Security−HOWTO

Table of Contents

Table of Contents

Table of Contents

Linux Security HOWTO

Kevin Fenzi, `kevin@tummy.com` & Dave Wreski, `dave@linuxsecurity.com`

v1.1.1, 17 March 2000

This document is a general overview of security issues that face the administrator of Linux systems. It covers general security philosophy and a number of specific examples of how to better secure your Linux system from intruders. Also included are pointers to security−related material and programs. Improvements, constructive criticism, additions and corrections are gratefully accepted. Please mail your feedback to both authors, with "Security HOWTO" in the subject.

1.Introduction

2.Overview

3.Physical Security

4.Local Security

5.Files and Filesystem Security

6.Password Security and Encryption

7.Kernel Security

8.Network Security

15.Acknowledgements

1.Introduction

This document covers some of the main issues that affect Linux security. General philosophy and net−born resources are discussed.

A number of other HOWTO documents overlap with security issues, and those documents have been pointed to wherever appropriate.

This document is *not* meant to be a up−to−date exploits document. Large numbers of new exploits happen all the time. This document will tell you where to look for such up−to−date information, and will give some general methods to prevent such exploits from taking place.

1.1 New Versions of this Document

New versions of this document will be periodically posted to *comp.os.linux.answers*. They will also be added to the various sites that archive such information, including:

`http://www.linuxdoc.org/`

In addition, you should generally be able to find this document on the Linux World Wide Web home page via:

`http://metalab.unc.edu/mdw/linux.html`

Finally, the very latest version of this document should also be available in various formats from:

`http://scrye.com/~kevin/lsh/`

or

http://www.linuxsecurity.com/Security−HOWTO

or

http://www.tummy.com/security−howto

1.2 Feedback

All comments, error reports, additional information and criticism of all sorts should be directed to:

`kevin@tummy.com`

and

`dave@linuxsecurity.com`

Note: Please send your feedback to *both* authors. Also, be sure and include "Linux" "security", or "HOWTO" in your subject to avoid Kevin's spam filter.

1.3 Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and other content at your own risk. Additionally, this is an early version, possibly with many inaccuracies or errors.

A number of the examples and descriptions use the RedHat(tm) package layout and system setup. Your mileage may vary.

As far as we know, only programs that, under certain terms may be used or evaluated for personal purposes will be described. Most of the programs will be available, complete with source, under GNU terms.

1.4 Copyright Information

2.Overview

This document will attempt to explain some procedures and commonly−used software to help your Linux system be more secure. It is important to discuss some of the basic concepts first, and create a security foundation, before we get started.

2.1 Why Do We Need Security?

In the ever−changing world of global data communications, inexpensive Internet connections, and fast−paced software development, security is becoming more and more of an issue. Security is now a basic requirement because global computing is inherently insecure. As your data goes from point A to point B on the Internet, for example, it may pass through several other points along the way, giving other users the opportunity to intercept, and even alter, it. Even other users on your system may maliciously transform your data into something you did not intend. Unauthorized access to your system may be obtained by intruders, also known as "crackers", who then use advanced knowledge to impersonate you, steal information from you, or even deny you access to your own resources. If you're wondering what the difference is between a "Hacker" and a "Cracker", see Eric Raymond's document, "How to Become A Hacker", available at http://www.netaxs.com/~esr/faqs/hacker−howto.html.

2.2 How Secure Is Secure?

First, keep in mind that no computer system can ever be completely secure. All you can do is make it increasingly difficult for someone to compromise your system. For the average home Linux user, not much is required to keep the casual cracker at bay. However, for high−profile Linux users (banks, telecommunications companies, etc), much more work is required.

Another factor to take into account is that the more secure your system is, the more intrusive your security becomes. You need to decide where in this balancing act your system will still usable, and yet secure for your purposes. For instance, you could require everyone dialing into your system to use a call−back modem to call them back at their home number. This is more secure, but if someone is not at home, it makes it difficult for them to login. You could also setup your Linux system with no network or connection to the Internet, but this limits its usefulness.

If you are a medium to large−sized site, you should establish a security policy stating how much security is required by your site and what auditing is in place to check it. You can find a well−known security policy example at http://www.faqs.org/rfcs/rfc2196.html. It has been recently updated, and contains a great framework for establishing a security policy for your company.

2.3 What Are You Trying to Protect?

Before you attempt to secure your system, you should determine what level of threat you have to protect against, what risks you should or should not take, and how vulnerable your system is as a result. You should analyze your system to know what you're protecting, why you're protecting it, what value it has, and who has responsibility for your data and other assets.

* *Risk* is the possibility that an intruder may be successful in attempting to access your computer. Can an intruder read or write files, or execute programs that could cause damage? Can they delete critical data? Can they prevent you or your company from getting important work done? Don't forget: someone gaining access to your account, or your system, can also impersonate you.

  Additionally, having one insecure account on your system can result in your entire network being compromised. If you allow a single user to login using a `.rhosts` file, or to use an insecure service such as `tftp`, you risk an intruder getting 'his foot in the door'. Once the intruder has a user account on your system, or someone else's system, it can be used to gain access to another system, or another account.
* *Threat* is typically from someone with motivation to gain unauthorized access to your network or computer. You must decide whom you trust to have access to your system, and what threat they could pose.

  There are several types of intruders, and it is useful to keep their different characteristics in mind as you are securing your systems.

  - **The Curious** − This type of intruder is basically interested in finding out what type of system and data you have.
  - **The Malicious** − This type of intruder is out to either bring down your systems, or deface your web page, or otherwise force you to spend time and money recovering from the damage he has caused.
  - **The High−Profile Intruder** − This type of intruder is trying to use your system to gain popularity and infamy. He might use your high−profile system to advertise his abilities.
  - **The Competition** − This type of intruder is interested in what data you have on your system. It might be someone who thinks you have something that could benefit him, financially or otherwise.
  - **The Borrowers** − This type of intruder is interested in setting up shop on your system and using its resources for their own purposes. He typically will run chat or irc servers, porn archive sites, or even DNS servers.
  - **The Leapfrogger** − This type of intruder is only interested in your system to use it to get into other systems. If your system is well−connected or a gateway to a number of internal hosts, you may well see this type trying to compromise your system.
* Vulnerability describes how well−protected your computer is from another network, and the potential for someone to gain unauthorized access.

  What's at stake if someone breaks into your system? Of course the concerns of a dynamic PPP home user will be different from those of a company connecting their machine to the Internet, or another large network.

  How much time would it take to retrieve/recreate any data that was lost? An initial time investment now can save ten times more time later if you have to recreate data that was lost. Have you checked your backup strategy, and verified your data lately?

2.4 Developing A Security Policy

Create a simple, generic policy for your system that your users can readily understand and follow. It should protect the data you're safeguarding as well as the privacy of the users. Some things to consider adding are: who has access to the system (Can my friend use my account?), who's allowed to install software on the system, who owns what data, disaster recovery, and appropriate use of the system.

A generally−accepted security policy starts with the phrase

> **That which is not permitted is prohibited**

This means that unless you grant access to a service for a user, that user shouldn't be using that service until you do grant access. Make sure the policies work on your regular user account. Saying, "Ah, I can't figure out this permissions problem, I'll just do it as root" can lead to security holes that are very obvious, and even ones that haven't been exploited yet.

rfc1244 is a document that describes how to create your own network security policy.

rfc1281 is a document that shows an example security policy with detailed descriptions of each step.

Finally, you might want to look at the COAST policy archive at ftp://coast.cs.purdue.edu/pub/doc/policy to see what some real−life security policies look like.

2.5 Means of Securing Your Site

This document will discuss various means with which you can secure the assets you have worked hard for: your local machine, your data, your users, your network, even your reputation. What would happen to your reputation if an intruder deleted some of your users' data? Or defaced your web site? Or published your company's corporate project plan for next quarter? If you are planning a network installation, there are many factors you must take into account before adding a single machine to your network.

Even if you have a single dialup PPP account, or just a small site, this does not mean intruders won't be interested in your systems. Large, high−profile sites are not the only targets −− many intruders simply want to exploit as many sites as possible, regardless of their size. Additionally, they may use a security hole in your site to gain access to other sites you're connected to.

Intruders have a lot of time on their hands, and can avoid guessing how you've obscured your system just by trying all the possibilities. There are also a number of reasons an intruder may be interested in your systems, which we will discuss later.

Host Security

Perhaps the area of security on which administrators concentrate most is host−based security. This typically involves making sure your own system is secure, and hoping everyone else on your network does the same. Choosing good passwords, securing your host's local network services, keeping good accounting records, and upgrading programs with known security exploits are among the things the local security administrator is responsible for doing. Although this is absolutely necessary, it can become a daunting task once your network becomes larger than a few machines.

Local Network Security

Network security is as necessary as local host security. With hundreds, thousands, or more computers on the same network, you can't rely on each one of those systems being secure. Ensuring that only authorized users can use your network, building firewalls, using strong encryption, and ensuring there are no "rogue" (that is, unsecured) machines on your network are all part of the network security administrator's duties.

This document will discuss some of the techniques used to secure your site, and hopefully show you some of the ways to prevent an intruder from gaining access to what you are trying to protect.

Security Through Obscurity

One type of security that must be discussed is "security through obscurity". This means, for example, moving a service that has known security vunerabilities to a non−standard port in hopes that attackers won't notice it's there and thus won't exploit it. Rest assured that they can determine that it's there and will exploit it. Security through obscurity is no security at all. Simply because you may have a small site, or a relatively low profile, does not mean an intruder won't be interested in what you have. We'll discuss what you're protecting in the next sections.

2.6 Organization of This Document

This document has been divided into a number of sections. They cover several broad security issues. The first, Physical Security, covers how you need to protect your physical machine from tampering. The second, Local Security, describes how to protect your system from tampering by local users. The third, Files and Filesystem Security, shows you how to setup your filesystems and permissions on your files. The next, Password Security and Encryption, discusses how to use encryption to better secure your machine and network. Kernel Security discusses what kernel options you should set or be aware of for a more secure system. Network Security, describes how to better secure your Linux system from network attacks. Security Preparation, discusses how to prepare your machine(s) before bringing them on−line. Next, What To Do During and After a Break−in, discusses what to do when you detect a system compromise in progress or detect one that has recently happened. In Security Resources, some primary security resources are enumerated. The Q and A section Frequently Asked Questions, answers some frequently−asked questions, and finally a conclusion in Conclusion.

The two main points to realize when reading this document are:

- Be aware of your system. Check system logs such as `/var/log/messages` and keep an eye on your system, and
- Keep your system up−to−date by making sure you have installed the current versions of software and have upgraded per security alerts. Just doing this will help make your system markedly more secure.

3.Physical Security

The first layer of security you need to take into account is the physical security of your computer systems. Who has direct physical access to your machine? Should they? Can you protect your machine from their tampering? Should you?

How much physical security you need on your system is very dependent on your situation, and/or budget.

If you are a home user, you probably don't need a lot (although you might need to protect your machine from tampering by children or annoying relatives). If you are in a lab, you need considerably more, but users will still need to be able to get work done on the machines. Many of the following sections will help out. If you are in an office, you may or may not need to secure your machine off−hours or while you are away. At some companies, leaving your console unsecured is a termination offense.

Obvious physical security methods such as locks on doors, cables, locked cabinets, and video surveillance are all good ideas, but beyond the scope of this document. :)

3.1 Computer locks

Many modern PC cases include a "locking" feature. Usually this will be a socket on the front of the case that allows you to turn an included key to a locked or unlocked position. Case locks can help prevent someone from stealing your PC, or opening up the case and directly manipulating/stealing your hardware. They can also sometimes prevent someone from rebooting your computer from their own floppy or other hardware.

These case locks do different things according to the support in the motherboard and how the case is constructed. On many PC's they make it so you have to break the case to get the case open. On some others, they will not let you plug in new keyboards or mice. Check your motherboard or case instructions for more information. This can sometimes be a very useful feature, even though the locks are usually very low−quality and can easily be defeated by attackers with locksmithing.

Some machines (most notably SPARCs and macs) have a dongle on the back that, if you put a cable through, attackers would have to cut the cable or break the case to get into it. Just putting a padlock or combo lock through these can be a good deterrent to someone stealing your machine.

3.2 BIOS Security

The BIOS is the lowest level of software that configures or manipulates your x86−based hardware. LILO and other Linux boot methods access the BIOS to determine how to boot up your Linux machine. Other hardware that Linux runs on has similar software (OpenFirmware on Macs and new Suns, Sun boot PROM, etc...). You can use your BIOS to prevent attackers from rebooting your machine and manipulating your Linux system.

Many PC BIOSs let you set a boot password. This doesn't provide all that much security (the BIOS can be reset, or removed if someone can get into the case), but might be a good deterrent (i.e. it will take time and leave traces of tampering). Similarly, on S/Linux (Linux for SPARC(tm) processor machines), your EEPROM can be set to require a boot−up password. This might slow attackers down.

Many x86 BIOSs also allow you to specify various other good security settings. Check your BIOS manual or look at it the next time you boot up. For example, some BIOSs disallow booting from floppy drives and some require passwords to access some BIOS features.

Note: If you have a server machine, and you set up a boot password, your machine will not boot up unattended. Keep in mind that you will need to come in and supply the password in the event of a power failure. ;(

3.3 Boot Loader Security

The various Linux boot loaders also can have a boot password set. LILO, for example, has `password` and `restricted` settings; `password` requires password at boot time, whereas `restricted` requires a boot−time password only if you specify options (such as `single`) at the `LILO` prompt.

From the lilo.conf man page:

```
password=password
              The per-image option `password=...' (see below) applies to all images.

restricted
              The per-image option `restricted' (see below) applies to all images.

       password=password
              Protect the image by a password.

       restricted
              A password is only required to boot the image if
              parameters are specified  on  the  command  line
              (e.g. single).
```

Keep in mind when setting all these passwords that you need to remember them. :) Also remember that these passwords will merely slow the determined attacker. They won't prevent someone from booting from a floppy, and mounting your root partition. If you are using security in conjunction with a boot loader, you might as well disable booting from a floppy in your computer's BIOS, and password−protect the BIOS.

If anyone has security−related information from a different boot loader, we would love to hear it. (`grub`, `silo`, `milo`, `linload`, etc).

Note: If you have a server machine, and you set up a boot password, your machine will *not* boot up unattended. Keep in mind that you will need to come in and supply the password in the event of a power failure. ;(

3.4 xlock and vlock

If you wander away from your machine from time to time, it is nice to be able to "lock" your console so that no one can tamper with, or look at, your work. Two programs that do this are: `xlock` and `vlock`.

`xlock` is a X display locker. It should be included in any Linux distributions that support X. Check out the man page for it for more options, but in general you can run `xlock` from any xterm on your console and it will lock the display and require your password to unlock.

`vlock` is a simple little program that allows you to lock some or all of the virtual consoles on your Linux box. You can lock just the one you are working in or all of them. If you just lock one, others can come in and use the console; they will just not be able to use your virtual console until you unlock it. `vlock` ships with redhat Linux, but your mileage may vary.

Of course locking your console will prevent someone from tampering with your work, but won't prevent them from rebooting your machine or otherwise disrupting your work. It also does not prevent them from accessing your machine from another machine on the network and causing problems.

More importantly, it does not prevent someone from switching out of the X Window System entirely, and going to a normal virtual console login prompt, or to the VC that X11 was started from, and suspending it, thus obtaining your privileges. For this reason, you might consider only using it while under control of xdm.

3.5 Detecting Physical Security Compromises

The first thing to always note is when your machine was rebooted. Since Linux is a robust and stable OS, the only times your machine should reboot is when *you* take it down for OS upgrades, hardware swapping, or the like. If your machine has rebooted without you doing it, that may be a sign that an intruder has compromised it. Many of the ways that your machine can be compromised require the intruder to reboot or power off your machine.

Check for signs of tampering on the case and computer area. Although many intruders clean traces of their presence out of logs, it's a good idea to check through them all and note any discrepancy.

It is also a good idea to store log data at a secure location, such as a dedicated log server within your well−protected network. Once a machine has been compromised, log data becomes of little use as it most likely has also been modified by the intruder.

The syslog daemon can be configured to automatically send log data to a central syslog server, but this is typically sent unencrypted, allowing an intruder to view data as it is being transferred. This may reveal information about your network that is not intended to be public. There are syslog daemons available that encrypt the data as it is being sent.

Also be aware that faking syslog messages is easy −− with an exploit program having been published. Syslog even accepts net log entries claiming to come from the local host without indicating their true origin.

Some things to check for in your logs:

- Short or incomplete logs.
- Logs containing strange timestamps.
- Logs with incorrect permissions or ownership.
- Records of reboots or restarting of services.
- missing logs.
- `su` entries or logins from strange places.

We will discuss system log data later in the HOWTO.

4.Local Security

The next thing to take a look at is the security in your system against attacks from local users. Did we just say *local* users? Yes!

Getting access to a local user account is one of the first things that system intruders attempt while on their way to exploiting the root account. With lax local security, they can then "upgrade" their normal user access to root access using a variety of bugs and poorly setup local services. If you make sure your local security is tight, then the intruder will have another hurdle to jump.

Local users can also cause a lot of havoc with your system even (especially) if they really are who they say they are. Providing accounts to people you don't know or for whom you have no contact information is a very bad idea.

4.1 Creating New Accounts

You should make sure you provide user accounts with only the minimal requirements for the task they need to do. If you provide your son (age 10) with an account, you might want him to only have access to a word processor or drawing program, but be unable to delete data that is not his.

Several good rules of thumb when allowing other people legitimate access to your Linux machine:

- Give them the minimal amount of privileges they need.
- Be aware when/where they login from, or should be logging in from.
- Make sure you remove inactive accounts
- The use of the same userid on all computers and networks is advisable to ease account maintence, and permits easier analysis of log data.
- The creation of group userid's should be absolutely prohibited. User accounts also provide accountability, and this is not possible with group accounts.

Many local user accounts that are used in security compromises have not been used in months or years. Since no one is using them they, provide the ideal attack vehicle.

4.2 Root Security

The most sought−after account on your machine is the root (superuser) account. This account has authority over the entire machine, which may also include authority over other machines on the network. Remember that you should only use the root account for very short, specific tasks, and should mostly run as a normal user. Even small mistakes made while logged in as the root user can cause problems. The less time you are on with root privileges, the safer you will be.

Several tricks to avoid messing up your own box as root:

- When doing some complex command, try running it first in a non−destructive way...especially commands that use globbing: e.g., if you want to do `rm foo*.bak`, first do `ls foo*.bak` and make sure you are going to delete the files you think you are. Using `echo` in place of destructive commands also sometimes works.
- Provide your users with a default alias to the `rm` command to ask for confirmation for deletion of files.
- Only become root to do single specific tasks. If you find yourself trying to figure out how to do something, go back to a normal user shell until you are *sure* what needs to be done by root.
- The command path for the root user is very important. The command path (that is, the `PATH` environment variable) specifies the directories in which the shell searches for programs. Try to limit the command path for the root user as much as possible, and *never* include `.` (which means "the current directory") in your PATH. Additionally, never have writable directories in your search path, as this can allow attackers to modify or place new binaries in your search path, allowing them to run as root the next time you run that command.
- Never use the rlogin/rsh/rexec suite of tools (called the r−utilities) as root. They are subject to many sorts of attacks, and are downright dangerous when run as root. Never create a `.rhosts` file for root.
- The `/etc/securetty` file contains a list of terminals that root can login from. By default (on Red Hat Linux) this is set to only the local virtual consoles(vtys). Be very wary of adding anything else to this file. You should be able to login remotely as your regular user account and then `su` if you need to (hopefully over `ssh` or other encrypted channel), so there is no need to be able to login directly as root.
- Always be slow and deliberate running as root. Your actions could affect a lot of things. Think before you type!

If you absolutely positively need to allow someone (hopefully very trusted) to have root access to your machine, there are a few tools that can help. `sudo` allows users to use their password to access a limited set of commands as root. This would allow you to, for instance, let a user be able to eject and mount removable media on your Linux box, but have no other root privileges. `sudo` also keeps a log of all successful and unsuccessful sudo attempts, allowing you to track down who used what command to do what. For this reason `sudo` works well even in places where a number of people have root access, because it helps you keep track of changes made.

Although `sudo` can be used to give specific users specific privileges for specific tasks, it does have several shortcomings. It should be used only for a limited set of tasks, like restarting a server, or adding new users. Any program that offers a shell escape will give root access to a user invoking it via `sudo`. This includes most editors, for example. Also, a program as innocuous as `/bin/cat` can be used to overwrite files, which could allow root to be exploited. Consider `sudo` as a means for accountability, and don't expect it to replace the root user and still be secure.

5.Files and Filesystem Security

A few minutes of preparation and planning ahead before putting your systems online can help to protect them and the data stored on them.

- There should never be a reason for users' home directories to allow SUID/SGID programs to be run from there. Use the `nosuid` option in `/etc/fstab` for partitions that are writable by others than root. You may also wish to use `nodev` and `noexec` on users' home partitions, as well as `/var`, thus prohibiting execution of programs, and creation of character or block devices, which should never be necessary anyway.
- If you are exporting filesystems using NFS, be sure to configure `/etc/exports` with the most restrictive access possible. This means not using wildcards, not allowing root write access, and exporting read−only wherever possible.
- Configure your users' file−creation `umask` to be as restrictive as possible. See umask settings.
- If you are mounting filesystems using a network filesystem such as NFS, be sure to configure /etc/exports with suitable restrictions. Typically, using `nodev', `nosuid', and perhaps `noexec', are desirable.
- Set filesystem limits instead of allowing `unlimited` as is the default. You can control the per−user limits using the resource−limits PAM module and `/etc/pam.d/limits.conf`. For example, limits for group `users` might look like this:

```
@users     hard  core    0
@users     hard  nproc   50
@users     hard  rss     5000
```

  This says to prohibit the creation of core files, restrict the number of processes to 50, and restrict memory usage per user to 5M.

- The `/var/log/wtmp` and `/var/run/utmp` files contain the login records for all users on your system. Their integrity must be maintained because they can be used to determine when and from where a user (or potential intruder) has entered your system. These files should also have `644` permissions, without affecting normal system operation.

- The immutable bit can be used to prevent accidentally deleting or overwriting a file that must be protected. It also prevents someone from creating a symbolic link to the file (such symbolic links have been the source of attacks involving deleting `/etc/passwd` or `/etc/shadow`). See the `chattr`(1) man page for information on the immutable bit.

- SUID and SGID files on your system are a potential security risk, and should be monitored closely. Because these programs grant special privileges to the user who is executing them, it is necessary to ensure that insecure programs are not installed. A favorite trick of crackers is to exploit SUID−root programs, then leave a SUID program as a backdoor to get in the next time, even if the original hole is plugged.

Find all SUID/SGID programs on your system, and keep track of what they are, so you are aware of any changes which could indicate a potential intruder. Use the following command to find all SUID/SGID programs on your system:

```
root#  find / -type f \( -perm -04000 -o -perm -02000 \)
```

The Debian distribution runs a job each night to determine what SUID files exist. It then compares this to the previous night's run. You can look in `/var/log/setuid*` for this log.

You can remove the SUID or SGID permissions on a suspicious program with `chmod`, then restore them back if you absolutely feel it is necessary.

- World−writable files, particularly system files, can be a security hole if a cracker gains access to your system and modifies them. Additionally, world−writable directories are dangerous, since they allow a cracker to add or delete files as he wishes. To locate all world−writable files on your system, use the following command:

```
root# find / -perm -2 ! -type l -ls
```

and be sure you know why those files are writable. In the normal course of operation, several files will be world−writable, including some from `/dev`, and symbolic links, thus the `! -type l` which excludes these from the previous `find` command.

- Unowned files may also be an indication an intruder has accessed your system. You can locate files on your system that have no owner, or belong to no group with the command:

```
root# find / -nouser -o -nogroup -print
```

- Finding `.rhosts` files should be a part of your regular system administration duties, as these files should not be permitted on your system. Remember, a cracker only needs one insecure account to potentially gain access to your entire network. You can locate all `.rhosts` files on your system with the following command:

```
root# find /home -name .rhosts -print
```

- Finally, before changing permissions on any system files, make sure you understand what you are doing. Never change permissions on a file because it seems like the easy way to get things working. Always determine why the file has that permission before changing it.

5.1 Umask Settings

The `umask` command can be used to determine the default file creation mode on your system. It is the octal complement of the desired file mode. If files are created without any regard to their permissions settings, the user could inadvertently give read or write permission to someone that should not have this permission. Typical `umask` settings include `022`, `027`, and `077` (which is the most restrictive). Normally the umask is set in `/etc/profile`, so it applies to all users on the system. The file creation mask can be calculated by subtracting the desired value from 777. In other words, a umask of 777 would cause newly−created files to contain no read, write or execute permission for anyone. A mask of 666 would cause newly−created files to have a mask of 111. For example, you may have a line that looks like this:

```
# Set the user's default umask
umask 033
```

Be sure to make root's umask `077`, which will disable read, write, and execute permission for other users, unless explicitly changed using `chmod`. In this case, newly−created directories would have 744 permissions, obtained by subtracting 033 from 777. Newly−created files using the 033 umask would have permissions of 644.

If you are using Red Hat, and adhere to their user and group ID creation scheme (User Private Groups), it is only necessary to use `002` for a `umask`. This is due to the fact that the default configuration is one user per group.

5.2 File Permissions

It's important to ensure that your system files are not open for casual editing by users and groups who shouldn't be doing such system maintenance.

Unix separates access control on files and directories according to three characteristics: owner, group, and other. There is always exactly one owner, any number of members of the group, and everyone else.

A quick explanation of Unix permissions:

Ownership − Which user(s) and group(s) retain(s) control of the permission settings of the node and parent of the node

Permissions − Bits capable of being set or reset to allow certain types of access to it. Permissions for directories may have a different meaning than the same set of permissions on files.

Read:

- To be able to view contents of a file
- To be able to read a directory

Write:

- To be able to add to or change a file

- To be able to delete or move files in a directory

Execute:

- To be able to run a binary program or shell script
- To be able to search in a directory, combined with read permission

Save Text Attribute: (For directories)

The "sticky bit" also has a different meaning when applied to directories than when applied to files. If the sticky bit is set on a directory, then a user may only delete files that the he owns or for which he has explicit write permission granted, even when he has write access to the directory. This is designed for directories like `/tmp`, which are world−writable, but where it may not be desirable to allow any user to delete files at will. The sticky bit is seen as a `t` in a long directory listing.

SUID Attribute: (For Files)

This describes set−user−id permissions on the file. When the set user ID access mode is set in the owner permissions, and the file is executable, processes which run it are granted access to system resources based on user who owns the file, as opposed to the user who created the process. This is the cause of many "buffer overflow" exploits.

SGID Attribute: (For Files)

If set in the group permissions, this bit controls the "set group id" status of a file. This behaves the same way as SUID, except the group is affected instead. The file must be executable for this to have any effect.

SGID Attribute: (For directories)

If you set the SGID bit on a directory (with `chmod g+s` *`directory`*), files created in that directory will have their group set to the directory's group.

You − The owner of the file

Group − The group you belong to

Everyone − Anyone on the system that is not the owner or a member of the group

File Example:

```
-rw-r--r--  1 kevin  users         114 Aug 28  1997 .zlogin
1st bit - directory?             (no)
 2nd bit - read by owner?         (yes, by kevin)
  3rd bit - write by owner?        (yes, by kevin)
   4th bit - execute by owner?      (no)
```

```
                5th bit − read by group?         (yes, by users)
                 6th bit − write by group?        (no)
                  7th bit − execute by group?      (no)
                   8th bit − read by everyone?      (yes, by everyone)
                    9th bit − write by everyone?     (no)
                     10th bit − execute by everyone?  (no)
```

The following lines are examples of the minimum sets of permissions that are required to perform the access described. You may want to give more permission than what's listed here, but this should describe what these minimum permissions on files do:

```
-r--------  Allow read access to the file by owner
--w-------  Allows the owner to modify or delete the file
            (Note that anyone with write permission to the directory
             the file is in can overwrite it and thus delete it)
---x------  The owner can execute this program, but not shell scripts,
             which still need read permission
---s------  Will execute with effective User ID = to owner
--------s-  Will execute with effective Group ID = to group
-rw------T  No update of "last modified time".  Usually used for swap
             files
---t------  No effect.  (formerly sticky bit)
```

Directory Example:

```
drwxr-xr-x  3 kevin  users         512 Sep 19 13:47 .public_html/
1st bit − directory?             (yes, it contains many files)
 2nd bit − read by owner?         (yes, by kevin)
  3rd bit − write by owner?        (yes, by kevin)
   4th bit − execute by owner?      (yes, by kevin)
    5th bit − read by group?         (yes, by users
     6th bit − write by group?        (no)
      7th bit − execute by group?      (yes, by users)
       8th bit − read by everyone?      (yes, by everyone)
        9th bit − write by everyone?     (no)
         10th bit − execute by everyone?  (yes, by everyone)
```

The following lines are examples of the minimum sets of permissions that are required to perform the access described. You may want to give more permission than what's listed, but this should describe what these minimum permissions on directories do:

```
dr--------  The contents can be listed, but file attributes can't be read
d--x------  The directory can be entered, and used in full execution
             paths
dr-x------  File attributes can be read by owner
d-wx------  Files can be created/deleted, even if the directory
             isn't the current one
d------x-t  Prevents files from deletion by others with write
```

```
                 access. Used on /tmp
d---s--s--  No effect
```

System configuration files (usually in `/etc`) are usually mode `640` (`-rw-r-----`), and owned by root. Depending on your site's security requirements, you might adjust this. Never leave any system files writable by a group or everyone. Some configuration files, including `/etc/shadow`, should only be readable by root, and directories in `/etc` should at least not be accessible by others.

SUID Shell Scripts

SUID shell scripts are a serious security risk, and for this reason the kernel will not honor them. Regardless of how secure you think the shell script is, it can be exploited to give the cracker a root shell.

5.3 Integrity Checking

Another very good way to detect local (and also network) attacks on your system is to run an integrity checker like `Tripwire`, `Aide` or `Osiris`. These integrety checkers run a number of checksums on all your important binaries and config files and compares them against a database of former, known−good values as a reference. Thus, any changes in the files will be flagged.

It's a good idea to install these sorts of programs onto a floppy, and then physically set the write protect on the floppy. This way intruders can't tamper with the integrety checker itself or change the database. Once you have something like this setup, it's a good idea to run it as part of your normal security administration duties to see if anything has changed.

You can even add a `crontab` entry to run the checker from your floppy every night and mail you the results in the morning. Something like:

```
# set mailto
MAILTO=kevin
# run Tripwire
15 05 * * * root /usr/local/adm/tcheck/tripwire
```

will mail you a report each morning at 5:15am.

Integrety checkers can be a godsend to detecting intruders before you would otherwise notice them. Since a lot of files change on the average system, you have to be careful what is cracker activity and what is your own doing.

You can find the open sourced version of `Tripwire` at http://www.tripwire.org, free of charge. Manuals and support can be purchased.

`Aide` can be found at http://www.cs.tut.fi/~rammer/aide.html.

`Osiris` can be found at http://www.shmoo.com/osiris/.

5.4 Trojan Horses

"Trojan Horses" are named after the fabled ploy in Homer's "Iliad". The idea is that a cracker distributes a program or binary that sounds great, and encourages other people to download it and run it as root. Then the program can compromise their system while they are not paying attention. While they think the binary they just pulled down does one thing (and it might very well), it also compromises their security.

You should take care of what programs you install on your machine. Redhat provides MD5 checksums and PGP signatures on it's RPM files so you can verify you are installing the real thing. Other distributions have similar methods. You should never run any unfamiliar binary, for which you don't have the source, as root! Few attackers are willing to release source code to public scrutiny.

Although it can be complex, make sure you are getting the source for a program from its real distribution site. If the program is going to run as root, make sure either you or someone you trust has looked over the source and verified it.

6.Password Security and Encryption

One of the most important security features used today are passwords. It is important for both you and all your users to have secure, unguessable passwords. Most of the more recent Linux distributions include `passwd` programs that do not allow you to set a easily guessable password. Make sure your `passwd` program is up to date and has these features.

In−depth discussion of encryption is beyond the scope of this document, but an introduction is in order. Encryption is very useful, possibly even necessary in this day and age. There are all sorts of methods of encrypting data, each with its own set of characteristics.

Most Unicies (and Linux is no exception) primarily use a one−way encryption algorithm, called DES (Data Encryption Standard) to encrypt your passwords. This encrypted password is then stored in (typically) `/etc/passwd` (or less commonly) `/etc/shadow`. When you attempt to login, the password you type in is encrypted again and compared with the entry in the file that stores your passwords. If they match, it must be the same password, and you are allowed access. Although DES is a two−way encryption algorithm (you can code and then decode a message, given the right keys), the variant that most unices use is one−way. This means that it should not be possible to reverse the encryption to get the password from the contents of `/etc/passwd` (or `/etc/shadow`).

Brute force attacks, such as "Crack" or "John the Ripper" (see Section crack) can often guess passwords unless your password is sufficiently random. PAM modules (see below) allow you to use a different encryption routine with your passwords (MD5 or the like). You can use Crack to your advantage, as well. Consider periodically running Crack against your own password database, to find insecure passwords. Then contact the offending user, and instruct him to change his password.

You can go to http://consult.cern.ch/writeup/security/security_3.html for information on how to choose a good password.

6.1 PGP and Public−Key Cryptography

Public−key cryptography, such as that used for PGP, uses one key for encryption, and one key for decryption. Traditional cryptography, however, uses the same key for encryption and decryption; this key must be known to both parties, and thus somehow transferred from one to the other securely.

To alleviate the need to securely transmit the encryption key, public−key encryption uses two separate keys: a public key and a private key. Each person's public key is available by anyone to do the encryption, while at the same time each person keeps his or her private key to decrypt messages encrypted with the correct public key.

There are advantages to both public key and private key cryptography, and you can read about those differences in the RSA Cryptography FAQ, listed at the end of this section.

PGP (Pretty Good Privacy) is well−supported on Linux. Versions 2.6.2 and 5.0 are known to work well. For a good primer on PGP and how to use it, take a look at the PGP FAQ: http://www.pgp.com/service/export/faq/55faq.cgi

Be sure to use the version that is applicable to your country. Due to export restrictions by the US Government, strong−encryption is prohibited from being transferred in electronic form outside the country.

US export controls are now managed by EAR (Export Administration Regulations). They are no longer governed by ITAR.

There is also a step−by−step guide for configuring PGP on Linux available at http://mercury.chem.pitt.edu/~angel/LinuxFocus/English/November1997/article7.html. It was written for the international version of PGP, but is easily adaptable to the United States version. You may also need a patch for some of the latest versions of Linux; the patch is available at ftp://metalab.unc.edu/pub/Linux/apps/crypto.

There is a project working on a free re−implementation of pgp with open source. GnuPG is a complete and free replacement for PGP. Because it does not use IDEA or RSA it can be used without any restrictions. GnuPG is nearly in compliance with OpenPGP. See the GNU Privacy Guard web page for more information: http://www.gnupg.org/.

More information on cryptography can be found in the RSA cryptography FAQ, available at http://www.rsa.com/rsalabs/newfaq/. Here you will find information on such terms as "Diffie−Hellman", "public−key cryptography", "digital certificates", etc.

6.2 SSL, S−HTTP, HTTPS and S/MIME

Often users ask about the differences between the various security and encryption protocols, and how to use them. While this isn't an encryption document, it is a good idea to explain briefly what each protocol is, and where to find more information.

- **SSL:** − SSL, or Secure Sockets Layer, is an encryption method developed by Netscape to provide security over the Internet. It supports several different encryption protocols, and provides client and server authentication. SSL operates at the transport layer, creates a secure encrypted channel of data, and thus can seamlessly encrypt data of many types. This is most commonly seen when going to a secure site to view a secure online document with Communicator, and serves as the basis for secure communications with Communicator, as well as many other Netscape Communications data encryption. More information can be found at http://www.consensus.com/security/ssl−talk−faq.html. Information on Netscape's other security implementations, and a good starting point for these protocols is available at http://home.netscape.com/info/security−doc.html.

- **S−HTTP:** − S−HTTP is another protocol that provides security services across the Internet. It was designed to provide confidentiality, authentication, integrity, and non−repudiability [cannot be mistaken for someone else] while supporting multiple key−management mechanisms and cryptographic algorithms via option negotiation between the parties involved in each transaction. S−HTTP is limited to the specific software that is implementing it, and encrypts each message individually. [From RSA Cryptography FAQ, page 138]
- **S/MIME:** − S/MIME, or Secure Multipurpose Internet Mail Extension, is an encryption standard used to encrypt electronic mail and other types of messages on the Internet. It is an open standard developed by RSA, so it is likely we will see it on Linux one day soon. More information on S/MIME can be found at http://home.netscape.com/assist/security/smime/overview.html.

6.3 Linux IPSEC Implementations

Along with CIPE, and other forms of data encryption, there are also several other implementations of IPSEC for Linux. IPSEC is an effort by the IETF to create cryptographically−secure communications at the IP network level, and to provide authentication, integrity, access control, and confidentiality. Information on IPSEC and Internet draft can be found at http://www.ietf.org/html.charters/ipsec−charter.html. You can also find links to other protocols involving key management, and an IPSEC mailing list and archives.

The x−kernel Linux implementation, which is being developed at the University of Arizona, uses an object−based framework for implementing network protocols called x−kernel, and can be found at http://www.cs.arizona.edu/xkernel/hpcc−blue/linux.html. Most simply, the x−kernel is a method of passing messages at the kernel level, which makes for an easier implementation.

Another freely−available IPSEC implementation is the Linux FreeS/WAN IPSEC. Their web page states,

> "These services allow you to build secure tunnels through untrusted networks. Everything passing through the untrusted net is encrypted by the IPSEC gateway machine and decrypted by the gateway at the other end. The result is Virtual Private Network or VPN. This is a network which is effectively private even though it includes machines at several different sites connected by the insecure Internet."

It's available for download from http://www.xs4all.nl/~freeswan/, and has just reached 1.0 at the time of this writing.

As with other forms of cryptography, it is not distributed with the kernel by default due to export restrictions.

6.4 `ssh` (Secure Shell) and `stelnet`

`ssh` and `stelnet` are suite's of programs that allow you to login to remote systems and have a encrypted connection.

`openssh` is a suite of programs used as a secure replacement for `rlogin`, `rsh` and `rcp`. It uses public−key cryptography to encrypt communications between two hosts, as well as to authenticate users. It can be used to securely login to a remote host or copy data between hosts, while preventing man−in−the−middle attacks (session hijacking) and DNS spoofing. It will perform data compression on your connections, and secure X11 communications between hosts.

There are several ssh implementiations now. The original commercial implementation by Data Fellows can be found at The `ssh` home page can be found at http://www.datafellows.com.

The excellent Openssh implementation is based on a early version of the datafellows ssh and has been totally reworked to not include any patented or propriatary peices. It is free and under a BSD liscence. It can be found at: http://www.openssh.com.

There is also a open source project to re−implement ssh from the ground up called "psst...". For more information see: http://www.net.lut.ac.uk/psst/

You can also use `ssh` from your Windows workstation to your Linux `ssh` server. There are several freely available Windows client implementations, including the one at http://guardian.htu.tuwien.ac.at/therapy/ssh/ as well as a commercial implementation from DataFellows, at http://www.datafellows.com.

SSLeay is a free implementation of Netscape's Secure Sockets Layer protocol, developed by Eric Young. It includes several applications, such as Secure telnet, a module for Apache, several databases, as well as several algorithms including DES, IDEA and Blowfish.

Using this library, a secure telnet replacement has been created that does encryption over a telnet connection. Unlike SSH, stelnet uses SSL, the Secure Sockets Layer protocol developed by Netscape. You can find Secure telnet and Secure FTP by starting with the SSLeay FAQ, available at http://www.psy.uq.oz.au/~ftp/Crypto/.

SRP is another secure telnet/ftp implementation. From their web page:

> "The SRP project is developing secure Internet software for free worldwide use. Starting with a fully−secure Telnet and FTP distribution, we hope to supplant weak networked authentication systems with strong replacements that do not sacrifice user−friendliness for security. Security should be the default, not an option!"

For more information, go to http://srp.stanford.edu/srp.

6.5 PAM − Pluggable Authentication Modules

Newer versions of the Red Hat Linux distribution ship with a unified authentication scheme called "PAM". PAM allows you to change your authentication methods and requirements on the fly, and encapsulate all local authentication methods without recompiling any of your binaries. Configuration of PAM is beyond the scope of this document, but be sure to take a look at the PAM web site for more information. http://www.kernel.org/pub/linux/libs/pam/index.html.

Just a few of the things you can do with PAM:

- Use encryption other than DES for your passwords. (Making them harder to brute−force decode)
- Set resource limits on all your users so they can't perform denial−of−service attacks (number of processes, amount of memory, etc)
- Enable shadow passwords (see below) on the fly
- allow specific users to login only at specific times from specific places

Within a few hours of installing and configuring your system, you can prevent many attacks before they even occur. For example, use PAM to disable the system−wide usage of `.rhosts` files in user's home directories by adding these lines to `/etc/pam.d/rlogin`:

```
#
# Disable rsh/rlogin/rexec for users
#
login auth required pam_rhosts_auth.so no_rhosts
```

6.6 Cryptographic IP Encapsulation (CIPE)

The primary goal of this software is to provide a facility for secure (against eavesdropping, including traffic analysis, and faked message injection) subnetwork interconnection across an insecure packet network such as the Internet.

CIPE encrypts the data at the network level. Packets traveling between hosts on the network are encrypted. The encryption engine is placed near the driver which sends and receives packets.

This is unlike SSH, which encrypts the data by connection, at the socket level. A logical connection between programs running on different hosts is encrypted.

CIPE can be used in tunnelling, in order to create a Virtual Private Network. Low−level encryption has the advantage that it can be made to work transparently between the two networks connected in the VPN, without any change to application software.

Summarized from the CIPE documentation:

> The IPSEC standards define a set of protocols which can be used (among other things) to build encrypted VPNs. However, IPSEC is a rather heavyweight and complicated protocol

> set with a lot of options, implementations of the full protocol set are still rarely used and some issues (such as key management) are still not fully resolved. CIPE uses a simpler approach, in which many things which can be parameterized (such as the choice of the actual encryption algorithm used) are an install−time fixed choice. This limits flexibility, but allows for a simple (and therefore efficient, easy to debug...) implementation.

Further information can be found at http://www.inka.de/~bigred/devel/cipe.html

As with other forms of cryptography, it is not distributed with the kernel by default due to export restrictions.

6.7 Kerberos

Kerberos is an authentication system developed by the Athena Project at MIT. When a user logs in, Kerberos authenticates that user (using a password), and provides the user with a way to prove her identity to other servers and hosts scattered around the network.

This authentication is then used by programs such as `rlogin` to allow the user to login to other hosts without a password (in place of the `.rhosts` file). This authentication method can also used by the mail system in order to guarantee that mail is delivered to the correct person, as well as to guarantee that the sender is who he claims to be.

Kerberos and the other programs that come with it, prevent users from "spoofing" the system into believing they are someone else. Unfortunately, installing Kerberos is very intrusive, requiring the modification or replacement of numerous standard programs.

You can find more information about kerberos by looking at the kerberos FAQ, and the code can be found at http://nii.isi.edu/info/kerberos/.

[From: Stein, Jennifer G., Clifford Neuman, and Jeffrey L. Schiller. "Kerberos: An Authentication Service for Open Network Systems." USENIX Conference Proceedings, Dallas, Texas, Winter 1998.]

Kerberos should not be your first step in improving security of your host. It is quite involved, and not as widely used as, say, SSH.

6.8 Shadow Passwords.

Shadow passwords are a means of keeping your encrypted password information secret from normal users. Recent versions of both Red Hat and Debian Linux use shadow passwords by default, but on other systems, encrypted passwords are stored in `/etc/passwd` file for all to read. Anyone can then run password−guesser programs on them and attempt to determine what they are. Shadow passwords, by contrast, are saved in `/etc/shadow`, which only privileged users can read. In order to use shadow passwords, you need to make sure all your utilities that need access to password information are recompiled to support them. PAM (above) also allows you to just plug in a shadow module; it doesn't require re−compilation of executables. You can refer to the Shadow−Password HOWTO for further information if necessary. It is available at http://metalab.unc.edu/LDP/HOWTO/Shadow−Password−HOWTO.html It is rather dated now, and will not be required for distributions supporting PAM.

6.9 "Crack" and "John the Ripper"

If for some reason your `passwd` program is not enforcing hard−to−guess passwords, you might want to run a password−cracking program and make sure your users' passwords are secure.

Password cracking programs work on a simple idea: they try every word in the dictionary, and then variations on those words, encrypting each one and checking it against your encrypted password. If they get a match they know what your password is.

There are a number of programs out there...the two most notable of which are "Crack" and "John the Ripper" (http://www.false.com/security/john/index.html) . They will take up a lot of your cpu time, but you should be able to tell if an attacker could get in using them by running them first yourself and notifying users with weak passwords. Note that an attacker would have to use some other hole first in order to read your `/etc/passwd` file, but such holes are more common than you might think.

Because security is only as strong as the most insecure host, it is worth mentioning that if you have any Windows machines on your network, you should check out L0phtCrack, a Crack implementation for Windows. It's available from http://www.l0pht.com

6.10 CFS − Cryptographic File System and TCFS − Transparent Cryptographic File System

CFS is a way of encrypting entire directory trees and allowing users to store encrypted files on them. It uses an NFS server running on the local machine. RPMS are available at http://www.zedz.net/redhat/, and more information on how it all works is at ftp://ftp.research.att.com/dist/mab/.

TCFS improves on CFS by adding more integration with the file system, so that it's transparent to users that the file system that is encrypted. More information at: http://edu−gw.dia.unisa.it/tcfs/.

It also need not be used on entire filesystems. It works on directory trees as well.

6.11 X11, SVGA and display security

X11

It's important for you to secure your graphical display to prevent attackers from grabbing your passwords as you type them, reading documents or information you are reading on your screen, or even using a hole to gain root access. Running remote X applications over a network also can be fraught with peril, allowing sniffers to see all your interaction with the remote system.

X has a number of access−control mechanisms. The simplest of them is host−based: you use `xhost` to specify the hosts that are allowed access to your display. This is not very secure at all, because if someone has access to your machine, they can `xhost + their machine` and get in easily. Also, if you have to allow access from an untrusted machine, anyone there can compromise your display.

When using `xdm` (X Display Manager) to log in, you get a much better access method: MIT−MAGIC−COOKIE−1. A 128−bit "cookie" is generated and stored in your `.Xauthority` file. If you need to allow a remote machine access to your display, you can use the `xauth` command and the information in your `.Xauthority` file to provide access to only that connection. See the Remote−X−Apps mini−howto, available at http://metalab.unc.edu/LDP/HOWTO/mini/Remote−X−Apps.html.

You can also use `ssh` (see ssh , above) to allow secure X connections. This has the advantage of also being transparent to the end user, and means that no unencrypted data flows across the network.

Take a look at the `Xsecurity` man page for more information on X security. The safe bet is to use `xdm` to login to your console and then use `ssh` to go to remote sites on which you wish to run X programs.

SVGA

SVGAlib programs are typically SUID−root in order to access all your Linux machine's video hardware. This makes them very dangerous. If they crash, you typically need to reboot your machine to get a usable console back. Make sure any SVGA programs you are running are authentic, and can at least be somewhat trusted. Even better, don't run them at all.

GGI (Generic Graphics Interface project)

The Linux GGI project is trying to solve several of the problems with video interfaces on Linux. GGI will move a small piece of the video code into the Linux kernel, and then control access to the video system. This means GGI will be able to restore your console at any time to a known good state. They will also allow a secure attention key, so you can be sure that there is no Trojan horse `login` program running on your console. http://synergy.caltech.edu/~ggi/

7.Kernel Security

This is a description of the kernel configuration options that relate to security, and an explanation of what they do, and how to use them.

As the kernel controls your computer's networking, it is important that it be very secure, and not be compromised. To prevent some of the latest networking attacks, you should try to keep your kernel version current. You can find new kernels at ftp://ftp.kernel.org or from your distribution vendor.

There is also a international group providing a single unified crypto patch to the mainstream Lnux kernel. This patch provides support for a number of cyrptographic subsystems and things that cannot be included in the mainstream kernel due to export restrictions. For more information, visit their web page at:

http://www.kerneli.org

7.1 2.0 Kernel Compile Options

For 2.0.x kernels, the following options apply. You should see these options during the kernel configuration process. Many of the comments here are from `./linux/Documentation/Configure.help`, which is the same document that is referenced while using the Help facility during the `make config` stage of compiling the kernel.

- Network Firewalls (CONFIG_FIREWALL)

  This option should be on if you intend to run any firewalling or masquerading on your Lnux machine. If it's just going to be a regular client machine, it's safe to say no.

- IP: forwarding/gatewaying (CONFIG_IP_FORWARD)

  If you enable IP forwarding, your Linux box essentially becomes a router. If your machine is on a network, you could be forwarding data from one network to another, and perhaps subverting a firewall that was put there to prevent this from happening. Normal dial−up users will want to disable this, and other users should concentrate on the security implications of doing this. Firewall machines will want this enabled, and used in conjunction with firewall software.

  You can enable IP forwarding dynamically using the following command:

  ```
  root#  echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

  and disable it with the command:

  ```
  root#  echo 0 > /proc/sys/net/ipv4/ip_forward
  ```

  Keep in mind the files in /proc are "virtual" files and the shown size of the file might not reflect the data output from it.

- IP: syn cookies (CONFIG_SYN_COOKIES)

  a "SYN Attack" is a denial of service (DoS) attack that consumes all the resources on your machine, forcing you to reboot. We can't think of a reason you wouldn't normally enable this. In the 2.2.x kernel series this config option merely allows syn cookies, but does not enable them. To enable them, you have to do:

  ```
  root# echo 1 > /proc/sys/net/ipv4/tcp_syncookies <P>
  ```

- IP: Firewalling (CONFIG_IP_FIREWALL)

This option is necessary if you are going to configure your machine as a firewall, do masquerading, or wish to protect your dial−up workstation from someone entering via your PPP dial−up interface.

- IP: firewall packet logging (CONFIG_IP_FIREWALL_VERBOSE)

  This option gives you information about packets your firewall received, like sender, recipient, port, etc.
- IP: Drop source routed frames (CONFIG_IP_NOSR)

  This option should be enabled. Source routed frames contain the entire path to their destination inside of the packet. This means that routers through which the packet goes do not need to inspect it, and just forward it on. This could lead to data entering your system that may be a potential exploit.
- IP: masquerading (CONFIG_IP_MASQUERADE) If one of the computers on your local network for which your Linux box acts as a firewall wants to send something to the outside, your box can "masquerade" as that host, i.e., it forwards the traffic to the intended destination, but makes it look like it came from the firewall box itself. See http://www.indyramp.com/masq for more information.
- IP: ICMP masquerading (CONFIG_IP_MASQUERADE_ICMP) This option adds ICMP masquerading to the previous option of only masquerading TCP or UDP traffic.
- IP: transparent proxy support (CONFIG_IP_TRANSPARENT_PROXY) This enables your Linux firewall to transparently redirect any network traffic originating from the local network and destined for a remote host to a local server, called a "transparent proxy server". This makes the local computers think they are talking to the remote end, while in fact they are connected to the local proxy. See the IP−Masquerading HOWTO and http://www.indyramp.com/masq for more information.
- IP: always defragment (CONFIG_IP_ALWAYS_DEFRAG)

  Generally this option is disabled, but if you are building a firewall or a masquerading host, you will want to enable it. When data is sent from one host to another, it does not always get sent as a single packet of data, but rather it is fragmented into several pieces. The problem with this is that the port numbers are only stored in the first fragment. This means that someone can insert information into the remaining packets that isn't supposed to be there. It could also prevent a teardrop attack against an internal host that is not yet itself patched against it.
- Packet Signatures (CONFIG_NCPFS_PACKET_SIGNING)

  This is an option that is available in the 2.2.x kernel series that will sign NCP packets for stronger security. Normally you can leave it off, but it is there if you do need it.
- IP: Firewall packet netlink device (CONFIG_IP_FIREWALL_NETLINK)

  This is a really neat option that allows you to analyze the first 128 bytes of the packets in a user−space program, to determine if you would like to accept or deny the packet, based on its validity.

7.2 2.2 Kernel Compile Options

For 2.2.x kernels, many of the options are the same, but a few new ones have been developed. Many of the comments here are from `./linux/Documentation/Configure.help`, which is the same document that is referenced while using the Help facility during the `make config` stage of compiling the kernel. Only the newly− added options are listed below. Consult the 2.0 description for a list of other necessary options. The most significant change in the 2.2 kernel series is the IP firewalling code. The `ipchains` program is now used to install IP firewalling, instead of the `ipfwadm` program used in the 2.0 kernel.

- Socket Filtering (CONFIG_FILTER)

  For most people, it's safe to say no to this option. This option allows you to connect a userspace filter to any socket and determine if packets should be allowed or denied. Unless you have a very specific need and are capable of programming such a filter, you should say no. Also note that as of this writing, all protocols were supported except TCP.
- Port Forwarding Port Forwarding is an addition to IP Masquerading which allows some forwarding of packets from outside to inside a firewall on given ports. This could be useful if, for example, you want to run a web server behind the firewall or masquerading host and that web server should be accessible from the outside world. An external client sends a request to port 80 of the firewall, the firewall forwards this request to the web server, the web server handles the request and the results are sent through the firewall to the original client. The client thinks that the firewall machine itself is running the web server. This can also be used for load balancing if you have a farm of identical web servers behind the firewall. Information about this feature is available from http://www.monmouth.demon.co.uk/ipsubs/portforwarding.html (to browse the WWW, you need to have access to a machine on the Internet that has a program like lynx or netscape). For general info, please see ftp://ftp.compsoc.net/users/steve/ipportfw/linux21/
- Socket Filtering (CONFIG_FILTER) Using this option, user−space programs can attach a filter to any socket and thereby tell the kernel that it should allow or disallow certain types of data to get through the socket. Linux socket filtering works on all socket types except TCP for now. See the text file `./linux/Documentation/networking/filter.txt` for more information.
- IP: Masquerading The 2.2 kernel masquerading has been improved. It provides additional support for masquerading special protocols, etc. Be sure to read the IP Chains HOWTO for more information.

7.3 Kernel Devices

There are a few block and character devices available on Linux that will also help you with security.

The two devices `/dev/random` and `/dev/urandom` are provided by the kernel to provide random data at any time.

Both `/dev/random` and `/dev/urandom` should be secure enough to use in generating PGP keys, `ssh` challenges, and other applications where secure random numbers are required. Attackers should be unable to predict the next number given any initial sequence of numbers from these sources. There has been a lot of effort put in to ensuring that the numbers you get from these sources are random in every sense of the word.

The only difference between the two devices, is that `/dev/random` runs out of random bytes and it makes you wait for more to be accumulated. Note that on some systems, it can block for a long time waiting for new user−generated entropy to be entered into the system. So you have to use care before using `/dev/random`. (Perhaps the best thing to do is to use it when you're generating sensitive keying information, and you tell the user to pound on the keyboard repeatedly until you print out "OK, enough".)

`/dev/random` is high quality entropy, generated from measuring the inter−interrupt times etc. It blocks until enough bits of random data are available.

`/dev/urandom` is similar, but when the store of entropy is running low, it'll return a cryptographically strong hash of what there is. This isn't as secure, but it's enough for most applications.

You might read from the devices using something like:

```
                root#  head −c 6 /dev/urandom | mimencode
```

This will print eight random characters on the console, suitable for password generation. You can find `mimencode` in the `metamail` package.

See `/usr/src/linux/drivers/char/random.c` for a description of the algorithm.

Thanks to Theodore Y. Ts'o, Jon Lewis, and others from Linux−kernel for helping me (Dave) with this.

8.Network Security

Network security is becoming more and more important as people spend more and more time connected. Compromising network security is often much easier than compromising physical or local security, and is much more common.

There are a number of good tools to assist with network security, and more and more of them are shipping with Linux distributions.

8.1 Packet Sniffers

One of the most common ways intruders gain access to more systems on your network is by employing a packet sniffer on a already compromised host. This "sniffer" just listens on the Ethernet port for things like `passwd` and `login` and `su` in the packet stream and then logs the traffic after that. This way, attackers gain passwords for systems they are not even attempting to break into. Clear−text passwords are very vulnerable to this attack.

Example: Host A has been compromised. Attacker installs a sniffer. Sniffer picks up admin logging into Host

B from Host C. It gets the admin's personal password as they login to B. Then, the admin does a `su` to fix a problem. They now have the root password for Host B. Later the admin lets someone `telnet` from his account to Host Z on another site. Now the attacker has a password/login on Host Z.

In this day and age, the attacker doesn't even need to compromise a system to do this: they could also bring a laptop or pc into a building and tap into your net.

Using `ssh` or other encrypted password methods thwarts this attack. Things like APOP for POP accounts also prevents this attack. (Normal POP logins are very vulnerable to this, as is anything that sends clear−text passwords over the network.)

8.2 System services and tcp_wrappers

Before you put your Linux system on *ANY* network the first thing to look at is what services you need to offer. Services that you do not need to offer should be disabled so that you have one less thing to worry about and attackers have one less place to look for a hole.

There are a number of ways to disable services under Linux. You can look at your `/etc/inetd.conf` file and see what services are being offered by your `inetd`. Disable any that you do not need by commenting them out (`#` at the beginning of the line), and then sending your inetd process a SIGHUP.

You can also remove (or comment out) services in your `/etc/services` file. This will mean that local clients will also be unable to find the service (i.e., if you remove `ftp`, and try and ftp to a remote site from that machine it will fail with an "unknown service" message). It's usually not worth the trouble to remove services from `/etc/services`, since it provides no additional security. If a local person wanted to use `ftp` even though you had commented it out, they would make their own client that used the common FTP port and would still work fine.

Some of the services you might want to leave enabled are:

- `ftp`
- `telnet` (or `ssh`)
- mail, such as `pop−3` or `imap`
- `identd`

If you know you are not going to use some particular package, you can also delete it entirely. `rpm −e` *`packagename`* under the Red Hat distribution will erase an entire package. Under debian `dpkg −−remove` does the same thing.

Additionally, you really want to disable the rsh/rlogin/rcp utilities, including login (used by `rlogin`), shell (used by `rcp`), and exec (used by `rsh`) from being started in `/etc/inetd.conf`. These protocols are extremely insecure and have been the cause of exploits in the past.

You should check `/etc/rc.d/rc[0−9].d` (on Red Hat; `/etc/rc[0−9].d` on Debian), and see if any of the servers started in those directories are not needed. The files in those directories are actually symbolic links to files in the directory `/etc/rc.d/init.d` (on Red Hat; `/etc/init.d` on Debian). Renaming the files in the `init.d` directory disables all the symbolic links that point to that file. If you only wish to disable a service for a particular run level, rename the appropriate symbolic link by replacing the upper−case

S with a lower−case s, like this:

```
root#  cd /etc/rc6.d
root#  mv S45dhcpd s45dhcpd
```

If you have BSD−style rc files, you will want to check /etc/rc* for programs you don't need.

Most Linux distributions ship with tcp_wrappers "wrapping" all your TCP services. A tcp_wrapper (tcpd) is invoked from inetd instead of the real server. tcpd then checks the host that is requesting the service, and either executes the real server, or denies access from that host. tcpd allows you to restrict access to your TCP services. You should make a /etc/hosts.allow and add in only those hosts that need to have access to your machine's services.

If you are a home dialup user, we suggest you deny ALL. tcpd also logs failed attempts to access services, so this can alert you if you are under attack. If you add new services, you should be sure to configure them to use tcp_wrappers if they are TCP−based. For example, a normal dial−up user can prevent outsiders from connecting to his machine, yet still have the ability to retrieve mail, and make network connections to the Internet. To do this, you might add the following to your /etc/hosts.allow:

ALL: 127.

And of course /etc/hosts.deny would contain:

ALL: ALL

which will prevent external connections to your machine, yet still allow you from the inside to connect to servers on the Internet.

Keep in mind that tcp_wrappers only protects services executed from inetd, and a select few others. There very well may be other services running on your machine. You can use netstat −ta to find a list of all the services your machine is offering.

8.3 Verify Your DNS Information

Keeping up−to−date DNS information about all hosts on your network can help to increase security. If an unauthorized host becomes connected to your network, you can recognize it by its lack of a DNS entry. Many services can be configured to not accept connections from hosts that do not have valid DNS entries.

8.4 identd

identd is a small program that typically runs out of your inetd server. It keeps track of what user is running what TCP service, and then reports this to whoever requests it.

Many people misunderstand the usefulness of identd, and so disable it or block all off site requests for it.

`identd` is not there to help out remote sites. There is no way of knowing if the data you get from the remote `identd` is correct or not. There is no authentication in `identd` requests.

Why would you want to run it then? Because it helps *you* out, and is another data−point in tracking. If your `identd` is un compromised, then you know it's telling remote sites the user−name or uid of people using TCP services. If the admin at a remote site comes back to you and tells you user so−and−so was trying to hack into their site, you can easily take action against that user. If you are not running `identd`, you will have to look at lots and lots of logs, figure out who was on at the time, and in general take a lot more time to track down the user.

The `identd` that ships with most distributions is more configurable than many people think. You can disable it for specific users (they can make a `.noident` file), you can log all `identd` requests (We recommend it), you can even have identd return a uid instead of a user name or even NO−USER.

8.5 SATAN, ISS, and Other Network Scanners

There are a number of different software packages out there that do port and service−based scanning of machines or networks. SATAN, ISS, SAINT, and Nessus are some of the more well−known ones. This software connects to the target machine (or all the target machines on a network) on all the ports they can, and try to determine what service is running there. Based on this information, you can tell if the machine is vulnerable to a specific exploit on that server.

SATAN (Security Administrator's Tool for Analyzing Networks) is a port scanner with a web interface. It can be configured to do light, medium, or strong checks on a machine or a network of machines. It's a good idea to get SATAN and scan your machine or network, and fix the problems it finds. Make sure you get the copy of SATAN from metalab or a reputable FTP or web site. There was a Trojan copy of SATAN that was distributed out on the net. http://www.trouble.org/~zen/satan/satan.html. Note that SATAN has not been updated in quite a while, and some of the other tools below might do a better job.

ISS (Internet Security Scanner) is another port−based scanner. It is faster than Satan, and thus might be better for large networks. However, SATAN tends to provide more information.

Abacus is a suite of tools to provide host−based security and intrusion detection. Look at it's home page on the web for more information. http://www.psionic.com/abacus/

SAINT is a updated version of SATAN. It is web−based and has many more up−to−date tests than SATAN. You can find out more about it at: http://www.wwdsi.com/~saint

Nessus is a free security scanner. It has a GTK graphical interface for ease of use. It is also designed with a very nice plugin setup for new port−scanning tests. For more information, take a look at: http://www.nessus.org

Detecting Port Scans

There are some tools designed to alert you to probes by SATAN and ISS and other scanning software. However, if you liberally use tcp_wrappers, and look over your log files regularly, you should be able to notice such probes. Even on the lowest setting, SATAN still leaves traces in the logs on a stock Red Hat system.

There are also "stealth" port scanners. A packet with the TCP ACK bit set (as is done with established connections) will likely get through a packet−filtering firewall. The returned RST packet from a port that *_had no established session_* can be taken as proof of life on that port. I don't think TCP wrappers will detect this.

8.6 `sendmail`, `qmail` and MTA's

One of the most important services you can provide is a mail server. Unfortunately, it is also one of the most vulnerable to attack, simply due to the number of tasks it must perform and the privileges it typically needs.

If you are using `sendmail` it is very important to keep up on current versions. `sendmail` has a long long history of security exploits. Always make sure you are running the most recent version from http://www.sendmail.org.

Keep in mind that sendmail does not have to be running in order for you to send mail. If you are a home user, you can disable sendmail entirely, and simply use your mail client to send mail. You might also choose to remove the "−bd" flag from the sendmail startup file, thereby disabling incoming requests for mail. In other words, you can execute sendmail from your startup script using the following instead:

```
# /usr/lib/sendmail −q15m
```

This will cause sendmail to flush the mail queue every fifteen minutes for any messages that could not be successfully delivered on the first attempt.

Many administrators choose not to use sendmail, and instead choose one of the other mail transport agents. You might consider switching over to `qmail`. `qmail` was designed with security in mind from the ground up. It's fast, stable, and secure. Qmail can be found at http://www.qmail.org

In direct competition to qmail is "postfix", written by Wietse Venema, the author of tcp_wrappers and other security tools. Formerly called vmailer, and sponsored by IBM, this is also a mail transport agent written from the ground up with security in mind. You can find more information about postfix at http://www.postfix.org

8.7 Denial of Service Attacks

A "Denial of Service" (DoS) attack is one where the attacker tries to make some resource too busy to answer legitimate requests, or to deny legitimate users access to your machine.

Denial of service attacks have increased greatly in recent years. Some of the more popular and recent ones are listed below. Note that new ones show up all the time, so this is just a few examples. Read the Linux security lists and the bugtraq list and archives for more current information.

- **SYN Flooding** − SYN flooding is a network denial of service attack. It takes advantage of a "loophole" in the way TCP connections are created. The newer Linux kernels (2.0.30 and up) have several configurable options to prevent SYN flood attacks from denying people access to your machine or services. See Kernel Security for proper kernel protection options.

- **Pentium "F00F" Bug −** It was recently discovered that a series of assembly codes sent to a genuine Intel Pentium processor would reboot the machine. This affects every machine with a Pentium processor (not clones, not Pentium Pro or PII), no matter what operating system it's running. Linux kernels 2.0.32 and up contain a work around for this bug, preventing it from locking your machine. Kernel 2.0.33 has an improved version of the kernel fix, and is suggested over 2.0.32. If you are running on a Pentium, you should upgrade now!
- **Ping Flooding −** Ping flooding is a simple brute−force denial of service attack. The attacker sends a "flood" of ICMP packets to your machine. If they are doing this from a host with better bandwidth than yours, your machine will be unable to send anything on the network. A variation on this attack, called "smurfing", sends ICMP packets to a host with *your* machine's return IP, allowing them to flood you less detectably. You can find more information about the "smurf" attack at http://www.quadrunner.com/~chuegen/smurf.txt

  If you are ever under a ping flood attack, use a tool like `tcpdump` to determine where the packets are coming from (or appear to be coming from), then contact your provider with this information. Ping floods can most easily be stopped at the router level or by using a firewall.
- **Ping o' Death −** The Ping o' Death attack sends ICMP ECHO REQUEST packets that are too large to fit in the kernel data structures intended to store them. Because sending a single, large (65,510 bytes) "ping" packet to many systems will cause them to hang or even crash, this problem was quickly dubbed the "Ping o' Death." This one has long been fixed, and is no longer anything to worry about.
- **Teardrop / New Tear −** One of the most recent exploits involves a bug present in the IP fragmentation code on Linux and Windows platforms. It is fixed in kernel version 2.0.33, and does not require selecting any kernel compile−time options to utilize the fix. Linux is apparently not vulnerable to the "newtear" exploit.

You can find code for most exploits, and a more in−depth description of how they work, at http://www.rootshell.com using their search engine.

8.8 NFS (Network File System) Security.

NFS is a very widely−used file sharing protocol. It allows servers running `nfsd` and `mountd` to "export" entire filesystems to other machines using NFS filesystem support built in to their kernels (or some other client support if they are not Linux machines). `mountd` keeps track of mounted filesystems in `/etc/mtab`, and can display them with `showmount`.

Many sites use NFS to serve home directories to users, so that no matter what machine in the cluster they login to, they will have all their home files.

There is some small amount of security allowed in exporting filesystems. You can make your `nfsd` map the remote root user (uid=0) to the `nobody` user, denying them total access to the files exported. However, since individual users have access to their own (or at least the same uid) files, the remote root user can login or `su` to their account and have total access to their files. This is only a small hindrance to an attacker that has access to mount your remote filesystems.

If you must use NFS, make sure you export to only those machines that you really need to. Never export your entire root directory; export only directories you need to export.

See the NFS HOWTO for more information on NFS, available at http://metalab.unc.edu/mdw/HOWTO/NFS−HOWTO.html

8.9 NIS (Network Information Service) (formerly YP).

Network Information service (formerly YP) is a means of distributing information to a group of machines. The NIS master holds the information tables and converts them into NIS map files. These maps are then served over the network, allowing NIS client machines to get login, password, home directory and shell information (all the information in a standard `/etc/passwd` file). This allows users to change their password once and have it take effect on all the machines in the NIS domain.

NIS is not at all secure. It was never meant to be. It was meant to be handy and useful. Anyone that can guess the name of your NIS domain (anywhere on the net) can get a copy of your passwd file, and use "crack" and "John the Ripper" against your users' passwords. Also, it is possible to spoof NIS and do all sorts of nasty tricks. If you must use NIS, make sure you are aware of the dangers.

There is a much more secure replacement for NIS, called NIS+. Check out the NIS HOWTO for more information: http://metalab.unc.edu/mdw/HOWTO/NIS−HOWTO.html

8.10 Firewalls

Firewalls are a means of controlling what information is allowed into and out of your local network. Typically the firewall host is connected to the Internet and your local LAN, and the only access from your LAN to the Internet is through the firewall. This way the firewall can control what passes back and forth from the Internet and your LAN.

There are a number of types of firewalls and methods of setting them up. Linux machines make pretty good firewalls. Firewall code can be built right into 2.0 and higher kernels. The user−space tools `ipfwadm` for 2.0 kernels and `ipchains` for 2.2 kernels, allows you to change, on the fly, the types of network traffic you allow. You can also log particular types of network traffic.

Firewalls are a very useful and important technique in securing your network. However, never think that because you have a firewall, you don't need to secure the machines behind it. This is a fatal mistake. Check out the very good `Firewall−HOWTO` at your latest metalab archive for more information on firewalls and Linux. http://metalab.unc.edu/mdw/HOWTO/Firewall−HOWTO.html

More information can also be found in the IP−Masquerade mini−howto: http://metalab.unc.edu/mdw/HOWTO/mini/IP−Masquerade.html

More information on `ipfwadm` (the tool that lets you change settings on your firewall, can be found at it's home page: http://www.xos.nl/linux/ipfwadm/

If you have no experience with firewalls, and plan to set up one for more than just a simple security policy, the Firewalls book by O'Reilly and Associates or other online firewall document is mandatory reading. Check out http://www.ora.com for more information. The National Institute of Standards and Technology have put together an excellent document on firewalls. Although dated 1995, it is still quite good. You can find it at http://csrc.nist.gov/nistpubs/800−10/main.html. Also of interest:

- The Freefire Project −− a list of freely−available firewall tools, available at http://sites.inka.de/sites/lina/freefire−l/index_en.html
- SunWorld Firewall Design −− written by the authors of the O'Reilly book, this provides a rough introduction to the different firewall types. It's available at http://www.sunworld.com/swol−01−1996/swol−01−firewall.html
- Mason − the automated firewall builder for Linux. This is a firewall script that learns as you do the things you need to do on your network! More info at: http://www.pobox.com/~wstearns/mason/

8.11 IP Chains − Linux Kernel 2.2.x Firewalling

Linux IP Firewalling Chains is an update to the 2.0 Linux firewalling code for the 2.2 kernel. It has many more features than previous implementations, including:

- More flexible packet manipulations
- More complex accounting
- Simple policy changes possible atomically
- Fragments can be explicitly blocked, denied, etc.
- Logs suspicious packets.
- Can handle protocols other than ICMP/TCP/UDP.

If you are currently using `ipfwadm` on your 2.0 kernel, there are scripts available to convert the `ipfwadm` command format to the format `ipchains` uses.

Be sure to read the IP Chains HOWTO for further information. It is available at http://www.rustcorp.com/linux/ipchains/HOWTO.html

8.12 VPNs − Virtual Private Networks

VPN's are a way to establish a "virtual" network on top of some already−existing network. This virtual network often is encrypted and passes traffic only to and from some known entities that have joined the network. VPNs are often used to connect someone working at home over the public Internet to an internal company network.

If you are running a Linux masquerading firewall and need to pass MS PPTP (Microsoft's VPN point−to−point product) packets, there is a Linux kernel patch out to do just that. See: ip−masq−vpn.

There are several Linux VPN solutions available:

- vpnd. See the http://sunsite.auc.dk/vpnd/.
- Free S/Wan, available at http://www.xs4all.nl/~freeswan/
- ssh can be used to construct a VPN. See the VPN mini−howto for more information.
- vps (virtual private server) at http://www.strongcrypto.com.

See also the section on IPSEC for pointers and more information.

9.Security Preparation (before you go on−line)

Ok, so you have checked over your system, and determined it's as secure as feasible, and you're ready to put it online. There are a few things you should now do in order to prepare for an intrusion, so you can quickly disable the intruder, and get back up and running.

9.1 Make a Full Backup of Your Machine

Discussion of backup methods and storage is beyond the scope of this document, but here are a few words relating to backups and security:

If you have less than 650mb of data to store on a partition, a CD−R copy of your data is a good way to go (as it's hard to tamper with later, and if stored properly can last a long time). Tapes and other re−writable media should be write−protected as soon as your backup is complete, and then verified to prevent tampering. Make sure you store your backups in a secure off−line area. A good backup will ensure that you have a known good point to restore your system from.

9.2 Choosing a Good Backup Schedule

A six−tape cycle is easy to maintain. This includes four tapes for during the week, one tape for even Fridays, and one tape for odd Fridays. Perform an incremental backup every day, and a full backup on the appropriate Friday tape. If you make some particularly important changes or add some important data to your system, a full backup might well be in order.

9.3 Backup Your RPM or Debian File Database

In the event of an intrusion, you can use your RPM database like you would use `tripwire`, but only if you can be sure it too hasn't been modified. You should copy the RPM database to a floppy, and keep this copy off−line at all times. The Debian distribution likely has something similar.

The files `/var/lib/rpm/fileindex.rpm` and `/var/lib/rpm/packages.rpm` most likely won't fit on a single floppy. But if compressed, each should fit on a seperate floppy.

Now, when your system is compromised, you can use the command:

```
root#  rpm −Va
```

to verify each file on the system. See the rpm man page, as there are a few other options that can be included to make it less verbose. Keep in mind you must also be sure your RPM binary has not been compromised.

This means that every time a new RPM is added to the system, the RPM database will need to be rearchived. You will have to decide the advantages versus drawbacks.

9.4 Keep Track of Your System Accounting Data

It is very important that the information that comes from `syslog` not be compromised. Making the files in `/var/log` readable and writable by only a limited number of users is a good start.

Be sure to keep an eye on what gets written there, especially under the `auth` facility. Multiple login failures, for example, can indicate an attempted break−in.

Where to look for your log file will depend on your distribution. In a Linux system that conforms to the "Linux Filesystem Standard", such as Red Hat, you will want to look in `/var/log` and check `messages`, `mail.log`, and others.

You can find out where your distribution is logging to by looking at your `/etc/syslog.conf` file. This is the file that tells `syslogd` (the system logging daemon) where to log various messages.

You might also want to configure your log−rotating script or daemon to keep logs around longer so you have time to examine them. Take a look at the `logrotate` package on recent Red Hat distributions. Other distributions likely have a similar process.

If your log files have been tampered with, see if you can determine when the tampering started, and what sort of things appeared to be tampered with. Are there large periods of time that cannot be accounted for? Checking backup tapes (if you have any) for untampered log files is a good idea.

Intruders typically modify log files in order to cover their tracks, but they should still be checked for strange happenings. You may notice the intruder attempting to gain entrance, or exploit a program in order to obtain the root account. You might see log entries before the intruder has time to modify them.

You should also be sure to separate the `auth` facility from other log data, including attempts to switch users using `su`, login attempts, and other user accounting information.

If possible, configure `syslog` to send a copy of the most important data to a secure system. This will prevent an intruder from covering his tracks by deleting his login/su/ftp/etc attempts. See the `syslog.conf` man page, and refer to the `@` option.

There are several more advanced `syslogd` programs out there. Take a look at http://www.core−sdi.com/ssyslog/ for Secure Syslog. Secure Syslog allows you to encrypt your syslog entries and make sure no one has tampered with them.

Another `syslogd` with more features is syslog−ng. It allows you a lot more flexibility in your logging and also can has your remote syslog streams to prevent tampering.

Finally, log files are much less useful when no one is reading them. Take some time out every once in a while to look over your log files, and get a feeling for what they look like on a normal day. Knowing this can help make unusual things stand out.

9.5 Apply All New System Updates.

Most Linux users install from a CD−ROM. Due to the fast−paced nature of security fixes, new (fixed) programs are always being released. Before you connect your machine to the network, it's a good idea to check with your distribution's ftp site and get all the updated packages since you received your distribution CD−ROM. Many times these packages contain important security fixes, so it's a good idea to get them installed.

10.What To Do During and After a Breakin

So you have followed some of the advice here (or elsewhere) and have detected a break−in? The first thing to do is to remain calm. Hasty actions can cause more harm than the attacker would have.

10.1 Security Compromise Underway.

Spotting a security compromise under way can be a tense undertaking. How you react can have large consequences.

If the compromise you are seeing is a physical one, odds are you have spotted someone who has broken into your home, office or lab. You should notify your local authorities. In a lab, you might have spotted someone trying to open a case or reboot a machine. Depending on your authority and procedures, you might ask them to stop, or contact your local security people.

If you have detected a local user trying to compromise your security, the first thing to do is confirm they are in fact who you think they are. Check the site they are logging in from. Is it the site they normally log in from? No? Then use a non−electronic means of getting in touch. For instance, call them on the phone or walk over to their office/house and talk to them. If they agree that they are on, you can ask them to explain what they were doing or tell them to cease doing it. If they are not on, and have no idea what you are talking about, odds are this incident requires further investigation. Look into such incidents , and have lots of information before making any accusations.

If you have detected a network compromise, the first thing to do (if you are able) is to disconnect your network. If they are connected via modem, unplug the modem cable; if they are connected via Ethernet, unplug the Ethernet cable. This will prevent them from doing any further damage, and they will probably see it as a network problem rather than detection.

If you are unable to disconnect the network (if you have a busy site, or you do not have physical control of your machines), the next best step is to use something like `tcp_wrappers` or `ipfwadm` to deny access from the intruder's site.

If you can't deny all people from the same site as the intruder, locking the user's account will have to do. Note that locking an account is not an easy thing. You have to keep in mind `.rhosts` files, FTP access, and a host of possible backdoors.

After you have done one of the above (disconnected the network, denied access from their site, and/or disabled their account), you need to kill all their user processes and log them off.

You should monitor your site well for the next few minutes, as the attacker will try to get back in. Perhaps using a different account, and/or from a different network address.

10.2 Security Compromise has already happened

So you have either detected a compromise that has already happened or you have detected it and locked (hopefully) the offending attacker out of your system. Now what?

Closing the Hole

If you are able to determine what means the attacker used to get into your system, you should try to close that hole. For instance, perhaps you see several FTP entries just before the user logged in. Disable the FTP service and check and see if there is an updated version, or if any of the lists know of a fix.

Check all your log files, and make a visit to your security lists and pages and see if there are any new common exploits you can fix. You can find Caldera security fixes at http://www.caldera.com/tech−ref/security/. Red Hat has not yet separated their security fixes from bug fixes, but their distribution errata is available at http://www.redhat.com/errata

Debian now has a security mailing list and web page. See: http://www.debian.org/security/ for more information.

It is very likely that if one vendor has released a security update, that most other Linux vendors will as well.

There is now a Linux security auditing project. They are methodically going through all the user−space utilities and looking for possible security exploits and overflows. From their announcement:

> "We are attempting a systematic audit of Linux sources with a view to being as secure as OpenBSD. We have already uncovered (and fixed) some problems, but more help is welcome. The list is unmoderated and also a useful resource for general security discussions. The list address is: security−audit@ferret.lmh.ox.ac.uk To subscribe, send a mail to: security−audit−subscribe@ferret.lmh.ox.ac.uk"

If you don't lock the attacker out, they will likely be back. Not just back on your machine, but back somewhere on your network. If they were running a packet sniffer, odds are good they have access to other local machines.

Assessing the Damage

The first thing is to assess the damage. What has been compromised? If you are running an integrity checker like `Tripwire`, you can use it to perform an integrity check; it should help to tell you what has been compromised. If not, you will have to look around at all your important data.

Since Linux systems are getting easier and easier to install, you might consider saving your config files, wiping your disk(s), reinstalling, then restoring your user files and your config files from backups. This will ensure that you have a new, clean system. If you have to restore files from the compromised system, be especially cautious of any binaries that you restore, as they may be Trojan horses placed there by the intruder.

Re−installation should be considered mandatory upon an intruder obtaining root access. Additionally, you'd like to keep any evidence there is, so having a spare disk in the safe may make sense.

Then you have to worry about how long ago the compromise happened, and whether the backups hold any damaged work. More on backups later.

Backups, Backups, Backups!

Having regular backups is a godsend for security matters. If your system is compromised, you can restore the data you need from backups. Of course, some data is valuable to the attacker too, and they will not only destroy it, they will steal it and have their own copies; but at least you will still have the data.

You should check several backups back into the past before restoring a file that has been tampered with. The intruder could have compromised your files long ago, and you could have made many successful backups of the compromised file!

Of course, there are also a raft of security concerns with backups. Make sure you are storing them in a secure place. Know who has access to them. (If an attacker can get your backups, they can have access to all your data without you ever knowing it.)

Tracking Down the Intruder.

Ok, you have locked the intruder out, and recovered your system, but you're not quite done yet. While it is unlikely that most intruders will ever be caught, you should report the attack.

You should report the attack to the admin contact at the site from which the attacker attacked your system. You can look up this contact with `whois` or the Internic database. You might send them an email with all applicable log entries and dates and times. If you spotted anything else distinctive about your intruder, you might mention that too. After sending the email, you should (if you are so inclined) follow up with a phone call. If that admin in turn spots your attacker, they might be able to talk to the admin of the site where they are coming from and so on.

Good crackers often use many intermediate systems, some (or many) of which may not even know they have been compromised. Trying to track a cracker back to their home system can be difficult. Being polite to the admins you talk to can go a long way to getting help from them.

You should also notify any security organizations you are a part of (CERT or similar), as well as your Linux system vendor.

11.Security Sources

There are a LOT of good sites out there for Unix security in general and Linux security specifically. It's very important to subscribe to one (or more) of the security mailing lists and keep current on security fixes. Most of these lists are very low volume, and very informative.

11.1 FTP Sites

CERT is the Computer Emergency Response Team. They often send out alerts of current attacks and fixes. See ftp://ftp.cert.org for more information.

ZEDZ (formerly Replay) (http://www.zedz.net) has archives of many security programs. Since they are outside the US, they don't need to obey US crypto restrictions.

Matt Blaze is the author of CFS and a great security advocate. Matt's archive is available at ftp://ftp.research.att.com/pub/mab

`tue.nl` is a great security FTP site in the Netherlands. ftp.win.tue.nl

11.2 Web Sites

- The Hacker FAQ is a FAQ about hackers: The Hacker FAQ
- The COAST archive has a large number of Unix security programs and information: COAST
- SuSe Security Page: http://www.suse.de/security/
- Rootshell.com is a great site for seeing what exploits are currently being used by crackers: http://www.rootshell.com/
- BUGTRAQ puts out advisories on security issues: BUGTRAQ archives
- CERT, the Computer Emergency Response Team, puts out advisories on common attacks on Unix platforms: CERT home
- Dan Farmer is the author of SATAN and many other security tools. His home site has some interesting security survey information, as well as security tools: http://www.trouble.org
- The Linux security WWW is a good site for Linux security information: Linux Security WWW
- Infilsec has a vulnerability engine that can tell you what vulnerabilities affect a specific platform: http://www.infilsec.com/vulnerabilities/
- CIAC sends out periodic security bulletins on common exploits: http://ciac.llnl.gov/cgi−bin/index/bulletins
- A good starting point for Linux Pluggable Authentication modules can be found at http://www.kernel.org/pub/linux/libs/pam/.
- The Debian project has a web page for their security fixes and information. It is at

http://www.debian.com/security/.
- WWW Security FAQ, written by Lincoln Stein, is a great web security reference. Find it at http://www.w3.org/Security/Faq/www−security−faq.html

11.3 Mailing Lists

Bugtraq: To subscribe to bugtraq, send mail to listserv@netspace.org containing the message body subscribe bugtraq. (see links above for archives).

CIAC: Send e−mail to majordomo@tholia.llnl.gov. In the BODY (not subject) of the message put (either or both):

```
subscribe ciac-bulletin
```

Red Hat has a number of mailing lists, the most important of which is the redhat−announce list. You can read about security (and other) fixes as soon as they come out. Send email to redhat−announce−list−request@redhat.com with the Subject

```
Subscribe
```

See http://www.redhat.com/mailing−lists/redhat−announce−list/ for more info and archives.

The Debian project has a security mailing list that covers their security fixes. See http://www.debian.com/security/ for more information.

11.4 Books − Printed Reading Material

There are a number of good security books out there. This section lists a few of them. In addition to the security specific books, security is covered in a number of other books on system administration.

Building Internet Firewalls By D. Brent Chapman & Elizabeth D. Zwicky

1st Edition September 1995

ISBN: 1−56592−124−0

Practical UNIX & Internet Security, 2nd Edition By Simson Garfinkel & Gene Spafford

2nd Edition April 1996

ISBN: 1−56592−148−8

Computer Security Basics By Deborah Russell & G.T. Gangemi, Sr.

1st Edition July 1991

ISBN: 0−937175−71−4

Linux Network Administrator's Guide By Olaf Kirch

1st Edition January 1995

ISBN: 1−56592−087−2

PGP: Pretty Good Privacy By Simson Garfinkel

1st Edition December 1994

ISBN: 1−56592−098−8

Computer Crime A Crimefighter's Handbook By David Icove, Karl Seger & William VonStorch (Consulting Editor Eugene H. Spafford)

1st Edition August 1995

ISBN: 1−56592−086−4

Linux Security By John S. Flowers

New Riders;

ISBN: 0735700354

March 1999

Maximum Linux Security : A Hacker's Guide to Protecting Your Linux Server and Network

Anonymous

Paperback − 829 pages

Sams;

ISBN: 0672313413

July 1999

Intrusion Detection By Terry Escamilla

Paperback − 416 pages (September 1998)

John Wiley and Sons;

ISBN: 0471290009

Fighting Computer Crime

Donn Parker

Paperback − 526 pages (September 1998)

John Wiley and Sons;

ISBN: 0471163783

12.Glossary

* **authentication:** The process of knowing that the data received is the same as the data that was sent, and that the claimed sender is in fact the actual sender.
* **bastion Host:** A computer system that must be highly secured because it is vulnerable to attack, usually because it is exposed to the Internet and is a main point of contact for users of internal networks. It gets its name from the highly fortified projects on the outer walls of medieval castles. Bastions overlook critical areas of defense, usually having strong walls, room for extra troops, and the occasional useful tub of boiling hot oil for discouraging attackers.
* **buffer overflow:** Common coding style is to never allocate large enough buffers, and to not check for overflows. When such buffers overflow, the executing program (daemon or set−uid program) can be tricked in doing some other things. Generally this works by overwriting a function's return address on the stack to point to another location.
* **denial of service:** An attack that consumes the resources on your computer for things it was not intended to be doing, thus preventing normal use of your network resources for legitmate purposes.
* **dual−homed Host:** A general−purpose computer system that has at least two network interfaces.
* **firewall:** A component or set of components that restricts access between a protected network and the Internet, or between other sets of networks.
* **host:** A computer system attached to a network.
* **IP spoofing:** IP Spoofing is a complex technical attack that is made up of several components. It is a security exploit that works by tricking computers in a trust relationship into thinking that you are someone that you really aren't. There is an extensive paper written by daemon9, route, and infinity in the Volume Seven, Issue Forty−Eight issue of Phrack Magazine.
* **non−repudiation:** The property of a receiver being able to prove that the sender of some data did in fact send the data even though the sender might later deny ever having sent it.
* **packet:** The fundamental unit of communication on the Internet.

* **packet filtering:** The action a device takes to selectively control the flow of data to and from a network. Packet filters allow or block packets, usually while routing them from one network to another (most often from the Internet to an internal network, and vice−versa). To accomplish packet filtering, you set up rules that specify what types of packets (those to or from a particular IP address or port) are to be allowed and what types are to be blocked.
* **perimeter network:** A network added between a protected network and an external network, in order to provide an additional layer of security. A perimeter network is sometimes called a DMZ.
* **proxy server:** A program that deals with external servers on behalf of internal clients. Proxy clients talk to proxy servers, which relay approved client requests to real servers, and relay answers back to clients.
* **superuser:** An informal name for `root`.

13.Frequently Asked Questions

1. Is it more secure to compile driver support directly into the kernel, instead of making it a module?

   Answer: Some people think it is better to disable the ability to load device drivers using modules, because an intruder could load a Trojan module or a module that could affect system security.

   However, in order to load modules, you must be root. The module object files are also only writable by root. This means the intruder would need root access to insert a module. If the intruder gains root access, there are more serious things to worry about than whether he will load a module.

   Modules are for dynamically loading support for a particular device that may be infrequently used. On server machines, or firewalls for instance, this is very unlikely to happen. For this reason, it would make more sense to compile support directly into the kernel for machines acting as a server. Modules are also slower than support compiled directly in the kernel.
2. Why does logging in as root from a remote machine always fail?

   Answer: See Root Security. This is done intentionally to prevent remote users from attempting to connect via `telnet` to your machine as `root`, which is a serious security vulnerability, because then the root password would be transmitted, in cleartext, across the network. Don't forget: potential intruders have time on their side, and can run automated programs to find your password.
3. How do I enable shadow passwords on my Red Hat 4.2 or 5.x Linux box?

   Answer:

   To enable shadow passwords, run `pwconv` as root, and `/etc/shadow` should now exist, and be used by applications. If you are using RH 4.2 or above, the PAM modules will automatically adapt to the change from using normal `/etc/passwd` to shadow passwords without any other change.

Some background: shadow passwords is a mechanism for storing your password in a file other than the normal `/etc/passwd` file. This has several advantages. The first one is that the shadow file, `/etc/shadow`, is only readable by root, unlike `/etc/passwd`, which must remain readable by everyone. The other advantage is that as the administrator, you can enable or disable accounts without everyone knowing the status of other users' accounts.

The `/etc/passwd` file is then used to store user and group names, used by programs like `/bin/ls` to map the user ID to the proper username in a directory listing.

The `/etc/shadow` file then only contains the username and his/her password, and perhaps accounting information, like when the account expires, etc.

To enable shadow passwords, run `pwconv` as root, and `/etc/shadow` should now exist, and be used by applications. Since you are using RH 4.2 or above, the PAM modules will automatically adapt to the change from using normal `/etc/passwd` to shadow passwords without any other change.

Since you're interested in securing your passwords, perhaps you would also be interested in generating good passwords to begin with. For this you can use the `pam_cracklib` module, which is part of PAM. It runs your password against the Crack libraries to help you decide if it is too−easily guessable by password−cracking programs.

4. How can I enable the Apache SSL extensions?

   Answer:

   1. Get SSLeay 0.8.0 or later from ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL
   2. Build and test and install it!
   3. Get Apache 1.2.5 source
   4. Get Apache SSLeay extensions from here
   5. Unpack it in the apache−1.2.5 source directory and patch Apache as per the README.
   6. Configure and build it.

   You might also try ZEDZ net which has many pre−built packages, and is located outside of the United States.

5. How can I manipulate user accounts, and still retain security?

   Answer: The Red Hat distribution, especially RH5.0, contains a great number of tools to change the properties of user accounts.

   - The `pwconv` and `unpwconv` programs can be used to convert between shadow and non−shadowed passwords.

- ♦ The `pwck` and `grpck` programs can be used to verify proper organization of the `passwd` and `group` files.
- ♦ The `useradd`, `usermod`, and `userdel` programs can be used to add, delete and modify user accounts. The `groupadd`, `groupmod`, and `groupdel` programs will do the same for groups.
- ♦ Group passwords can be created using `gpasswd`.

All these programs are "shadow−aware" −− that is, if you enable shadow they will use `/etc/shadow` for password information, otherwise they won't.

See the respective man pages for further information.

6. How can I password−protect specific HTML documents using Apache?

I bet you didn't know about http://www.apacheweek.org, did you?

You can find information on user authentication at http://www.apacheweek.com/features/userauth as well as other web server security tips from http://www.apache.org/docs/misc/security_tips.html

---

14.Conclusion

By subscribing to the security alert mailing lists, and keeping current, you can do a lot towards securing your machine. If you pay attention to your log files and run something like `tripwire` regularly, you can do even more.

A reasonable level of computer security is not difficult to maintain on a home machine. More effort is required on business machines, but Linux can indeed be a secure platform. Due to the nature of Linux development, security fixes often come out much faster than they do on commercial operating systems, making Linux an ideal platform when security is a requirement.

15.Acknowledgements

Information here is collected from many sources. Thanks to the following that either indirectly or directly have contributed: following who either indirectly or directly have contributed:

`Rob Riggs rob@DevilsThumb.com`

S. Coffin scoffin@netcom.com

Viktor Przebinda viktor@CRYSTAL.MATH.ou.edu

Roelof Osinga roelof@eboa.com

Kyle Hasselbacher kyle@carefree.quux.soltc.net

David S. Jackson dsj@dsj.net

Todd G. Ruskell ruskell@boulder.nist.gov

Rogier Wolff R.E.Wolff@BitWizard.nl

Antonomasia ant@notatla.demon.co.uk

Nic Bellamy sky@wibble.net

Eric Hanchrow offby1@blarg.net

Robert J. Berger rberger@ibd.com

Ulrich Alpers lurchi@cdrom.uni−stuttgart.de

David Noha dave@c−c−s.com

Pavel Epifanov. epv@ibm.net

Joe Germuska. joe@germuska.com

Franklin S. Werren fswerren@bagpipes.net

Paul Rusty Russell <Paul.Russell@rustcorp.com.au>

Christine Gaunt <cgaunt@umich.edu>

lin bhewitt@refmntutl01.afsc.noaa.gov

A. Steinmetz astmail@yahoo.com

Jun Morimoto morimoto@xantia.citroen.org

Xiaotian Sun sunx@newton.me.berkeley.edu

Eric Hanchrow offby1@blarg.net

The following have translated this HOWTO into various other languages!

A special thank you to all of them for help spreading the Linux word...

Polish: Ziemek Borowski ziembor@FAQ−bot.ZiemBor.Waw.PL

Japanese: FUJIWARA Teruyoshi fjwr@mtj.biglobe.ne.jp

Indonesian: Tedi Heriyanto 22941219@students.ukdw.ac.id

Korean: Bume Chang Boxcar0001@aol.com

Spanish: Juan Carlos Fernandez piwiman@visionnetware.com

Dutch: R. Ekkebus reggy@zeelandnet.nl
