts."

# The secure host
#
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp -d $SECUREHOST $UNPRIV-
PORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp-data -d $SECUREHOST $UNPRIVPORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ssh -d $SECUREHOST $UNPRIVPORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP pop-3 -d $SECUREHOST $UNPRIVPORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP telnet -d $SECUREHOST $UNPRIVPORT
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP telnet -d $SECUREHOST2 $UNPRIVPORT
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp -d $SECUREHOST2 $UNPRIVPORT
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp-data -d $SECUREHOST2 $UNPRIVPORT
```



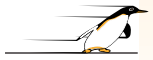
```
# Allow all High Ports for return traffic.
#
echo "  - Enabling all output REPLY (TCP/UDP) traffic on high ports."
/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP $UNPRIVPORTS -d $UNIVERSE
/sbin/ipchains -A output -j ACCEPT -p udp -s $EXTIP $UNPRIVPORTS -d $UNIVERSE


#--------------------------------------
# Catch All Rule
#--------------------------------------
echo "  - Final output catch all rule."

# All other outgoing is denied and logged.  This ruleset should catch
# everything including samba that hasn't already been blocked.#
/sbin/ipchains -A output -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING


#*********************************************************************
# Forwarding Rules
#*********************************************************************
#
echo "--------------------------------------"
echo "Forwarding Rules:"


#--------------------------------------
# Enable TCP/IP forwarding and masquerading from the Internal LAN
#--------------------------------------

# Masquerade from local net on local interface to anywhere.
#
echo "  - Enable IP Masquerading from the internal LAN."
/sbin/ipchains -A forward -j MASQ -i $EXTIF -s $INTLAN -d $UNIVERSE

# Catch all rule, all other forwarding is denied.
#

/sbin/ipchains -A forward -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING

#*********************************************************************
# The end
#*********************************************************************
echo "--------------------------------------"
echo -e "Firewall implemented. \n\n"
```

APPENDIX E
Script to modify default permissions of system binaries

The default file permissions on most Linux distributions give away too much information to the user, and/or allow ordinary users to perform actions they have no business doing. The settings below may be paranoid but it's better to be safe than sorry.

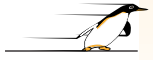
NOTE: most of these permissions will work for a common installation but some files won't exist depending on what packages were and weren't installed. Run this script everytime you reinstall Linux, or install updated packages.

```
#!/bin/sh
#————————————————————————————
# Author: David A. Ranch
# Based on the TrinityOS file permissions corrections
#————————————————————————————

MODE="o-rwx"

# Files in /bin
cd /bin
chmod $MODE linuxconf mount mt setserial umount

# Files in /sbin
cd /sbin
chmod $MODE badblocks ctrlaltdel chkconfig debugfs depmod dump*
chmod $MODE fdisk fsck* ftl* halt hdparm hwclock if* init insmod isapnp
chmod $MODE kerneld killall* lilo mgetty mingetty mk* mod* netreport
chmod $MODE pam* pcinitrd pnpdump portmap quotaon restore runlevel
chmod $MODE stinit swapon tune2fs uugetty

# Files in /usr/bin
cd /usr/bin
chmod $MODE control-panel comanche eject gnome* gpasswd kernelcfg

chmod 755 lp*
chmod 4755 lpr

#NOTE: I feel setting "lpr" to allow any group to execute it is
#      a bad thing.
#
#      I would like to add UNIX users and even the Samba process to
#      the "lp" group already defined in /etc/groups and then be able
#      to put  things back to 4750.  BUT.. this really isn't possible.
#      Linux doesn't support multiple groups per file and Linux
#      doesn't support access lists (ACLs') yet.  So.. you either have
#      either leave these files SUID or run LPRng.

chmod $MODE minicom netcfg
```



APPENDIX E
Script to modify default permissions of system binaries

```
# Files in /usr/sbin
cd /usr/sbin
chmod $MODE at* crond dhc* edquota exportfs ftpshut group* grp*
chmod $MODE imapd in.* inetd ipop* klogd logrotate lp*
chmod 755 lsof
chmod $MODE makemap mouseconfig named* nmbd newusers ntp* ntsysv
chmod $MODE pppd pw* quota* rdev repquota rotatelogs rpc* samba
chmod $MODE setup showmount smb* squid syslogd taper tcpd* time*
chmod $MODE tmpwatch tunelp user* vi* xntp*
```



ADDITIONAL RESOURCES FROM THE SANS INSTITUTE

The SANS Institute is a cooperative research and education organization through which system administrators, security professionals, and network administrators share the lessons they are learning.

It offers educational conferences and in-depth courses, cooperative research reports, and electronic digests of authoritative answers to current questions.

TO ORDER:

http://www.sansstore.org/

ELECTRONIC DIGESTS

SANS NewsBites

A summary, distributed via email, of the dozen most important news articles that have been published on computer security during the past week. Each entry includes a brief highlight of the article and a url to allow you to read the whole story if it is still posted. To Subscribe, send email to info@ sans.org with the subject "SANS NewsBites."

The SANS Network Security Digest

Published every month, and distributed via email, the SANS Network Security Digest reports on the most important new security threats and provides guidance on where to fine the latest patches or additional information on the threats. Each issue can be read in about eight minutes. The SANS Digest is written by Michele Guel with assistance from Matt Bishop, Gene Spafford, Steve Bellovin, Bill Cheswick, Gene Schultz, Marcus Ranum, Rob Kolstad, Hal Pomeranz, Dorothy Denning, and several other leading experts. Subscribe to this digest by sending a message with the subject "subscribe" to digest@sans.org.

The NT Digest

The digest provides updates to NT Security: Step-by-Step plus up-to-date guidance on new Hotfixes and Service Packs that should and should not be implemented. It also summarizes new threats and new bugs found in NT and its services. Subscribe to this digest by sending a message with the subject "subscribe nt digest" to digest@sans.org.

ROADMAP TO NETWORK SECURITY POSTER

The SANS Roadmap To Network Security Wall Poster
Updated twice a year, these posters present "top ten" lists of answers to common questions: the best security books, the best security web sites, the biggest threats, the vendor contacts, the top tools and more. They are mailed automatically to all Network Security Digest subscribers and people who attend the Institute conferences.

COOPERATIVE RESEARCH REPORTS AND PROJECTS

Intrusion Detection Shadow Style: A Step-by-Step Guide

A booklet describing how to perform advanced intrusion detection. It uses the open source, public domain Shadow software developed at the Naval Surface Warfare Center and used to protect more than 15,000 hosts through high-speed intrusion detection and analysis.

The SANS Salary Survey

Published annually, the survey reports salaries of sysadmin, networking, and security professionals based on their primary operating environment (UNIX, NT, Netware, or combination) where they live, the type and size of employer, the machines they manage, whether they are employees or consultants, and other characteristics. It also reports the size of their raises, by salary level, and the principal reasons report for above-average raises. More than 15,000 people participated in the 1999 survey.

Windows NT Security: Step-by-Step

A consensus of security professionals from seventy-seven large user organizations who worked together to develop a list of ninety-three actions in eight phases that should be done to secure and NT server. 36 pages.

Computer Security Incident Handling: Step-by-Step

A consensus of the leading incident handling agencies and experts plus fifty other experienced incident handling professionals. 44 pages.

Windows NT PowerTools: Administrator's Consensus

This is a consensus report which two-hundred and twenty NT administrators shared their experiences in implementing and using twenty of the most popular tools for improving efficiency and security on Windows NT systems.

Solaris Security: Step-by -Step

An extremely valuable collection of the expert knowledge on how to make a Solaris system ready for the internet.

Linux Security: Step-by-Step

Red Hat and other Linux variants have great technology. To use them in the world of business they also need great security. This booklet provides the roadmap.