Linux Security Administrator's Guide

Dave Wreski, dave@nic.com

v0.98, 22 August 1998

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.

Contents

1 Introduction

This document covers the major security issues that affect Linux security. General philosophy and net born resources are also discussed.

A number of other HOWTO documents overlap with security issues, and those 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 point you where to look for such up to date information, and some general methods to prevent such exploits from taking place.

Additionally, while there are several resources available in various places on the Internet regarding general security, we are trying to consolidate much of this general information, and provide information a general system administrator can use as a practical guide. This should in no means substitute for reading books on the appropriate subject, and practical experience which works for you.

The US Government has several organizations devoted to computer security, and generally the information they have online is quite extensive, and very useful. A general introduction to computer security is available at *http://csrc.ncsl.nist.gov/nistpubs/800-12/* which will be very useful.

See the *References* section for pointers to security references. It is also a tremendous advantage if you understand how TCP/IP works, and some of the common system administration functions. You might find this guide helpful in a beginner introduction *http://www.sunworld.com/sunworldonline/swol-11-1995/swol-11-sysadmin.html* While it is Solaris-centric, you'll find much of this information general enough to still be applicable.

You may also find this link helpful *http://www.cis.ohio-state.edu/~dolske/gradwork/cis694q/* for another introduction to TCP, including how sequence numbers work, which is the foundation of "man in the middle" attacks, a description of the SYN/ACK handshake used to initiate a TCP connection, a description of a few of the problems in TCP/IP, a few other types of attacks, and how they work, as well as some solutions to these problems.

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 anonymous FTP sites who archive such information, including:

ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO

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

http://sunsite.unc.edu/LDP/

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

http://nic.com/~dave/Security/

1.2 Feedback

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

<*dave@nic.com*>

1.3 Disclaimer

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. Additionally, this is an early version, with many possibilities for inaccuracies and errors. It is provided "as is" without express or implied warranty.

Many of the examples and descriptions in this document refer specifically to the Red Hat distribution. We are very interested in incorporating other distributions as well. If you have ideas on how other distributions perform the same measures as are listed here, we would be interested in hearing from you.

1.4 Copyright Information

This document is copyrighted (c)1998 Dave Wreski, and distributed under the following terms:

- This document may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is allowed and encouraged; however, the authors would like to be notified of any such distributions.
- All translations, derivative works, or aggregate works incorporating any Linux documents must be covered under this copyright notice. That is, you may not produce a derivative work from this document and impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the author of this document for further information.

2 Overview

This document will discuss procedures and commonly used software to increase the trust level of your system. It is important to discuss the basic concepts first, and create a security foundation before we get started.

2.1 Organization of This Document

This document has been dividedinto a number of sections. They cover several broad kinds of security issues. So far these sections include:

- **Physical Security** - covers how you need to protect your physical machine from tampering.
- **Files and File System Security** - shows you how to setup your file-systems and permissions on your files.
- **Data Encryption, Cryptography and Authentication** - discusses how to use encryption to better secure your machine and network.
- **Kernel Security** - discusses what you can do at the kernel level to protect yourself, as well as improve security.
- **Network Security** - describes how to better secure your Linux system from network attacks.

- **Incident Control** - discusses the six stages in dealing with an incident, including the preperation before one actually occurs.
- **Host Security** - discusses what can be done to further secure individual hosts, and what to watch out for.
- **Exploits** - attempts to familiarize the reader with some of the most common types of exploits, so you know when and how to recognize one when it does happen.
- **Security Sources** - Here is a list of the resources that are most usable to a Linux Security Administrator.
- **Firewalls and Border Patrol** - discusses the various types of firewalls available for Linux, as well as pointers to general firewall information.
- **Glossary** - Here is a list of the most frequently used acronyms and definitions that a Security Administrator should be aware of to be effective.
- **Frequently Asked Questions** - These FAQs should help to reduce some of the more frequently encountered problems.

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, or otherwise improved the security of any suspect programs. Just doing this will help make your system markedly more secure.

2.2 Host Security

Perhaps the area of most concentration on security is done with 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 services your hosts offer, keeping good accounting records, and upgrading programs that have 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 of machines becomes larger. It can be said that host-based security does not scale. A host-based security exploit must be repaired on each machine on your network, which requires accessing each machine individually and applying the fix.

2.3 Network Security

Network security is as necessary as local host security. With your single system, or a distributed computing network, the Internet, or hundreds, if not thousands or more computers on the same network, you can't rely on each one of those systems being secure. Making sure authorized users are the only ones permitted to use your network resources, building firewalls, using strong encryption, and ensuring there are no rogue, or 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.

2.4 Security Through Obscurity

One type of security that must be discussed is "security through obscurity". This means that by doing something like changing the login name from 'root' to 'toor', for example, to try and obscure someone from breaking into your system as root may be thought of as a false sense of security, and can result in very unpleasant and unexpected consequences.

However, it can also be used to your benefit if done properly. If you tell all the users who are authorized to use the root account on your machines to use the root equivilent instead, entries in the `/var/log/secure` for the real root user would surely indicate an attempted break-in, giving you some advance notice. You'll have to decide if this advantage outweighs the additional administration overhead.

In most cases, though, any system attacker will quickly see through such empty security measures. Simply because you may have a small site, or relatively low profile does not mean an intruder won't be interested in what you have. We'll discuss what your protecting in the next sections.

2.5 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, your data. 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 still wondering what the difference is between a "Hacker" and a "Cracker", see Eric Raymond's document, "How to Become A Hacker", available at *http://sagan.earthspace.net/˜esr/faqs/hacker-howto.html*.

2.6 How Vulnerable Are We?

While it is difficult to determine just how vulnerable a particular system is, there are several indications we can use:

- The Computer Emergency Response Team consistently reports an increase in computer vulnerabilities and exploits.
- TCP and UDP, the protocols that comprise the Internet, were not written with security as their first priority when it was created more than 30 years ago.
- A version of software on one host has the same vulnerabilities as the same version of software on another host. Using this information, an intruder can exploit multiple systems using the same attack method.
- Many administrators don't even take simple security measures necessary to protect their site, or don't understand the ramifications of implementing some services. Many administrators are not given the additional time necessary to integrate the necessary security measures.

2.7 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. 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 is 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 makes it harder to surf the web.

If you have more than one person logging on to your machine, or machines, 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://ds.internic.net/rfc/rfc2196.txt*. It has been recently updated, and contains a great framework for establishing a security policy for your company.

It is even advisable to generate a security policy for systems with just two users, or even a desktop machine, used for normal Internet dialup access.

While developing your security policy, you will have to decide on that balance between security and ease-of-use. You will also need to determine the current level of security on your systems. Ask yourself questions such as these:

- How often do you change your passwords?
- How would *you* improve security?
- How many guessable passwords are there on your system?
- Do you have any intentional backdoors to your system?

Improving security at your site will have to be a progressive process – you can not secure your systems overnight, and most likely your users will be reluctant to change, because they feel they will be losing usability. Also, don't discount the possibility that there are several packages and binaries on your system that are not even used, and can be removed without affecting functionality, yet improving security by limiting the available exploits.

2.8 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, write files, or execute programs that could cause damage? Can they delete critical data? 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. A single user that is allowed to login using an `rhosts` file, or through the use of an insecure service, increases the ability for the intruder using this to "get his foot in the door". Once the intruder has even a normal user account on your system, or someone else's system, the likelihood it can be used to gain access to another system, or another account is quite high.

- **Threat** is typically from someone with motivation to gain unauthorized access to your network, or computer. You must decide who you trust to have access to your system, and what threat they could impose. There are several types of intruders, and it is useful to keep the 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 cause you time and money to recover.
  - **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.
- **Vulnerability** - describes how well protected your computer is from another network, and the potential for someone gaining unauthorized access. What's at stake if someone breaks into your system? How much is it worth? When making the evaluation, you should consider items such as computer hardware and software, intellectual property, employee's, resources, such as network bandwidth, disk space, etc.

  Of course the concerns of a dynamic PPP home user will be different than 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?

2.9 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 expressly 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 this permissions problem out, 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.

Additionally, there are several questions you will need to answer to successfully develop a security policy:

- What level of security do your users expect?
- How much is there to protect, and what is it worth?
- Can you afford the down-time of an intrusion?
- Should there be different levels of security for different groups?
- Do you trust your internal users?
- Have you found the balance between acceptable risk and secure?

You should develop a plan on who to contact when there is a security problem that needs attention.

There are quite a few documents available on developing a Site Security Policy. You can start with this one from Sun Microsystems *http://wwwwseast2.usec.sun.com/security/sec.policy.wp.html*

2.10 Means of Securing Your Site

This document will discuss various means in which you can secure the assets you have worked hard for: your local machine, data, users, network, even your reputation. What would happen to your reputation if an intruder deleted some of your user's 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 several reasons an intruder may be interested in your systems, which we will discuss later.

See the *Host Security* and *Network Security* sections for further information on steps to perform to secure your hosts.

2.11 Temporary Changes

Changes made for supposedly brief periods of time are also a great security risk. Subverting your firewall so you can dial-in from home to your workstation also allows an attacker to do the same. Also, temporary changes easily become permanent, as we quickly forget about such changes.

Remember, the weakest link in the security implementation is likely to be exploited first.

3 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 physical or local, 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.

3.1 Windows Networking

Most likely your network will also include Microsoft clients, presumably using either NetBIOS or other inheriently insecure networking protocols.

Among other things, NetBIOS is the protocol Microsoft uses to publicize share names, user names, and host names within the network.

Disabling NetBIOS on any Windows workstations is a prudent idea, as is blocking TCP and UDP ports 137 through 139 on your border routers or firewalls.

A detailed discussion on the actual reasons for this insecurity is available in a paper written by Hobbit, and can be found at his site here *http://avian.org:4687/web1/hak/cifs.txt*

Unfortunately, disabling NetBIOS also will disable any Remote Access Service it may be offering, as well as browsing (Network Neighborhood). If you must retain your NT server on your network, you may consider two NICs in the machine, one outbound via TCP/IP and one internal only. Disable NetBIOS binding to the TCP/IP side. This keeps enterprising folks from poking into the network via TCP/IP, then using various NET commands to gather network information.

The hacker group called l0pht have written a utility similar to how Crack works on UNIX, called *l0phtcrack* and is available at their site *http://www.l0pht.com* as is other generally useful information.

The file `security_level.txt`, distributed with SAMBA, discusses the various security levels that can be set using SAMBA, including encrypted passwords, server security, share-level security, and user security. It does a good job of explaining the general security concerns you must deal with.

The security research group called Rhino9, have also put together in depth information on the NetBIOS protocol and interface. You can find it at *http://207.98.195.250/texts/netbios.doc*

Internet Security Systems also produces a document on Windows file sharing security, and is available here *http://www.iss.net/vd/fileshare.html* This document, titled *File Sharing: Unknown Dangers on Your Network*, helps to describe some of the security issues you should be aware of, and just how insecure Windows 95 really is. It is a good overview, whereas Hobbit's document is more of a low-level description at the protocol level.

3.2 Identify Gateway Machines

Special attention should be paid to gateway or firewall systems, as they usually control access to the services running on the entire network. Such gateways should be identified, its function within the network shouild be assessed and owners or administrators should be identified. These hosts, often referred to as "bastion hosts" are a prime target for an intruder. They should be some of the most fortified machines on the network.

Be sure to regularly review the current access policies and security of the system itself.

These "systems" should absolutely only be running the services necessary to perform it's operation. Your firewall should not be your mail server, web server, contain user accounts, etc. Some of the things you should check for, and absolutely fortify on these hosts include:

- Turn off access to all but necessary services.
- Depending on the type of firewall, disable IP Forwarding, preventing the system from routing packets unless absolutely instructed to do so.
- Update machine by installing vendor patches immediately.
- Restrict network management utilities, such as SNMP, "public" communities, and write access.
- Be sure firewall policy includes mechanisms for preventing common attacks such as IP Spoofing, Fragmentation attacks, Denial of Service, etc.
- Monitor status very closely. You should develop a reference point in which the machine normally operates to be able to detect variations which may indicate an intrusion.
- Develop a comprehensive firewall model. Firewalls should be treated as a security system, not just a program that runs on a machine and has an access control list. Firewall administration should be centrally controlled and evaluation of firewall policies should be done prior to actual firewall deployment.

3.3 Network Monitoring

It is important to keep aware of the status of your network, so you can not only detect when there is an intrusion, but when there is abnormal system activity, such as system load, increased disk usage, slower network, etc. There are many tools available for network monitoring, most of which were developed on other platforms first, then ported to Linux.

See the COAST archives, available at *http://www.cs.purdue.edu/coast/hotlist/* for network monitoring tools.

Matthew Franz *mdfranz@txdirect.net* has put together a Linux distribution that runs on two or three floppies, and includes many of the tools necessary to probe a network and the services it has available. This sounds like a great method in which to test your current security policy, as well as find otherwise unknown vulnerabilities. You can find the latest version, as well as more information, at *http://www.txdirect.net/users/mdfranz/trinux.html*

3.4 Network Configuration Files

Improperly configured network services and configuration files can lead to a system lacking full control over its services. You can configure your systems to be secure, yet still offer the services necessary. As a general rule:

- Remove the `/etc/hosts.equiv` file. A properly configured system, using TCP Wrappers, offers much better control over which hosts and users are allowed access to the other machines on your network.
- Disable the use of `$HOME/.rhosts` files. By properly configuring PAM, you can eliminate the risk of a user subverting system security by allowing unauthorized access from a remote system via a `.rhosts` file. This should be replaced by the functionally equivilent SSH file called `.shosts`. If this is not possible, Wietse Venema wrote a more secure rsh and rlogin daemon replacement, available in the logdaemon package. You can find this at *ftp://ftp.win.tue.nl/pub/security/logdaemon-5.6.tar.gz*
- Verify the `/etc/exports` configuration. Be sure 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 access, and exporting read-only wherever possible. Verify who can mount these filesystems using */usr/sbin/showmount -e localhost*.
- Secure access to your console. Check the `/etc/securetty` file for the list of tty's that root is permitted to login from. This should only include the local tty's, and never including pseudo-ttys (from a remote location). The absense of this file indicates root is permitted to login from anywhere. Use */bin/su* or sudo, available at *ftp://ftp.cs.colorado.edu/pub/sudo/*
- Be sure to review your `/etc/inetd.conf` 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. All services running from inetd should be wrapped using TCP Wrappers.
- Disable all services such as the "r-utilities" including *exec* (used by *rsh*, *login* (used by *rlogin*), and *shell*, (used by *rcp*) should be disabled immediately from being started in `/etc/inetd.conf`. These protocols are extremely insecure and have been the cause of exploits in the past.
- Disable all unnecessary RPC services. Disable any non-essential services that are registered with the portmapper. RPC services are generally insecure, and have typically been replaced by newer forms of an equivilent service. Use *rpcinfo -p hostname* to find the list of RPC services running on *hostname*.

The best method of configuration here is to only enable the services in which the box is intended to serve. Network-based exploits are equally as common as other forms of exploits, and they are performed by finding weaknesses in services, or poorly configured services.

3.5 Check for Poor Topology Configuration

Poor network configurations can also lead to a very difficult intrusion to track. Protecting the "front door" with a very well configured firewall will not prevent someone from entering through the "back door" via the modem bank with poor authorization.

3.6 Disable Unnecessary and Unauthorized Services

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.

You should check your `/etc/rc.d/rcN.d`, where N is your systems run level and see if any of the servers started in that directory are not needed. The files in `/etc/rc.d/rcN.d` are actually symbolic links to the directory `/etc/rc.d/init.d`. Renaming the files in the `init.d` directory has the effect of disabling all the symbolic links in `/etc/rc.d/rcN.d`. If you only wish to disable a service for a particular runlevel, rename the appropriate file with a lower-case "s", instead of the upper-case "S", such as in S45dhcpd.

If you have BSD style rc files, you will want to check `/etc/rc*` for programs you don't need. The Red Hat distribution includes *tksysv*, a graphical program to change what runlevel a particular server runs in. The newer distributions also include *linuxconf*, which can also do this.

Additionally, machines on your network running unauthorized services can create an opportunity for a cracker to gain access to the system. Regular port scanning of your machines, as well as running network security scanning tools, can help to find these potential exploits before an intruder does.

3.7 Monitoring Network Services with TCP Wrappers

Most Linux distributions ship with tcp_wrappers "wrapping" all your TCP services. A tcp_wrapper (known as `/usr/sbin/tcpd`) is invoked from `/sbin/inetd` instead of the real service, such as *telnet* or *ftp*. *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 machines services.

By making simple changes to the inetd configuration file, `/etc/inetd.conf` you can monitor and control incoming requests to network services. Such a modification might look like the following:

Typical

```
telnet  stream   tcp   nowait root /usr/sbin/in.telnetd
```

TCP Wrappers

```
telnet  stream   tcp   nowait root /usr/sbin/tcp /usr/etc/in.telnetd
```

In default mode the wrappers report the name of the client host and of the requested service. Be sure you have `syslogd` configure properly to ensure correct logging.

As no information is exchanged between the wrappers and the client or server applications there is no overhead on the actual conversation between the client and server applications occurs.

Additionally, you can configure:

- Access control to restrict what systems can connect to what network daemons.

- Client user name lookups with the RFC 931 (ident) protocol.
- Additional protection against hosts that pretend to have someone else's host name.
- Additional protection against hosts that pretend to have someone else's host address.
- Notification upon usage of specific services, such as may be used to set trap doors for attempted intrusion.

3.8 Running Services in a `chroot` Environment

Several network services can now be configured to run in a restricted environment, called a "chroot jail". This is an isolated environment seperated from the "real" operating system. Services such as `Apache` or `bind` can be operated in this environment. A special root directory is created, with a complete installation of all programs and libraries necessary to execute the service. The intention is to prevent someone from obtaining root privilege on the "real" operating system, due to a bug in the service that is operating in the chroot jail.

This should not be treated as a panacea, however. It may help to restrict a process' filesystem access, but it doesn't affect its ability to make privileged system calls (e.g. init_module, modify_ldt, bind to a priviliged port, etc.) So ultimately a root process can break out of a chroot environment; it just makes the necessary shellcode more involved than just "exec("/bin/sh")". You can find more information on it's advantages and disadvantages at *http://www.ssc.com/lg/issue30/tag_chroot.html* This isn't explicitly a chroot discussion, but is helpful, nevertheless.

3.9 Domain Name Service (DNS) Security

Keeping up-to-date DNS information about all hosts on your network can help to increase security. In the event of 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.

Descriptive hostnames are just as useful to attackers as they are to internal users. Host names such as "firewall.mydomain.com" is obvious to an attacker, as is "ns.mydomain.com". These are likely to be prime targets to an attacker. A machine named "fred.mydomain.com" likely indicates a normal user's PC, which is also least likely to have an updated security mechanism installed, making it also a prime target.

Keep conscious of possible DNS spoofing. You can find more information on this in the *Exploits* section of this document.

Further information on securing DNS can be obtained from *http://www.psionic.com/papers/dns-linux.html*

Cricket Liu and Paul Albitz, the authors of the famed *DNS and BIND* O'Reilly book, contributed an article on *Sun World* with hints on how to secure DNS. You can find it, as well as some other excellent general security information at *http://www.sunworld.com/swol-11-1997/swol-11-bind.html* which discusses information on how to prevent being DNS spoofed.

Additonally, BIND can now successfully be run in a chroot() environment. John A. Martin *<jam@jamux.com>* has put together a set of Red Hat packages that can be used to install BIND in a chroot jail. You can find more information on this available at *ftp://ftp.tux.org/pub/tux/jam/*

Be sure to configure a separate user for BIND. This not only restricts the amount of damage an intruder can do after exploiting a security hole in BIND, but also allows administration of the zone files without having to be root. This is generally a good practice, and more packages are configured for doing this more easily than before possible.

3.10 Network File System (NFS) Security

NFS is a very widely used file sharing protocol. It allows servers running `nfsd(8)` and `mountd(8)` to "export" entire filesystems to other machines with nfs filesystem support built-in to their kernels (or some other client support if they are non Linux machines). `mountd(8)` keeps track of mounted filesystems in `/etc/mtab`, and can display them with showmount(8).

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 superuser 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 export only. Never export your entire root directory, export only directories you need to export and export read-only wherever possible.

Filter TCP port 111, UDP port 111 (portmapper), TCP port 2049, and UDP port 2049 (nfsd) on your firewall or gateway to prevent external access.

The NFS HOWTO also discusses some of the security issues with NFS, and it is available at *http://sunsite.unc.edu/LDP/HOWTO/NFS-HOWTO.html* for more information on NFS.

3.11 Network Information Service (NIS)

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), among other information.

NIS is not at all secure. It was never meant to be. It was meant to be handy and useful. Not only was it not intended to be secure, it also has characteristics which inherently make it insecure. Among these are:

- Lack of access control for contents of NIS maps
- Negation of password shadowing
- Rogue servers acting as authentic ones

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 against your users passwords.

If you must use NIS, make sure you are aware of the dangers.

Control the use of `/etc/netgroup` file for NIS systems. Explicitly define which hosts and which users can connect from a known list of machines.

There is a much more secure replacement for NIS, called NIS+. Check out the NIS HOWTO for more information, available at *http://sunsite.unc.edu/LDP/HOWTO/NIS-HOWTO.html*

3.12 File Transport Protocol (FTP)

The Washington University FTP server is the default server on Linux distributions. It has the ability to run in a *chroot* environment, thus (theoretically) protecting the real root environment, limiting the damage an exploit can do.

FTP sites are easily misconfigured, and doing so can lead to a false sense of security, as well as easily exploitable holes. Attackers can use a misconfigured site to transfer pirate software, gain remote access, corrupt downloadable files, cause a denial of service, among other misuses.

Be sure to disable FTP entirely if you don't have any reason to leave it enabled (such as replacing it with ssh) and definately enable quotas on the FTP filesystem. Additionally, disable anonymous FTP access if it is not necessary.

3.13 Simple Mail Transport Protocol (SMTP)

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. *http://www.sendmail.org*

An alternative to sendmail is qmail, which alledges to be more secure, and easier to configure. qmail was designed with security in mind from the ground up. It's reported that it's fast, stable and secure. You can find it at *http://www.qmail.org*

Wietse Venema <*wietse@wzv.win.tue.nl*> is writing a mail server that is still in testing stages, but also promotes improved security. You can find out more about vmail at *http://www.vmailer.org*

Significant improvements in preventing unsolicited bulk email (spam) have been made with recent versions of the available SMTP servers. Starting with sendmail-8.9, anti-relaying is enabled by default, which prevents a remote host from using your network and mail servers for forwarding mail to other hosts. Additional filters are also available for preventing spam.

4 Host 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 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 have no contact information for is a very bad idea.

4.1 Delete Unnecessary Packages

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

If you are configuring a new machine to be installed on the network, only initially install the packages that are necessary for its normal operation.

Removing unnecessary setuid and setgid binaries should be a priority. You should always be aware of which ones are available on your system. You can do this using the following:

```
user@myhost$ find / -type f -perm +6000
```

This will find all the setuid and setgid binaries on your system. You can find more about the setuid and setgid permissions in the *File System Security* section.

4.2 Default System Configuration

The default Linux system installation is generally far more secure than other operating systems, due to not having to conform to older standards and traditions.

However, installing any operating system, and connecting it to the network is a foolish idea. Many system defaults are still more lenient than is intended to be used in a production network system.

Spend some time to customize it to your environment. Be sure to follow these guidelines, as well as the ones refered to herein, including disabling any services that are not necessary, configuring auditing, etc, before connecting a machine to the network.

4.3 Make a Full Backup of Your Machine

Discussion of backup methods and storage is beyond the scope of this document, but 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 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.

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

4.4 Backup Your Red Hat 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 and */bin/rpm* executable to a floppy or Zip disk, and keep this copy off-line at all times. The Debian distribution likely has something similar. (Would someone fill me in here, until I get Debian re-installed?) See the section on Integrity Checking for further information, and instructions on how to do this.

4.5 Make Use of Your System Accounting Data

It is very important that the information that comes from your system accounting files has not been compromised, and is installed and configured properly. Making the files in `/var/log`, `/var/run/utmp`, and `/var/log/wtmp` readable, and writable only by the root user is a good start. Knowing which tools to use at what times is a good practice.

You can find more information on this in the *User and System Accounting* section.

4.6 Apply All New System Updates

Most Linux users install from a CDROM. 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 (ftp.redhat.com for example) and get all the updated packages since you received your distribution CDROM. Many times these packages contain important security fixes, so it's a good idea to get them installed.

4.7 Creating New Accounts

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

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

- Limit access privileges given to new users.
- Be aware when/where they login from, or should be logging in from.
- Make sure to remove inactive accounts
- The use of the same user-ID on all computers and networks is advisable to ease account maintenance, as well as permit easier analysis of log data (but I'm sure someone will dispute this). However, it's practically essential if using NFS. There are several other protocols that use UIDs for local and remote access as well.
- The creation of group user-IDs should be absolutely prohibited. User accounts also provide accountability, and this is not possible with group accounts.
- Be sure shadow passwords are enabled. See the *Password Security* section for more information.
- Regularly audit user accounts for invalid or unused accounts, expired accounts, etc.
- Check for repeated login failures
- Be sure to enable quotas, to prevent denial of service attacks involving filling disk partitions, or appending exploits to group-writable files.
- Disable group accounts, and unused system accounts, such as `sys` or `uucp`. These accounts should be locked, and given non-functional shells.

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

4.8 Root Security

The most sought-after account on your machine is the 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. Running as root all the time is a very very very bad idea.

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., you are going to do a `rm foo*.bak`, instead, 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 `/bin/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, or the PATH environment variable, defines the location the shell searches for programs. Try and limit the command path for the root user as much as possible, and never use '.', meaning 'the current directory', in your PATH statement. 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* (called the "r-utilities") suite of tools as root. They are subject to many sorts of attacks, and are downright dangerous 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 careful of adding anything else to this file. You should be able to login remotely as your regular user account and then use *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 superuser 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. *sudo* 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, but use *sudo* so you can 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 the user root access. 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 yet be secure.

4.9 Workstations and DialUp Security

User of computers to connect to the Internet via a dial-up line, or workstations that otherwise offer no services to external hosts can also improve their security with relatively easy modifications to the stock Linux installation.

If there is never have a need to connect to your machine from another one on the network, the quickest solution is to simply disable `/usr/sbin/inetd` from even being started. This is the master Internet daemon, which controls some normal server services, such as `telnet`, `ftp`, etc. If you retrieve your mail from a remote host, and your Internet Service Provider is hosting your web page, then most likely there is not a need to enable these services.

On stock Red Hat systems, the file `/etc/rc.d/rc3.d/S50inet` controls the starting and stopping of the `inetd` server. Simply rename the `S50inet` file to `s50inet` to disable it, or see your Red Hat administration manual for further information.

Alternatively, if you are a home dialup user, it is also possible to deny all incoming connections using TCP Wrappers. TCP Wrappers, `/usr/sbin/tcpd`, also logs failed attempts to access services, so this can give you an idea that you are under attack. If you add new services, you should be sure to configure it to use tcp_wrappers TCP based. For example, a normal dial-up user can prevent outsiders from connecting to your 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.
```

(including the ending period) 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. TCP Wrappers can be combined with several other services, such as `sendmail` and `sshd` to give even further control over access. See the respective documentation for further information.

4.10 X11, SVGA and display security

4.10.1 X11

It's important for you to secure your graphical display to prevent attackers from doing things such as grabbing your passwords as you type them without you knowing it, reading documents or information you are reading on your screen, or even using a hole to gain superuser 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 can use xhost to specify what hosts are allowed access to your display. This is not very secure at all. If someone has access to your machine they can xhost + their machine and get in easily.

When using xdm (X Display Manager) to login, you get a much better access method: MIT-MAGIC-COOKIE-1. A 128bit cookie is generated and stored in your .Xauthority file. These cookies need to be transferred in confidence, and you really don't gain anything if your home directory is shared via NFS. 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 only that connection access. See the Remote-X-Apps mini-howto, available at *http://sunsite.unc.edu/LDP/HOWTO/mini/Remote-X-Apps.html*.

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

Take a look at the `Xsecurity(1)` man page for more information on X security. The safe bet is to use `xdm(1)` to login to your console and then use ssh to go to remote sites you wish to run X programs off of.

4.10.2 SVGA

SVGAlib programs are typically `setuid-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.

4.10.3 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/*

4.11 identd

`identd` is a small program that typically runs out of your `inetd`. 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 they 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` has not been compromised, then you know it is telling remote sites the user-name or user-ID of people using TCP services. If the admin at a remote site comes back to you and tells you a user was trying to hack into their site, you can easily take action against that user at your site who is misusing a service. 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 `identd` for specific users (they can make a `.noident` file), you can log all `identd` requests, which is recommended, and `identd` can return a uid instead of a user name or even NO-USER. Keep in mind it is really only useful is on a network where nobody hostile has root access. Then it can help in catching mail forgeries, for instance.

5 User, System, and Process Accounting

All Linux systems support system-wide process, user, and system accounting, and it is wise to take advantage of it. You will need this information when troubleshooting a possible security incident, and your ability to address all aspects of a specific incident strongly depends on the success of this analysis.

There are quite a few things, as the Security Administrator, of which you should be aware. These include at least the following:

- Login activity
- Authorization information
- Authentication information
- Commands users have run
- Restarts and shutdowns of the system
- Network transactions records
-

5.1 Using Syslog

The system daemon called `syslog` is the program used to log system events such as kernel messages, login or logout messages, general system messages, etc.

Be sure to keep an eye on its normal operation and what gets written to it's log files, especially under the "auth" facility. Multiple login failures, for example, can indicate an attempted break-in. Keep in mind that the lack of information does not indicate the opposite.

Where to look for your log file will depend on your distribution. In a Linux system that conforms to the "Linux File-system 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 `/usr/sbin/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 in recent Red Hat distributions. Other distributions likely have a similar process. It seems that many distributions default to only logging the most basic information, so you should spend some time and customize it for your environment.

If your log files have been tampered with, see if you can determine when the tampering started, and what sort of things they appeared to tamper 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.

Log files are typically modified by the intruder in order to cover his 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 seperate the "authpriv" facility from other log data, including attempts to switch users using `/bin/su`, login attempts, and other user accounting information.

5.1.1 Storing Log Data Securely

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. It most likely of little value in a criminal investigation. It helps if the log data, which has been stored remotely, indicates when root access was gained so that logs before that point are okay.

The `syslogd` daemon can be configured to automatically send log data to a central `syslogd` server, but this is typically sent in cleartext data, 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 has been reported, 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. A more secure implementation has been written by CORE-SDI, and is available at *http://www.core-sdi.com/ENGLISH/CoreLabs/ssyslog/index.html*

If possible, configure `syslogd` 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(5)` man page, and refer to the "@" option.

If you've already decided to use a central syslog server, the additional security this provides is well worth it. However, you should consider the additional overhead involved with sending this data real-time across your network.

5.2 Using User Accounting

User accounting can be used to discover information about who is currently using the system. While you cannot necessarily verify the integrity of this information once your machine has been exploited, it can be a useful tool to track the systems a particular user has logged into, what time he or she logged in, when the system was last rebooted, etc.

There are also utilities available for locking There are several tools available to process this information, including `last(1)`, `who(1)`, `ac(1)`, `utmpdump(1)` (typically for debugging only), among others.

For example, using the `/usr/bin/last` command, you can view quite a bit of information about your system:

```
root     tty1                              Fri Jul  3 21:02   still logged in
reboot   system boot                       Fri Jul  3 21:01
dave     ttyp2        localhost            Wed Jul  1 23:11 - 23:11  (00:00)
david    ttyp2        localhost            Wed Jul  1 22:47 - 22:47  (00:00)
```

The `last(1)` command, which shows a listing of last logged in users, and `lastb(1)`, which shows a listing of failed login attempts (assuming `/var/log/btmp` exists), both consult the `/var/log/wtmp` file, which contains the following information:

- Type of Login
- Process ID of login process
- Device name of tty
- Init ID or abbreviated ttyname
- User Name
- Hostname for remote login
- Exit Status of a process
- Time entry was made
- IP address of remote host

See the man page for `wtmp(5)` for a description of any of the fields you do not understand.

The file `/var/run/utmp` is the file that is consulted to find out who is currently on the system (and primarily used by the `who(1)` command). However, there may be more users currently using the system because not all programs use utmp logging. This file is typically truncated upon each system boot, by one of the `/etc/rc.d/rc.*` files. Be sure this file is not writable by users other than root, as it is possible to insert or delete entries from this file otherwise. This file really serves very little purpose.

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 (especially when you suspect an unauthorized visitor), and get a feeling for what the look like on a normal day. Knowing this can help make unusual things stand out.

5.3 Using Process Accounting

Process accounting support has also been integrated into the new kernels. To use this feature, you'll need to get *ftp://sunsite.unc.edu:/pub/Linux/system/admin/accounts/acct-1.3.73.tar.gz*

It no longer requires patching the kernel for this ability. This package includes several program to manage the kernel-level functions, including:

- accton (8) - Turn on accounting of processes
- accttrim (8) - Trim down the size of an accounting file
- lastcomm (1) - show last commands executed in reverse order

It really works quite well, and is highly recommended for systems that have a large number of users.

5.4 Managing User Accounts

Having control over the resources and data your users have access to is an essential part of maintaining security. Linux provides a large number of tools including account permissions, passwords, account aging, adding and deleting of users, etc.

Some of the programs you should become familiar with to manage users and groups include:

- chage (1) - change user password expiry information
- groups (1) - print the groups a user is in
- newusers (8) - update and create new users in batch
- passwd (1) - update a user's authentication tokens(s)
- nologin (5) - prevent non-root users from log into the system
- su (1) - run a shell with substitute user and group IDs
- useradd (8) - Create a new user or update default new user information
- userdel (8) - Delete a user account and related files
- usermod (8) - Modify a user account
- chgrp (1) - change the group ownership of files
- chown (1) - change the user and group ownership of files
- gpasswd (1) - administer the `/etc/group` file
- groupadd (8) - Create a new group
- groupdel (8) - Delete a group
- groupmod (8) - Modify a group
- groups (1) - print the groups a user is in
- grpck (8) - verify integrity of group files
- pwconv (8) - convert to and from shadow passwords
- pwunconv (8) - convert to and from shadow passwords
- grpconv (8) - convert to and from shadow passwords
- grpunconv (8)- convert to and from shadow passwords
- vipw (8) - edit the password or group files
- vigr (8) - edit the password or group files

You can read the online manual pages for these commands using a syntax similiar to the following:

```
user@myhost# man 8 pwunconv
```

This refers to `pwunconv` in section 8 of the manual pages.

You can find additional account management packages at *ftp://sunsite.unc.edu:/pub/Linux/system/admin/accounts*

6 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 environment, 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 a 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 a good idea, but beyond the scope of this document. :)

Make use of `/etc/shutdown.allow` to prevent someone from rebooting your machine. This file is consulted when the machine is rebooted using the Control-Alt-Del keys. It contains a list of usernames that are authorized to reboot the machine.

6.1 Computer Locks

Many more 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 on 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 make it so that it will not let you plug in new keyboards and 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 cases (most notably SPARC and Mac) 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.

6.2 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.

Under Linux/x86 many PC BIOSs let you set a boot password. This doesn't provide all that much security (BIOS can be reset, or removed if someone can get into the case), but might be a good deterrent (e.g., it will take time and leave traces of tampering).

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. Some examples are: disallow booting from floppy drives and passwords to access some BIOS features.

On Linux/SPARC, your SPARC EEPROM can be set to require a boot-up password. This might slow attackers down.

NOTE: If you have a server machine, and you setup 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.

6.3 Boot Loader Security

The various Linux boot loaders also can have a boot password set. Using LILO, take a look at the "restricted" and "password" settings. "password" allows you to set a boot-up password. "restricted" will let the machine boot _unless_ someone specifies options at the LILO: prompt (like "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. This 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, as well as password-protecting your computer's BIOS.

If anyone has security related information from a different boot loader, we would love to hear it. (SILO, MILO, loadlin, etc).

NOTE: If you have a server machine, and you setup 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. ;(

6.4 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 tampers with or looks 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 TTY 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 does not 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. At the very least, start X in the background, and log out of the console.

7 Intrusion Detection

Intruders are constantly attempting different mechanisms to attack your system. You must be able to detect these varying attempts, and know what to do when they happen. You should also be able to distinguish the normal operating conditions from an actual attack.

You must be able to determine things like whether or not there really was an intrusion, to what extent the attack occured.

7.1 What is Intrusion Detection?

Intrusion Detection is the method in which a security administrator uses to detect the presence of an unauthorized intruder. An *Intrusion Detection System (IDS)* are the combination of tools that a security administrator uses to detect the intrusion. Briefly, the available types of intrusion detection include:

- **Network Based Intrusion Detection** - These mechanisms typically consist of a black box that sits on the network in promiscious mode, listening for patterns indictive of an intrusion.
- **Host Based Intrusion Detection** - These mechanisms typically include auditing for specific events that occur on a specific host. These are not as common, due to the overhead they incur by having to monitor each system event.
- **Log File Monitoring** - These mechanisms are typically programs that parse log files after an event has already occured, such as failed login attempts, etc.
- **File Integrity Checking** - These mechanisms typically check for trojan horses, or files that have otherwise been modified, indicating an intruder has already been there. The Red Hat Package Manager, RPM, has this capability, as does the well-known *Tripwire* package.

7.2 General Indications of Intrusion

Being capable of detecting an intrusion is as important as being able to stop it once it happens. It is important that you are able to detect the subtle signs left by an intruder during his attack of your system.

Suspicious signs of intrusion include at least the following:

7.2.1 User Indications

- Failed log-in attempts
- Log-ins to accounts that have not been used for an extended period of time
- Log-ins during hours other than non-working hours
- The presence of new user accounts that were not created by the system administrator
- su entries or logins from strange places, as well as repeated failed attempts

7.2.2 System Indications

- Modifications to system software and configuration files
- Gaps in system accounting that indicate that no activity has occurred for a long period of time
- Unusually slow system performance
- System crashes or reboots
- Short or incomplete logs
- Logs containing strange timestamps
- Logs with incorrect permissions or ownership
- Missing logs
- Abnormal system performance
- Unfamiliar processes
- Unusual graphic displays or text messages.

7.2.3 File System Indications

- The presence of new, unfamiliar files or programs
- Changes in file permissions
- Unexplained changes in file size. Be sure to analyize all your system files, including those in your `$HOME/` directory such as `$HOME/.bashrc` for modified `$PATH` entries, as well as changes in system configuration files in `/etc`
- Rogue suid and sgid files on the system that do not correspond to your master list of suid and sgid files
- Unfamiliar file names in directories
- Missing files

7.2.4 Network Indications

- Repeated probes of the available services on your machines
- Connections from unusual locations
- Repeated login attempts from remote hosts
- Arbitrary log data in log files, indicating attempt at creating either Denial of Service, or crash service

7.3 General Methods for Detecting Intrusions

In order to determine if an intruder has violated your system, you must be familiar with the normal system administration tools, and be able to use them to find the "footprint" a cracker may have left behind. This procedure can be relatively easy, or practically impossible, depending on how much preparation you have done, as well as the stage you've detected the intruder, and how skilled the intruder is.

There are pointers throughout this document that list the various tools available. Some of the tools and methods you should become familiar with include:

- Log file analysis. Be sure to see the *User Security* section for information on `syslog(8)` which is responsible for logging many system events that are helpful in tracking connections to your system, as well as local system events.
- Become familiar with the `last(1)`, `lastcomm(1)`, and `netstat(8)` commands. These are available to show valuable information about the users, commands, and connections on your system. More information on these commands are available in the *User Security* section.
- Look for signs of physical intrusion.
- Ensure that the software you are using to search for the intruder hasn't itself been compromised. Do not place all your trust in the tools you are using, and the output they produce. Consider placing a set of secure binaries on external media that can be used later, with confidence. See the *http://www.txdirect.net/users/mdfranz/trinux.html* package for a starting point.
- Follow the guidelines provided by CERT in this document *ftp://info.cert.org/pub/tech_tips/UNIX_configuration_guide*
- Check other local systems that may have been used to attack at yours
- Check for systems at remote sites that may be involved or affected
- Investigate unauthorized hardware attached to the network
- Observe your systems for anything unusual, and certainly investigate anything you find
- Notify your incident response team if you find something that could have been performed by an unauthorized user
- Use the network monitoring tools. There are also several nifty network monitoring tools there that are also very helpful. It is important to keep aware of the status of your network, so you know when to be alerted to a specific event. See the *Network Monitoring* section for more information.

7.4 Intrusion Detection Tools

There are many intrusion detection tools available for Linux, and many new tools are constantly becomming available. While the majority of the tools are host-based intrusion detection tools, there are a number of network-based tools as well.

7.4.1 Host Based Detection Tools

- Tripwire
- Make use of the available tools. There are several tools available to detect when someone is portscanning your network. Start with *http://www.psionic.com/abacus/abacus_sentry.html* which is the Sentry intrusion detection tool.

There are also several intrusion detection tools available at *http://www.eng.auburn.edu/users/doug/second.html* including a tool called *klaxton* which basically sets a trap for an intruder, then notifies you when some is "doorknob rattling".

7.5 Integrity Checking

A very good way to determine if you have an unwanted visitor is to check your local files for possible trojan horses, missing files, files that are larger or smaller than they are supposed to be, etc.

Fortunately, there are several tools that can verify the file integrity. Many Linux distributions use RPM for their package management, which inherently has integrity checking. Also available is the well-known program called `tripwire`.

7.6 Using `tripwire`

Tripwire runs 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 tripwire onto a floppy, and then physically set the write protect on the floppy. This way intruders can't tamper with tripwire itself or change the database. Once you have tripwire 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 tripwire 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.

Tripwire 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, which is a solid reason to keep track of the status of the binaries on your system.

A company called *Visual Computing Corporation* now apparently has been given exclusive rights to continue development of tripwire, originally developed at Purdue University. It looks to be so-far-so-good, as there is still a working version for Linux. You can find more information from them at *http://www.visualcomputing.com*

7.7 Using The Red Hat Package Mangaer

The Red Hat Package Manager (RPM) program includes the ability to verify all packages that it has installed on the system.

RPM has facilities for verifying that a package is not corrupt or has components missing. A program added or removed by a cracker will not match the original and RPM will generally report a verification failure.

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. RPM can also be combined with PGP to check a package's signature. Typical output might look like the following:

```
..5....T /bin/login
```

should sound alarm bells. RPM produces the following useful output fields:

- S - file size changed
- M - file mode changed
- 5 - MD5 checksum failed
- U - file owner changed
- G - group changed

This means that every time a new RPM is added to the system, the RPM database will need to be re-archived. You will have to decide the advantages versus drawbacks. Also, keep in mind that it won't verify programs that RPM did not install.

Specifically, the files `/var/lib/rpm/fileindex.rpm` and `/var/lib/rpm/packages.rpm` most likely won't fit on a single floppy. Compressed, each should fit on a separate floppy. Consider storing this (as well as the actual `/bin/rpm` executable!!) on a Zip cartrige.

7.8 File System Guidelines

Intruders often either modify, delete, or replace existing files in order to either cover their tracks, assist them in gaining access, or to gather further information.

Ensuring the integrity of the files and programs on your system is vital in intrusion detection. Several means can be used to determine if files have been tampered with on your system:

- Look for suspicious files on your system, or even system files that may have been tampered with, or missing. You can find the list of the most recently modified files with the following command:

  ```
  user@host# /usr/bin/find / -ctime -1 -print
  ```

  Read the *File System Security* section for tips on scanning your filesystem for changed files, as well as setuid and sgid files.

- Verify the integrity of the files. If you are prepared, you can use your Red Hat RPM database, or Tripwire database stored on external media at this time to verify the integrity of the most important files on your system.

7.9 Physical Intrusion Detection

Intruders may attempt to breach your network's by physical infitration as well as via the network. Keep in mind that one system can be used to penetrate many others, so securing one machine is as important as securing another.

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. You should always investigate machine reboots.

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.

7.10 Packet Sniffers

One of the more common ways intruders gain access to multiple systems on your network is by employing a packet sniffer on a already compromised host. This software-based "sniffer" just listens on the Ethernet port for things like "password" 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.

An attacker doesn't even need to compromise a system to do this, they could also bring a laptop or PC into your building and tap into your net.

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

If you are using `syslog` to send your data to a central log server, consider that the data is sent in clear text, and much information can be gathered from this data. Consider using a secure implementation of syslog, which encrypts and compresses the data before it is sent. See the *Using Syslog* section for more information on configuring `syslogd(8)` securely.

8 Files and File System Security

A few minutes of preparation and planning ahead before putting your systems online can help to protect your system, and the data that is stored on it.

This section discusses some of the methods in which you can use to secure the files on your system, some general guidelines for improving the overall security of the files on your system, and some ideas for preventing problems from occuring in the first place. It also discusses the commands to use to modify the permissions and ownership of files and directories on your system.

Before we discuss some of these methods of improving file system security, it is important to have an understanding of basic Linux file security, ownership, and what each of the fields from a file listing actually mean.

To display the ownership and permissions of a file on your system, use the *long-listing* option, as well as the *display all files* option to the `ls(1)` command. A typical `/bin/ls -la` command might show the following, with the first line being a field marker:

```
    |----1----|-2--|---3----|----4-----|---5--|-----6------|---7-----|
1.  drwxrwxr-x  24 root     users        1024 Aug 19 00:05 .
2.  drwxr-xr-x  22 root     root         1024 Aug 11 22:04 ..
3.  drwxr-xr-x   3 root     root         1024 Jun 19 03:40 Mail
4.  -rw-rw-r--   1 dave     security    43244 Jul 20 14:11 README
5.  drwxrwsr-x  17 dave     security     1024 Jul 31 01:48 Security
        [More not shown]
```

Each of these fields provide useful information to the security administrator. First, a description of each field (as shown from left to right), then a more in-depth explanation of the most important ones. The numbers down the left side represent the line numbers, which will be referred to later.

- **Field One:** Permissions for this file or directory. The first nine positions from the right describe the *user*, *group*, and *other* permissions, in groups of three. Within each group of three, the first character denotes read access, the second denotes write access, and the last denotes execute, working from left to right. The tenth position describes the type of file, which can be either a regular file, directory, FIFO, symbolic link, or other type of special file.
- **Field Two:** Number of hard links to this file or directory. These links can be directories, for example. In this case the current directory (line 1) most likely has 24 directories below it, of which only two are shown here (`Security` and `Mail`)
- **Field Three:** Owner of the file or directory. This field is as important as the permissions themselves.
- **Field Four:** The group to which the file belongs. This field, in conjunction with the owner field (field three) are necessary in order to set the permissions correctly.
- **Field Five:** Size of file
- **Field Six:** Modification time
- **Field Seven:** File name

8.1 File Permissions and Ownership

Continuing where we left off in the previous section, we can now discuss some of the fields described above. Particularly, field one and fields three and four are the most exiciting.

Linux 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.

The files within each of these categories have specific permissions with which they are accessed. File permissions, including regular files, special files (such as FIFOs, sockets, etc), or symbolic links (which dereference the permissions to the file they point to) can have any one, or any, of the following:

```
Symbol        Permission      Description
------------------------------------------------------------------
r             Read            Can be opened to read the contents
w             Write           Can be modified, including appending
                              and deleting
x             Execute         Can execute the file if it is a
                              program or shell script
s             Special Perm    setuid or setgid permission
-             Access Denied   Cannot be read, written, or executed,
                              depending on the position of the '-'
```

The read, write, and execute permissions should be pretty clear as to their meaning. However, the "s" symbol may need to explanation. The next two sections address this symbol.

8.1.1 Set User Identification Attribute

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 the owner of the file.

Be extremely careful when setting these permissions. Any user who runs that file assumes the permissions of the owner of the executable file, instead of the user who created the process. This is the cause of many "buffer overflow" exploits, typically resulting in superuser privileges.

The setuid permission is shown as an `s` in the file permissions. For example, the `setuid` permission on the `/usr/bin/passwd` command enables normal users to read and write an otherwise inaccessible `/etc/passwd` file:

```
user@myhost $ ls -l /etc/shadow /etc/passwd /usr/bin/passwd
-r--------   1 root     root          659 Jul 25 19:40 /etc/shadow
-rw-r--r--   1 root     root          711 Jul 25 19:40 /etc/passwd
-r-sr-xr-x   1 root     bin         15613 Apr 27 12:29 /usr/bin/passwd
```

You will notice that the `s` takes the place of the execute bit in the example above. This special permission mode really has no meaning unless the file also has execute permission as well.

In the example we see the `/etc/shadow` file is only readable by root, yet the `/usr/bin/passwd` file enables us to write our password changes there. When either a normal user, a member of the `bin` group, or even anyone else executes `/usr/bin/passwd`, it is really run as *root*, due to the "`s`" bit set in the *owner*'s permissions field.

Keep in mind that setuid has a different meaning when applied to directories. See the explanation for directories that follows.

It is advisable to keep `setuid` and `setgid` binaries on your system to a minimum, in order to reduce the possiblity of their being exploited. You should never execute an suid or sgid binary as a normal user, without knowing what it does. And certainly do not arbitrarily modify an otherwise non-setuid binary to have setuid permissions, simply for convience.

8.1.2 Set Group Identification Attribute

If set in the group permissions, this bit controls the "set group ID" status of a file. This behaves the same way as setuid, except the group is affected instead. The file must also be executable for this to have any effect. Upon execution of a file with this bit set, the effective group ID for the process is changed to the group owner of the file and a user is granted access based on the permissions given to that group. The `wall(1)` program, `/usr/bin/wall`, is used to "write all" users that are logged on to the system at the same time. It must be set group ID in order to have enough permission to write to terminals which do not belong to the user running the program:

```
user@myhost$ ls -l /usr/bin/wall
-r-xr-sr-x   1 root     tty          5492 May  7 14:02 /usr/bin/wall
```

We see here that everyone has the ability to execute the binary. It is owned by *root*, and a member of the `tty` group. Having each user on the system a member of the `tty` is not practical, and neither is changing the group to which the wall program belongs.

It is advisable to keep `setuid` and `setgid` binaries on your system to a minimum, in order to reduce the possiblity of their being exploited. You should never execute an suid or sgid binary as a normal user, without knowing what it does. And certainly do not arbitrarily modify an otherwise non-setuid binary to have setuid permissions, simply for convience.

Keep in mind that setgid has a different meaning when applied to directories. See the explanation for directories that follows.

8.2 Directory Permissions and Ownership

You can protect the files in a directory, and its subdirectories, by denying access to the entire directory itself. The permissions of a directory typically have a slightly different meaning than the equivilent permissions on a file. Additional permissions are available on directories, including setuid, setgid, and the sticky bit. Directory entries can have any one, or any, of the following:

```
Symbol        Permission      Description
----------------------------------------------------------------------
r             Read            List file contents
w             Write           Add, modify or remove files in the
                               directory
x             Execute         Open or execute files in the directory
-             Access Denied   Cannot be read, written, or executed,
                               depending on the position of the '-'
s             Special Mode    Set group ID bit is active (only in
                               ''group'' section
t             Special Mode    Save text attribute
```

It is important to understand the meanings of each of these symbols, and how you can use them to protect your files. Many of these symbols may be clear as to its meaning, but perhaps the other modes deserve a more in-depth explanation.

The `read` symbol indicates the ability to list the contents within the directory, assuming you also have access to open the directory.

The `write` symbol indicates the ability to add, remove, or modify files within the directory, also assuming you have access to open the directory. It is important to note that write access on a file within a directory is not required to delete it!

8.2.1 Save Text Attribute (Sticky Bit)

The *Save Text* (also known as the *sticky bit*) is an option really only available to directories. If the sticky bit is set on a directory, then a user may only delete files that the user owns or for which he has explicit write permission granted, even when he has write access to the directory. This is designed for directories 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.

For example, the `/tmp` directory is typically world-writable, so everyone has a place in which to write temporary files. The `/tmp` directory looks like this in a long-listing:

```
user@myhost$ ls -ld /tmp
drwxrwxrwt   3 root     root         2048 Aug 23 16:25 /tmp
```

This shows that everyone can read, write, and access the directory. But the "`t`" shows us that only the user (and root, of course) that created a file there can delete that file.

The `chmod(1)` command controls the sticky bit permissions. For example, you can add the sticky bit to a directory using the following:

```
root@myhost# ls -ld spool
drwxrwxrwx   3 root     root         2048 Aug 23 16:25 spool
root@myhost# chmod +t spool
root@myhost# ls -ld spool
drwxrwxrwt   3 root     root         2048 Aug 23 16:25 spool
```

While you can use the sticky bit on files, it does not really serve a purpose on Linux systems, as it did on UNIX systems of yester-year.

Additionally, this option should not be used casually. Instead, create a directory in the user's home directory to which he or she can write temporary files. The TMPDIR environment variable can be set, and programs that use the `tempnam(3)` system call will look for this variable and use it, instead of `/tmp` See the section on *Writing Secure Code* for a further explanation why there are hidden security problems with `/tmp`

8.2.2 Set Group Identification Attribute

If you set the setgid bit on a directory, files created in that directory will have the same group ownership as the directory itself, rather than the primary group of the user that created the file.

This attribute is useful when multiple users need to access specific files, but still require isolation from other files. Having them work from a common directory with the setgid attribute set means that any files created there will obtain the permissions of that common directory. For example, Joe and Mary might be in different primary groups, but need to collaborate on a common project. In this case, creating a common directory can be used to which both have write access.

You can control the setgid attribute on a directory with the following command:

```
joe@myhost$ ls -ld common_dir
drwxrwxr-x   2 joe     dev         1024 Aug 23 17:03 common_dir
joe@myhost$ chmod g+s common_dir
joe@myhost$ ls -ld common_dir
drwxrwsr-x   2 joe     dev         1024 Aug 23 17:03 common_dir
```

We can see here that the "s" in place of the execute bit in the group permissions indicates all files written to the `common_dir` will now belong to group `dev`

8.3 Changing File and Directory Permissions

The `chmod(1)` command controls the changing of file and directory permissions. Only the owner (or superuser, of course) can change the permissions of a file or directory.

The `chmod(1)` command has two modes of operation. The first one, called *absolute mode*, works by explictly specifying the permissions using an octal value, such as 644 or 755. The second mode of operation, called *symbolic mode*, works by using combinations of letters and symbols to add or remove permissions.

Using the octal values method of changing permissions can be more difficult to use at first, but you'll find it is faster and easier, once you have made the inital time investment, and learned how to do it correctly.

8.3.1 Changing File Permissions Using Octal Values (Absolute Mode)

The octal value for specifying permissions works by specifying a numeric argument for the permissions for which you wish to change. These numbers are used in sets of three to set permissions for *owner*, *group*, and *other* (everyone else). The following table shows what each octal value means:

```
Value          Permissions     Description
-------------------------------------------------------------------
0              ---             No permission
1              --x             Execute only
2              -w-             Write only
3              -wx             Write and execute (shell scripts need
                                read permission to be executed)
4              r--             Read only
5              r-x             Read and execute
6              rw-             Read and write
7              rwx             Read, write, and execute (full
                                control)
```

Using the table above, you can use `chmod(1)` to modify file and directory permissions. It helps to disect each of the sections, and explain one at a time. Given the following example:

```
user@myhost$ ls -l
-rwxrw-r--   1 dave     sysadmin    36012 Aug 21 01:06 run.pl
```

We see from this example that `dave` is the owner, and the file belongs to group `sysadmin`. From the information in the first field, we see this is a normal file, as shown by the `-` as the left-most character in the left-most field. The owner of this perl script, `dave`, has permission to read, write, and execute this file. The group, `sysadmin` has permission to read and write to it (including deleting it). Everyone else can only read this file. Using that information, we can look more closely at the permissions that file has:

```
Access Class    user    group   other
Symbolic Mode   r w x   r w -   r - -
Binary Mode     1 1 1   1 1 0   1 0 0
Octal Equiv       7       6       4
```

The octal equivilent of the binary number is generated using powers of two. Each position that is enabled, as shown by a `1` instead of a `0`, represents a power of two. Specifically, from right to left, we have 2ˆ0, or 1, then 2ˆ1, or 2, then 2ˆ2, or 4. Adding the enabled values corresponding to the bits that are enabled gives the octal number we use with `chmod(1)`.

One might decide to remove the ability for *other* to read this file. You can do this using `chmod(1)` as follows:

```
user@myhost$ ls -l run.pl
-rwxrw-r--   1 dave     sysadmin    36012 Aug 21 01:06 run.pl
user@myhost$ chmod 760 run.pl
user@myhost$ ls -l run.pl
-rwxrw----   1 dave     sysadmin    36012 Aug 21 01:06 run.pl
```

We see here that `run.pl` has now been modified to deny read access (as well as all other types of access) to users other than those in group `sysadmin`, and the owner (`dave` in this case)

8.3.2 Changing Directory Permissions Using Octal Values (Absolute Mode)

Using the same format as used to describe file permissions shown above, we will continue, and explain how changing directory permissions using octal values work.

The octal value for specifying permissions works by specifying a numeric argument for the permissions for which you wish to change. These numbers are used in sets of three to set permissions for *owner*, *group*, and *other* (everyone else).

The primary difference between permissions on files and permissions on directories is access control. Permissions on directories typically indicate accessibility. *Hint: You cannot execute a directory* ;->

The following table shows what each octal value means, as well as what access control is given for the corresponding permissions:

```
Value   Permissions     Description
--------------------------------------------------------------------
0       ---             No permission
1       --x             Access - gives ability to work with programs
                         and files in the directory that they already
                         know the name of, but hides all others
2       -w-             Write - really has no meaning on its own
3       -wx             Write and execute - ability to write to files
                         you already know the name of
4       r--             Read only - really has no meaning on its own
5       r-x             Read and execute - gives ability to enter
                        directory, and list contents, but cannot write
                         or delete
6       rw-             Read and write - really has no meaning on its
                         own
7       rwx             Read, write, and access - ability to list
                         contents of directory, as well as read and
                         write in it
```

Using the table above, you can use `chmod(1)` to modify file and directory permissions. It helps to disect each of the sections, and explain one at a time. Given the following example:

```
user@myhost$ ls -l
drwxr-x---   1 dave     sysadmin     1024 Aug 21 01:06 games
```

We see from this example that `dave` is the owner, and the directory belongs to group `sysadmin`. From the information in the first field, we see this is a directory, as shown by the `d` as the left-most character in the left-most field. The owner of this directory, `dave`, has permission to read, write, and access this directory. The group, `sysadmin` has permission to access the directory, as well as list its contents. Files within this directory with the appropriate read permission would also be able to be read. Other users are not allowed to access this directory at all. Using that information, we can look more closely at the permissions that directory has:

```
Access Class            User    Group   Other
Symbolic Mode           r w x   r - x   - - -
Binary Mode             1 1 1   1 0 1   0 0 0
Octal Equivilent          7       5       0
```

The octal equivilent of the binary number is generated using powers of two. Each position that is enabled, as shown by a `1` instead of a `0`, represents a power of two. Specifically, from right to left, we have 2^0, or 1, then 2^1, or 2, then 2^2, or 4. Adding the enabled values corresponding to the bits that are enabled gives the octal number we use with `chmod(1)`.

One might decide to give other users the ability for *other* to access this file, and list the contents within it. You can do this using `chmod(1)` as follows:

```
user@myhost$ ls -ld games
drwxr-x---   1 dave     sysadmin     1024 Aug 21 01:06 games
```

```
user@myhost$ chmod 755 games
user@myhost$ ls -ld games
drwxr-xr-x   1 dave     sysadmin     1024 Aug 21 01:06 games
```

We see here that `games` has now been modified to permit access to users other than those in group `sysadmin`, and the owner (`dave` in this case)

8.3.3 Changing Permissions Using Symbols (Symbolic Mode)

The *symbolic mode* is perhaps the easier of the two methods to use to change file permissions. It is probably the one you should work with first if you are just learning this. This section discusses the basic means in which one can change the permissions of a file or directory, using `chmod(1)`

The symbolic mode of `chmod(1)` works on the concept of access classes. These classes consist of *(u)ser*, which is the owner of the file, *(g)roup*, of which the user is a member, and *(o)ther*, which is those users not a member of the group, or the owner of the file. The final mode is *(a)ll*, which consists of all three of the previous modes.

Using these modes, in conjunction with the desired permissions, you can modify the access to a particular file or directory. The permissions are one or more of *(r)ead*, *(w)rite*, and *e(x)ecute*.

Combining the access class and the new permissions desired, with an operator, gives you the ability to change the permissions on a file or directory. The available operators are `+`, which means to add to the existing permissions, `-`, which means to subtract from the existing permissions, and `=`, which means set the new permissions equal to those provided.

For example, "`a+rw`" means to add *read* and *write* permission to *all* three groups of users. Using "`go=r`" means to set the *group* and *other* fields to only have *read* access, regardless of what they had previously.

A more complete example is as follows:

```
dave@myhost$ ls -l nsmail
drwxr-xr-x   2 dave     dave         1024 Aug  7 00:17 nsmail
dave@myhost$ chmod go=rx nsmail
dave@myhost$ ls -l nsmail
drwx------   2 dave     dave         1024 Aug  7 00:17 nsmail
```

To remove write access for everyone from a file, use the minus sign:

```
dave@myhost$ chmod a-w myfile
dave@myhost$ ls -l myfile
-r--r--r--   1 dave     dave          424 Aug 23 23:10 myfile
```

You can control the setuid and setgid on files and directories, as well as the sticky bit, using the symbolic mode with `chmod(1)`. Such an example might be as follows:

```
1.    root@myhost# ls -l
2.    drwxr-xr-x   2 root     sysadmin     1024 Aug 24 01:18 groupdir
3.    -rwxr-x---   1 root     sysadmin     8077 Aug 24 01:19 myprog
4.    drwxr-xr-x   2 root     root         1024 Aug 24 01:18 spool
5.    root@myhost# chmod g+ws groupdir
6.    root@myhost# chmod u+s myprog
7.    root@myhost# chmod o+t,a+w spool
8.    root@myhost# ls -l
9.    drwxrwsr-x   2 root     sysadmin     1024 Aug 24 01:18 groupdir
```

```
10.   -rwsr-x---   1 root     sysadmin     8077 Aug 24 01:19 myprog
11.   drwxrwxrwt   2 root     root         1024 Aug 24 01:18 spool
```

This is an interesting example which uses many of the features of `chmod(1)`. Lines 1 through 4 show the long-list of the file and two directories before any changes were made. We see here that `groupdir` and `myprog` are members of group `sysadmin`. Another point of interest is that no one but the owner of these files (`root` in all these cases) is able to write to the file or directories.

Line 5 shows how to add both group `write` permission, and `setgid` access to the `groupdir` directory. This will enable members of group `sysadmin` to write files there, and retain the `sysadmin` group.

Line 6 shows how to add the setuid bit to the `myprog` binary. This means that any user in the `sysadmin` group that executes this binary is granted access based on the owner of the file, in this case `root`, rather than the user who executed it.

Line 7 shows how to add the sticky bit to the `spool` directory, as well as add `write` permission for `all` users. This is a publicy-accessible directory, and writable by all. However, only those who actually own the files can delete them.

Lines 8 through 11 show the directories and file after the modifications have been made.

8.4 Changing File Ownership

This section discusses the methods in which an administrator can change the owner and group to which a file belongs. Use the `chown(1)` command to change a files owner (can only be done by root), and `chgrp` to change the group to which a file or directory belongs.

As with any security-related task, you should use caution when changing the ownership of a file or directory. Most times you can add a user to a group without having to change the ownership. You should also re-evaluate the permissions of the file or directory after you have made the change.

To use the `chown(1)`, supply the new username and the files you wish to change:

```
root@myhost# ls -l myfile
-r--r--r--   1 fred     sysadmin      424 Aug 23 23:10 myfile
root@myhost# chown root myfile
root@myhost# ls -l myfile
-r--r--r--   1 root     sysadmin      424 Aug 23 23:10 myfile
```

You can also change ownership of files recursively by using the `chown -R` option. When you use the `-R` option, the `chown` command descends through the directory and any subdirectories below that one, changing the ownership.

If a symbolic link is encountered, the group ownership is changed on the file to which the link points.

8.5 Changing Group Ownership

This section is very similiar to the previous section. It discusses the methods in which an administrator can change the groups to which a file belongs. Use the `chgrp(1)` command to change group ownership. In order for a normal user to change a file's group from one to another, the user must be a member of both groups.

To use the `chgrp(1)`, supply the new group name and the files you wish to change:

```
root@myhost# ls -l myfile
-r--r--r--   1 fred     sysadmin      424 Aug 23 23:10 myfile
```

```
root@myhost# chgrp root myfile
root@myhost# ls -l myfile
-r--r--r--   1 fred     root          424 Aug 23 23:10 myfile
```

You can also change group ownership of files recursively by using the `chgrp -R` option. When you use the `-R` option, the `chgrp` command descends through the directory and any subdirectories below that one, changing the ownership.

You can also use the `chown(1)` command to change both the owner and group at the same time. Use a colon between the desired new owner and group. For example:

```
root@myhost# ls -l myfile
-r--r--r--   1 fred     sysadmin      424 Aug 23 23:10 myfile
root@myhost# chown root:root myfile
root@myhost# ls -l myfile
-r--r--r--   1 root     root          424 Aug 23 23:10 myfile
```

Notice the permissions do not change simply because you have changed the ownership. Use caution here to be sure you are not inadvertantly giving permission to someone that should not have it.

If a symbolic link is encountered, the group ownership is changed on the file to which the link points.

8.6 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, a user could inadvertently give read or write permission to someone that should not have this permission.

The umask for the creation of new executable files is calculated as follows:

```
  777 Default Permissions
 -022 Subtract umask value, for example
-----
  755 Allowed Permissions
```

So in this example we chose `022` as our umask. This shows us that new executables that are created are given mode `755`, which means that the owner can read, write, and execute the binary, while members of the group to which the binary belongs, and all others, can only read and execute it.

The umask for the creation of new text files is calculated as follows:

```
  666 Default Permissions
 -022 Subtract umask mask, for example
-----
  644 Allowed Permissions
```

This example shows us that given the default umask of `666`, and subtracting our sample umask value of `022`, new text files are created with mode `644`, which states that the owner can read and write the file, while members of the group to which the file belongs, and everyone else can only read the new file.

Typically 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 must be set while keeping in mind the purpose of the account. Permissions that are too restrictive may cause users to start sharing accounts or passwords, or otherwise compromise security. 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 to at least 022, which will disable write and execute permission for other users, unless explicitly changed using `chmod(1)`.

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

In addition to setting the user's default umask, you should be sure you are aware of the umask value that is set in startup scripts as well. Any files that are created during the boot process may be created with the default umask of 666 if it is not explictly specified.

Additionally, any servers that are started at boot time, such as `inetd(8)`, may inherit the umask at boot time, which in turn will be passed down to the services, and servers, that it controls.

The umask value that the FTP server, spawned by `inetd(8)` uses, for example, can be easily overlooked, allowing the potential for too lenient permissions on files.

In this specific example, the FTP server has command-line options for controlling umask values. Many do not, however. For this reason, you might consider creating a file that gets run at system boot time, before any others, that simply explictly sets the umask to a known value.

8.7 Monitoring Files with Special Permissions

You should regularly monitor your systems for any unauthorized use of the `setuid` or `setgid` permissions to gain superuser privileges.

`setuid` and `setgid` 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 "`setuid root`" programs, then leave a `setuid` program as a back door to get in the next time, even if the original hole is plugged.

Find all `setuid` and `setgid` 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 `setuid` and `setgid` programs on your system:

```
root@myhost#  find / -type f -perm +6000 -ls
```

You can discriminately remove the `setuid` or `setgid` permissions on a suspicious program with `chmod(1)`, then change it 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@myhost# 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 writable, including some from `/dev`.

Unowned files may also be an indication an intruder has accessed your system. You can locate files on your system that do not have an owner, or belong to a group with the command:

```
root@myhost# find / -nouser -o -nogroup
```

8.8 General Guidelines

The following is a list of general guidelines you should be aware of when configuring the files on your hosts.

- There should never be a reason for user's home directories to allow `setuid` and `setgid` 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 user's home partitions, as well as `/var`, which prohibits execution of programs, and creation of character or block devices, which should never be necessary anyway.
- User files can introduce system vulnerabilities. Some of the things you should watch for include:
  - Installation of Trojan horse programs
  - Protect personal start-up files from modification by others
  - Do not specify personal or shared directories before system-provided directories in executable search paths. (This invites the installation of Trojan horses.)
  - Default protections assigned at file creation should meet system standards
  - Limit write access in a user's personal file space (by appropriate protection of user directories).
- System files are a crucial component in preventing a security incident. Some of the things to watch for here include:
  - The system configuration files and shared binaries must be protected against Trojan horses
  - Audit trails must be protected against undesired modification
  - Restrict modification privileges for system binaries to systems staff
  - Review the content of system binaries for unexpected changes
  - Restrict modification of system start-up scripts to systems staff
  - Review content of system start-up scripts to ensure that secure defaults are specified and programs executed are not candidates for Trojan horse conversion
  - Protect audit trail log files from unauthorized modification
- If you are mounting filesystems using a network filesystem such as NFS, be sure to configure `/etc/fstab` 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    5000
  @users     hard  nproc   50
  @users     hard  rss     5000
  ```

  This says to limit the creation of core files, restrict the number of processes to 50, and restrict memory usage per user to 5 Meg.
- The `/var/log/wtmp` and `/var/run/utmp` files contain the login records for all users on your system. Its integrity must be maintained because it 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.
- System configuration files (usually in `/etc`) are usually mode `644 (-rw-r-r-)`, and owned by root. Depending on your sites 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.

- `setuid` 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.

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. And removing permissions from files is typically a good idea, but it is not always practical. Change permissions slowly, and watch carefully for undesired results.

9 Data Encryption, Cryptography and Authentication

An integral part of host and network security is data encryption. There are vast resources of information on the Internet available on the topic of data security. Various data encryption mechanisms are available for use with Linux.

This section attempts to discuss some of the encryption features that are available for use with Linux. For an overview of encryption and cryptography, be sure to consult the RSA Cryptography FAQ, available at *http://www.rsa.com/rsalabs/newfaq/*

9.1 Password Security

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 password 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.

Most UNIXs (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, whatever 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 unicies 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`).

Any entry in the password file with a user-ID of "0" (zero) is a root entry, regardless of what it's called.

Choose effective passwords. There is a great deal of information available on the Internet regarding choosing good passwords. A password minimum of 6 characters should be enforced, and 8 characters provides a significant improvement in security. You can find more information on improving password security at *ftp://sunos-wls.acs.ohio-state.edu:/pub/security/Dan_Klein_password_security.ps.Z* which is titled "Foiling the Cracker: A Survey of, and Improvements, to Password Security".

Brute force attacks, such as "Crack" or "John the Ripper" (see below) 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 go to *http://consult.cern.ch/writeup/security/security_3.html* for information on how to choose a good password.

There is also a quick list of things to keep in mind when choosing a password available at *http://www.alw.nih.gov/Security/Docs/passwd.html* and should be consulted when developing your security policy.

9.2 PGP and Public Key Cryptography

Public Key Cryptography, such as that which is used for PGP, involves cryptography that uses one key for encryption, and one key for decryption. Traditionally, cryptography involves using the same key for encryption that is used for decryption. This "secret key" must be known to both parties, and somehow transferred from one another securely.

Public key encryption alleviates the need to securely transmit the key that is used for encryption by using 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 a the PGP FAQ. *http://www.pgp.com/service/export/faq/55faq.cgi*

Be sure to use the version that is applicable to your country, as 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 the International Traffic in Arms Regulations (ITAR).

There is a good introductory guide explaning public key cryptography, that includes graphic illustrations, available at PC Magazine Online, available *http://www8.zdnet.com/pcmag/features/inetsecurity/howencrypt.htm*

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, which is available at *ftp://sunsite.unc.edu/pub/Linux/apps/crypto.*

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.

An excellent 147-page publication written by the government describing practically all you'll need to know unless you're a cryptographer is available at *http://csrc.nist.gov/nistpubs/800-2.txt*

There is a project working on a free re-implementation of PGP with open source. See the GNU Privacy Guard web page for more information, available at *http://www.d.shuttle.de/isil/crypt/gnupg.html*

9.3 SSL, S-HTTP, HTTPS and S/MIME

Often times 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 are, 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, authenticity, integrity, and non-repudiability (cannot be mistaken for someone else, and I cannot deny my actions later) 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, or 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*.

9.4 IPSec and S/WAN and other IP Encryption Implementations

IPSec is an effort by the IETF to create cryptographically secure communications at the IP network level, which also provides authentication, integrity, access control, and confidentiality. IPsec is the basic host-to-host security mechanism. It is appropriate for use any time address-based protection would have been used, including with such programs as `rsh` and `rlogin`. If and when platforms support user-based keying, this scope may be expanded. 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.

A good starting point for Linux implementations of Virtual Private Networking is available at *http://www.imib.med.tu-dresden.de/imib/Internet/index.html*

One of the Linux implementations, 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.

There is also an implementation of RSA's Secure Wide Area Networking, S/WAN, called FreeSWAN, available at *http://www.xs4all.nl/˜freeswan/*

A description of S/WAN is available at *http://www.sunworld.com/swol-06-1996/swol-06-swan.html*

Microsoft Point-to-Point Tunneling Protocol is also available for Linux. You can find more information on this at *http://www.pdos.lcs.mit.edu/˜cananian/Projects/PPTP/*. More information on this protocol is available from the Linux PPTP page.

An implementation of PPTP that works with Linux masquerading is available at *http://bmrc.berkeley.edu/people/chaffee/linux_pptp.html* as well as kernel patches, and a pointer to more information.

It is well known now that PPTP is insecure, and really should only be used in existing installations. Rhino9, the security research group, have put together an exploit, as well as more documentation on the protocols involved. You can find it at *http://www.rhino9.ml.org/texts/pptp.doc*

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

9.5 The Secure Shell and Secure Telnet

SSH and stelnet are programs that allow you to login to remote systems and have a encrypted connection.

SSH 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 for user authentication. This 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. The SSH home page can be found at *http://www.cs.hut.fi/ssh/*

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.*

There is also an open source implementation of SSH being developed. You can find more information about this at *http://www.net.lut.ac.uk/psst/*

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/*

An SSL-based POP3 daemon is also available at *http://mike.daewoo.com.pl/computer/stunnel/*

9.6 SKIP - Simple Key management for Internet Protocols

SKIP, which provides IP-Level cryptography, much like SSH, is available for Linux. A quick overview from *http://www.skip.org* states:

> SKIP secures the network at the IP packet level. Any networked application gains the benefits of encryption, without requiring modification. SKIP is unique in that an Internet host can send an encrypted packet to another host without requiring a prior message exchange to set up a secure channel. SKIP is particularly well-suited to IP networks, as both are stateless protocols. Some of the advantages of SKIP include:

- No connection setup overhead
- High availability - encryption gateways that fail can reboot and resume decrypting packets instantly, without having to renegotiate (potentially thousands) of existing connections
- Allows uni-directional IP (for example, IP broadcast via satellite or cable)
- Scalable multicast key distribution
- SKIP gateways can be configured in parallel to perform instant-failover

There is a wealth of information available at *http://www.skip.org* as well as the actual Linux implementation available at *http://www.tik.ee.ethz.ch/˜skip/*

9.7 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 on the fly your authentication methods, requirements, and encapsulate all local authentication methods without re-compiling 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 a non DES encryption 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 dot-rhosts files in user's home directories by adding these lines to /etc/pam.d/login:

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

9.8 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 travelling 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 tunneling, 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.

9.9 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.

The overall effect of installing Kerberos and the numerous other programs that go with it is to virtually eliminate the ability of users to "spoof" 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 on kerberos at *http://www.veritas.com/common/f/97042301.htm* 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.]

9.10 Shadow Passwords.

Shadow passwords are a means of keeping your encrypted password information secret from normal users. Normally this encrypted password is stored in your /etc/passwd file for all to read. They can then run password guesser programs on it and attempt to determine what it is. Shadow passwords save this information to a /etc/shadow file that only privileged users can read. In order to run shadow passwords you need to make sure all your utilities that need access to password information are recompiled to support it. PAM (above) also allows you to just plug in a shadow module and 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://sunsite.unc.edu/LDP/HOWTO/Shadow-Password-HOWTO.html* It is rather dated now, and will not be required for distributions supporting PAM.

9.11 Crack and John the Ripper

If for some reason your passwd program is not enforcing non easily guessable 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. They encrypt each one and check it against your encrypted password. If they get a match they are in. Also, the "dictionary" may include usernames, Star Trek ships, foreign words, keyboard patterns, etc...

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 get your passwd (Unix /etc/passwd) file, but these are more common than you might think.

9.12 Cryptography and File Systems

Linux provides several mechanisms in which to encrypt data on a filesystem.

CFS is a way of encrypting entire directory trees and allow users to store encrypted files on them. It uses a NFS server running on the local machine. RPMs are avail at *http://www.replay.com/redhat/* and more information on how it all works is at: *ftp://ftp.research.att.com/dist/mab/*

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

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

There are two implementations of DES encryption on the loopback device also available. It is available at *ftp://ftp.csua.berkeley.edu/pub/cypherpunks/filesystems/linux* Patches to the 2.0 kernel and the mount executable are available at *ftp://ftp.is.co.za/linux/local/kernel/crypto/loopback-device-berkeley-recent/* Patches to the 2.1 kernel, written by Andrew E. Mileski, *aem@netcom.ca* are available at *ftp://ftp.is.co.za/linux/local/kernel/crypto/loopback-device-aem*

10 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 the kernel is very secure, and the kernel itself won't be compromised. To prevent some of the latest networkworking attacks, you should try and keep your kernel version current. You can find new kernels at *ftp://ftp.kernel.org*

10.1 Securing Hosts with Many Users

Your terminal servers, or servers with many users, can be further protected by employing Solar Designer's <*solar@false.com*> experimental "Secure Linux" kernel patches. As this is still experimental, and not a guaranteed fix, this patch is typically only advisable for hosts with many users, and not servers such as web or email servers. His work states he has done the following:

- Make user stack area non-executable
- Restricted links in `/tmp`
- Restricted pipes in `/tmp`
- Further restrict accessibility to /proc

See the documentation that comes with the software for more information.

10.2 Securelevel, Capabilities and Linux-Privs

Much of the security work being done these days is to prevent someone from "upgrading" their normal user privileges, and becomming root. If we could remove that ability entirely, it may help to solve many of the exploits we currently see.

Andrew Morgan has written a series of kernel patches, called *Linux-Privs* works towards implementing POSIX.1e (formerly POSIX 6) security model under Linux. It is a scheme that more precisely defines device and application permissions. Andrew states, *Typically, the user will have to authenticate himself to such an applciation before it will perform its privileged task. This new scheme for system privilege lends itself well to restricting privileged access to the system and reduces the risk of intruders or poorly written applications running amok on the system.*

There is an introductory document available at *http://www.kernel.org/pub/linux/libs/security/linux-privs/doc/linux-privs.html/linux-privs.html* which discusses the features he has implemented, as well an outline for those which still need to be worked on. Some of the available abilities include Access Control Lists (ACL), Mandatory Access Control (MAC), and Kernel-level auditing.

To quote James T. Dennis <*answerguy@ssc.com*> in the July 1998, issue of Linux Gazette (available at *http://www.ssc.com/lg/*)

One approach would be the POSIX.1e "capabilities" (which are more like VMS style "privileges" than true "capabilities"). There is a bit of preliminary work being done on this in the 2.1.x kernels — but nothing is likely be usable in 2.2 (so you're looking at Linux 2.4 before there is "stable" support for any of that).

Another approach is to limit the damage that "root" can do using something like the BSD securelevel features. Last I heard on the Linux kernel mailing list they had dropped plans to put in simple "securelevel" support in favor of a "more flexible" approach — which would mesh better with the eventual POSIX.1e ("Orange Book") work.

His discussion is really based on further securing the chroot() function call using "capabilities", but has some generally useful descriptions as well. You can find this discussion at *http://www.ssc.com/lg/issue30/tag_chroot.html*

Briefly, it will allow you to disable access to functions such as mknod(), chroot(), mount(), etc, and move the privileges to the executable itself, rather than simply by being the root user.

There is further information in the Linux kernel 2.1.x sources, as well as the June 25 issue of Linux Weekly News available at *http://www.lwn.net/980625/*

10.3 Kernel Compile Options

- 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 the packet goes thru does not need to inspect the packet, and just forwards it on. This could lead to data entering your system that may be a potential exploit.
- 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: 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 and disable 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
  ```

  This file (and many other files in /proc) will always appear to be zero length, but in fact aren't. This is a newly introduced kernel feature, so be sure you are using a kernel 2.0.33 or later.
- IP: firewall packet logging (CONFIG_IP_FIREWALL_VERBOSE) This option gives you information about packets your firewall received, like sender, receipient, port, etc.
- 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 for your connection that aren't supposed to be there. It could also prevent a teardrop attack against an internal host that is not yet itself patched against it.

- IP: syn cookies (CONFIG_SYN_COOKIES) 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 2.1 kernel series this config option mearly allows syn cookies, but does not enable them. To enable them, you have to do:

```
root@myhost# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

- Packet Signatures (CONFIG_NCPFS_PACKET_SIGNING) This is an option that is available in the 2.1 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 userspace program, to determine if you would like to accept or deny the packet, based on its validity.

10.4 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 retrieve 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 requisite. 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 random.

The only difference 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 entry 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:

```
user@myhost# head -c 6 /dev/urandom | mmencode
```

This will print (approximately) six random characters on the console, suitable for password generation. You can find `mmencode(1)` (perhaps also known as mimencode on some systems) in the metamail mail 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.

11 Exploits

The diversity of today's networks exposes your system to a wide variety of possible security-related incidents. In order to protect your systems, you must be aware of these exploits in order to protect yourself from them. While previous sections explained the types of people to protect against, and the reasons they attack, this section attempts to explain the types of exploits that are typically performed to break into a computer system.

There are several exploits that won't be mentioned here, such as Macro Code Attacks and Virus Infections, of which Linux and Unix itself in general is not susceptible. However, any Windows-based systems that connect to it will be suseptible, via shared filesystems, electronic mail, etc.

There are now several programs available to check your system for the most common exploits. The rootshell site, *http://www.rootshell.com* has several of these programs, and there is also the following available *ftp://ftp.fu-berlin.de/unix/security/chkexploit/*

11.1 Worm Attacks

Worms are problems which replicate themselves, but unlink viruses they do not modify other programs and are not triggered by user actions. Worms are self-contained programs that attack systems or other programs without changing them in any way, and that typically use networks to accomplish this. The Internet Worm, which reportedly gained access to more than 6,000 Unix systems, flooded the Internet with so many access requests that it became unusable. These are no where near as common as they once were.

11.2 Trojan Horse Programs

A trojan horse is a program that is an unauthorized, self-contained program that is not self-replicating. It is often hidden or given a misleading name to deter suspicion.

A Trojan Horse is named after the fabled ploy in Homer's great literary work. The idea is that you put up a program or binary that sounds great, and get other people to download it and run it as root. Then, you 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 signs RPM files so you can verify you are installing the real thing. Other distributions have similar methods. You should never run any binary you don't have the source for or a well known binary 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 some program from it's 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.

11.3 Cracking Attacks

Cracking attacks are attacks perpetrated by network intruders, or crackers (formally known as hackers). These attacks take the form of network intrusions, which are break-ins into remote systems, or the use of the services they provide, without authorization. The number of cracker attacks is proliferating more rapidly than any other type of incident, in large part because the Internet provides broad connectivity without intrinsic security mechanisms.

Information security professionals have long accepted the premise that more incidents are caused by insiders (e.g., company employees and contractors) than by outsiders. Many feel this trend is now reversing, and news of organizations' incurring major financial losses as the result of network intrusions is becoming commonplace.

Obviously, neither type of exploit should be taken lightly.

11.4 Direct Physical Access

Users often log on to workstations and then leave them unattended for long periods of time. This allows unauthorized individuals physical access to the workstations and to the organization's systems. An attacker can enter the office and use the workstation to attack numerous systems at a commercial site.

Attacks involving direct physical access can be extremely costly, because the attacker is often an insider who knows exactly where valuable data and applications reside on the system.

See the section on physical security for more information on how to protect your system.

11.5 Spoofing

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 that you are someone that you really aren't. Spoofing of network connections involves forging an IP source address to trick the destination into thinking you are someone you really aren't. Spoofing of network services involves using poorly configured (or misconfigured) applications, typically SMTP, to trick the client, server, or receipient into thinking you are someone you are not.

Using the most recent implementations of the available service can help to protect against this "masquerading". Preventing internal IP addresses from seemingly entering your firewall from the outside is something that should be a mandatory addition to your rulebase. There is some information on preventing DNS spoofing available at *http://www.sunworld.com/swol-11-1997/swol-11-bind.html*

A general guide to securing DNS is available at *http://www.psionic.com/papers/dns-linux.html*

A great reference of spoofing information is available at *http://www.unitedcouncil.org/text.html* including the excellent article published in Volume Seven, Issue Forty-Eight of Phrack, available here *http://www.unitedcouncil.org/spoof/IPSpoofing.txt* This paper will help you understand the low-level TCP details.

11.6 Denial of Service Attacks

A Denial of Service (DoS) attack is one where the attacker prevents legimitate users from accessing a service. Denial of service attacks either try to make some resource too busy to answer legitimate service requests, or to deny legitimate users access to a machine.

Also of significant concern is a denial of service attack that is really intended to keep the victim busy while really the intruder is impersonating the host, preventing it from replying. These are typically referred to as "man in the middle" attacks.

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.

- **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 the section on kernel security for proper kernel protection options.

- **Pentium "F00F" Bug** - It was discovered in the summer of 1997 that a series of assembly codes send 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 kernel 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 broadcast address with *your* machines 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 available at *ftp://ftp.ee.lbl.gov/tcpdump.tar.Z* (although it should be part of your Linux vendor's distribution) and is used 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.

- **Email Bombing and Spamming** - Sending out large quantities of unsolicited email can clog networks, causing depletion of resources, and degradation of network bandwidth. The newer versions of sendmail have greatly improved support for eliminiating this problem.

- **Ping 'o Death** - The Ping 'o Death attack is a result of incoming ICMP ECHO REQUEST packets being larger than the kernel data structures that store this information can hold. 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. Someone has put together a further discussion of the Ping 'o Death attack, and is available at *http://www.sophist.demon.co.uk/ping/*

- **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 most exploit code, and a more in-depth description of how they work at *http://www.rootshell.com* using their search engine. Keep in mind that exploits are found on a periodic basis, and this list just describes the most popular.

11.7 Program Code Exploits

Much work is being done in this area by some very capable people to proactively catch these problems before further exploits are discovered. The Linux Security Audit Group is working on auditing many of the stock packages that vendors ship with their distributions. You can follow their efforts, or even help evaluate programs by joining the security audit list, *security-audit-subscribe@ferret.lmh.ox.ac.uk* and using "subscribe" in the body of the message. You can find the mailing list archives at *http://www.nas.nasa.gov/Pubs/Mail/archive/linux-security-audit/* This is strictly an auditing list. It does not discuss issues regarding configuring your system to be more secure, reporting an exploit, etc. Do not expect to find information here about steps to perform an exploit.

Be sure to keep your subscription information, as it is very distracting to see unsubscribe requests being sent to the list. You can unsubscribe from the list by sending "unsubscribe security-audit" in the body of the message to *security-audit-unsubscribe@ferret.lmh.ox.ac.uk*

Chris Evans is doing a fine job of maintaining the mailing list, as well as a list of outstanding security issues, specifically, those in Red Hat 5.1. You can find this list at *http://www-jcr.lmh.ox.ac.uk/˜chris/rhbugs.txt*

Some of the types of exploits performed on flaws in programming consist of at least the following:

- **Exploits in Vendor Packages** - Vendors frequently update the packages in their distribution to include security exploit fixes. It is important that you remain aware of these changes, and apply their fixes as they are distributed. Most vendors have mailing lists to notify users of changes as they happen, and you should subscribe to those lists. See the Web Links section for URLs to the most common Linux vendor security updates, and the Mail Links section for addresses for notification from most security vendors. Additionally, there are several user-contributed programs that will monitor particular ftp sites for changes, and either notify you when they change, or update automatically.
- **Buffer Overflow** - Common coding style is never to allocate buffers "large enough" and not checking for overflows. When such buffers are overflows, 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.
- **World Writable Directories** - Directories such as `/tmp` are typically used for temporary files, such as are created by the line printer daemon, X11, accounting programs, etc. A potential for guessing the names of the files written to this directory exists. As a result, poorly coded programs may have the potential for being exploited by writing into a prexisting file. For a more complete explanation, see the *Writing Secure Code* section.

11.8 Misconfigured Services

Misconfigured, or unnecessary services pose a significant threat to both host and network security. Exportable filesystems, inherently insecure services, too lenient configuration of a service, can all lead to a compromise.

Be sure to turn off any service that is not being used, and remove any executables that are not used. See the *Network Security* and *Host Security* for further information.

11.9 Known Vulnerabilities

There is certainly nothing easier than gathering the latest exploits from *http://www.rootshell.com* and trying them out on a list of machines.

Typically by the time the exploits are available on the Internet, the vendor has distributed a patched version of the susecptible program. Be sure to install these updated versions, or at the least disable the service until you can do so. See the *Contacts* section of this document for the locations of vendors' updates.

11.10 WWW and CGI-BIN attacks

Please see the WWW Security FAQ *http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html* for more information.

You can also find information on further securing Apache at *http://www.apache.org/docs/misc/security_tips.html*

12 Firewalls and Border Patrol

Linux can also be used as a full-featured utility to protect your internal network.

There are currently several firewall systems that run on Linux. Packet filters, application gateways (proxy gateways), IP masquerading, Network Address Translation (NAT), as well as IP accounting are all available on Linux.

Firewalls are a means of restricting what information is allowed into and out of your local network. Typically the firewall host is connected to the Internet and your local network, and the only access from your network to the Internet is through the firewall. A firewall is not just a program that runs on a Linux box. A firewall establishes a perimeter that controls all entry and exit points to your internal network. The strongest firewall won't protect you from a modem on one of the PCs inside your network.

There are a number of types and methods of setting up firewalls. Linux machines make pretty good low cost firewalls. Firewall code can be built right into 2.0 and higher kernels. The `ipfwadm(8)` user space tool configures the kernel-based packet filtering, allowing you to change what types of network traffic you allow on the fly. You can also log particular types of network traffic.

Firewalls are a very useful and important technique in securing your network. It is important to realize that you should never think that because you have a firewall, you don't need to secure the machines behind it. This is a fatal mistake.

12.1 Introduction

It is important you are familiar with not only the methods in which you can configure a firewall, but also which type of firewall gets used in which situation. This document, *http://www.sunworld.com/swol-01-1996/swol-01-firewall.html* is an excerpt from the O'Reilly book *Building Internet Firewalls* which is practically mandatory reading if you've never worked with firewalls first.

There are, however, several very well written documents available on the Internet, including some of these:

- *An Introduction to Firewalls*, written by the National Institute of Standards and Technology, has written a very impressive guide to understanding firewalls, available at *http://csrc.nist.gov/nistpubs/800-10/main.html*. Be sure to browse around that site for some other excellent information.
- *Internet Firewalls and Security*, written by the good folks at 3Com, also gives an excellent overview of the various types of firewalls, which one performs which functions, a few examples, the types of attacks you can prevent with a firewall, etc. It is available in Adobe PDF format, as well as online, here *http://www.3com.com/nsc/500619.html*
- *Internet Firewalls Frequently Asked Questions*, This document explains why you would want a firewall, how a proxy server works, etc. Written by Marcus Ranum, the highly-respected authority of firewalls, and is available on his home page, *http://www.clark.net/pub/mjr/pubs/fwfaq/*
- *Thinking About Firewalls* is a paper written by Marcus Ranum, discussing the various components of firewalls, why you would want one, several different configurations, and generally good advise. You can find this one at *ftp://ftp.tis.com/pub/firewalls/firewall.ps.Z*
- *Building Internet Firewalls*, the highly-acclaimed book that is well-worth it's money. Written by O'Reilly and Associates, ISBN 1565921240.

12.2 General Documentation

There is also a wealth of Linux-specific firewall-related material available. Some of them are on very specific topics, but others are excellent general overviews, written to address problems with documenting the proper procedures to use the tools Linux has available. Some of the documents you should check out include:

- A general introduction to using `ipfwadm(8)` is available here *http://linux.samiam.org/firewall.html* and should be read in conjunction with the Firewall-HOWTO.
- The *Freefire Project* is a project developed by Bernd Eckenfels attempts to *"help Developers in building Firewall and IT-Security Solutions based on Free Tools.* You can find additional information about the Project at the Homepage *http://www.inka.de/sites/lina/freefire-l/index_ en.html*
- The *firewalls* mailing list is available to ask questions regarding configuration, implementation, exploits involving firewalls, commercial as well as freely available implementations, encryption involving firewalls, etc. You can find subscription information, as well as searchable archives at *http://www.lists.gnac.net/firewalls/*
- Alan Cox has written quite a few articles for Boardwatch magazine, *http://www.boardwatch.com* that specifically discuss Linux security and firewall issues. You can find some general *ipfwadm(8)* examples, as well as a security overview, in two parts, at:
  - *http://boardwatch.internet.com/mag/97/aug/bwm70.html.*
  - *http://boardwatch.internet.com/mag/97/oct/bwm70.html*

12.3 The Firewall Toolkit

The most full-featured firewall available for Linux is the Firewall Toolkit (FWTK), written by Trusted Information Systems. Their web page states: *The TIS Internet Firewall Toolkit is a set of programs and configuration practices designed to facilitate the building of network firewalls. Components of the toolkit, while designed to work together, can be used in isolation or can be combined with other firewall components. The toolkit software is designed to run on UNIX systems using TCP/IP with a Berkeley-style "socket" interface.* You can find an overview of its features, a description of how it works, and the source code itself at *http://www.tis.com/prodserv/fwtk/fwtkoverview.html.* Trusted Information Systems, now really Network Associates *http://www.nai.com/*, has some generally good information also on their web site.

You can download the source for the Firewall Toolkit at *ftp://ftp.tis.com/pub/firewalls/toolkit/fwtk-v1.3.tar.Z* The documentation may be downloaded seperately from *ftp://ftp.tis.com/pub/firewalls/toolkit/fwtk-doc-only.tar.Z*

The Linux Journal wrote an excellent article on configuring the Firewall Toolkit in their Issue 25. The transcript of that article is available at *http://www.ssc.com/lj/issue25/1204.html*

Additionally, it may be necessary to patch the downloaded version of the Firewall Toolkit. You can find these patches at *ftp://ftp.tisl.ukans.edu/pub/security/firewalls/fwtkpatches.tgz*

12.4 Packet Filtering and Accounting

The `ipfwadm(8)` tool is used to build packet filtering rules on the 2.0 series kernels. With this tool, you can accept or deny packets based on their source or destination address, port number, protocol, including TCP, UDP, and ICMP, as well as monitor number of packets and bytes transferred. You can find a well-written document on implementation, and the source code at *http://www.xos.nl/linux/ipfwadm/.* Your Linux vendor should have also included the latest version with your distribution.

The Linux Firewall HOWTO is also available for some sample implementations and brief descriptions. You can find the Firewall-HOWTO at *http://sunsite.unc.edu/LDP/HOWTO/Firewall-HOWTO.html* It is intended to be used with *ipfwadm(8).*

In addition to a kernel-based packet filter for the 2.0 kernel series, there is also *IP Chains*, which is a complete rewrite of the code used in the `ipfwadm(8)` utility. The *IP Chains* package is also available in the 2.0 kernels, but only with a kernel patch. It is primarily intended to be used with the 2.1, and 2.2 stable kernels when they are released. Quoting Paul Russell, the author, *Paul.Russell@rustcorp.com.au* it was written because *The current Linux firewalling code doesn't deal with fragments, has 32-bit counters (on Intel at least), doesn't allow specification of protocols other than TCP, UDP or ICMP, can't make large changes atomically, can't specify inverse rules, has some strange quirks, and can be tough to manage (making it prone to user error).*

More information on IP Chains can be found at *http://www.adelaide.net.au/˜rustcorp/ipfwchains/HOWTO.html* which is an introduction on what it can do, and how to use it. There are also tips on converting your old `ipfwadm(8)` rules to use the new program and format.

The *sf* Firewall is an TCP/IP packet filter with quite a few features. Quoting from their web site, *"In addition to a human-readable configuration language, we implemented dynamic rules, variables and time-outs, extensive logging, alerting and counter intelligence, RIP, FTP, ICMP, IGMP, UDP and TCP filtering and offer control of all IP fields. The firewall also prevents packet address spoofing."*

It is available at *http://www.ifi.unizh.ch/ikm/SINUS/firewall.html* You can find the documentation and quite a few configuration examples at *http://www.ifi.unizh.ch/ikm/SINUS/sf-doc/*

12.5 Linux Firewall Tools

http://www.ejj.net/ sonny/fwconfig/fwconfig.html

12.6 Proxy Servers

12.7 Masquerading and Address Translation

Linux also supports masquerading of IP packets. With this ability, and using the *ipfwadm(8)* tool, all packets being forwarded through a Linux box have their source address translated to the IP address of the Linux box. On the packet's return trip, the proper address gets substituted into the translated address, and delivered to the proper host.

Not only does this provide ambiguity for the host behind the Linux box, it also has the advantage of preserving the amount of registered IP addresses that must be used, where instead only a single IP address for the Linux box is needed.

Masquerading is typically used for a home network, to allow both your Windows machine, as well as your Linux box, to use the same dialup connection. It can also be used on your firewall to provide security for the hosts behind the Linux box, and as previously stated, preserving the amount of necessary registered IP addresses in an attempt to solve the address space problem the Internet is currently suffering from.

More information can also be found in the IP-Masquerade mini-howto, available at *http://sunsite.unc.edu/LDP/HOWTO/mini/IP-Masquerade.html* and the IP Masquerading home page, available at *http://ipmasq.home.ml.org/*

Linux also implements Network Address Translation (NAT) which is another method of masquerading a number of hosts behind one IP address (called *m:1* translation), but is far more advanced. It includes *m:1* translation, (that is, it translates *m* hosts to *1* addresses), *m:n* translation, *m!=n* translation, referred to as dynamic NAT, as well as *m=n* translation, referred to as static NAT.

Michael Hasenstein, *M.Hasenstein@rocketmail.com* has written some excellent documentation, which explains each aspect very clearly, including how and why he wrote it, and is available at *http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html*

13 Writing Secure Code

The ability to audit software for existing vulnerabilities, as well as preventing them from happening in code that you write, are important qualities to develop. There is a wealth of information available on the Internet to inform you of the type of vulnerabilities, and how they are created, as well as preventing your code from becomming a future vulnerability.

13.1 Preventing `/tmp` Exploits

On Monday, March 9, 1998, Rogier Wolff *R.E.Wolff@BitWizard.nl* summarized on the *linux-security@redhat.com* mailing list the issues with an exploit that has recently been found. By being able to predict a temporary filename that will be created in the world-writable directory `/tmp`, it may be possible for a user (especially files created by the root user) to unknowingly reveal confidential information about the system. He wrote:

13.1.1 Introduction

Every now and then a new "exploit" turns up of some program that uses tmp files. The first solution was "sticky bits", but since links exist (that's a LONG time), that solution is inadequate.

13.1.2 Discussion

The problem is that you put an object (link/pipe) in the place where you expect a program to put its tempfile, and wait for another user to open the tempfile. Usually a method can be found that would allow you to gain access to rights of the user opening the tempfile.

Sometimes a program already checks for the existence of the file, and creates it if its not there. This is not atomic, and cannot easily be made secure. The standard trick is to create a symlink that you move back and forth between the "expected" file name and some "storage place". On operating systems, like HPUX and SunOS, this has a much better than 50% chance of success because they have synchronous directory updates. On operating systems that have an efficient buffer cache, like Linux, the chances are much worse. But that won't save your machine from someone gaining root access: the bad guys simply write a program to try it 100 times.

The Unix philosophy is that things that should be easy also -=are=- easy. So, a program that has setuid rights might need to be careful not to give those rights away. A non-setuid program should not have to worry about buffer overruns (you can crash the program, wow!). It should similarly not have to worry about temporary files.

13.1.3 Solutions

- Have a libc function that securely creates a file and passes a file descriptor to that file.
- Have all programs respect an environment variable $TMP that says where to put temporary files. Make the distributions set this to $HOME/tmp, and make the "create a user" script add that directory.
- A special file system (option) that doesn't allow symlinks or hardlinks to files that you don't own. This has been suggested in the past.
- A special `/tmp` directory, that's private for every userid.

The discussion continued on stating that each of these solutions may have problems, but I don't believe there is one sure-fire solution to this problem yet.

You should be sure to use temporary filenames that are not easily guessable. The `mktemp(1)` and `mkstemp(1)` program exists to create a more secure temporary file.

13.2 References

There are quite a few documents out there that describe some procedures to write secure code.

- SunWorld Online article on Designing Secure Software *http://www.sun.com/sunworldonline/swol-04-1998/swol-04-security.html*
- Adam Shostack *http://www.homeport.org/~adam/review.html* discusses some mechanisms for architecting secure software available specifically at *http://www.homeport.org/~adam/review.html*
- A presentation given by Steve Bellovin in 1996 *http://www.research.att.com/~smb/talks/odds.pdf*
- *Writing Solid Code*, written by Steve Maguire, and published by Microsoft Press, focuses on writing bug-free software, and not on security issues, but there is a larger overlap.
- Auditing existing code for vulnerabilities *http://www.pobox.com/~kragen/security-holes.html*
- Checking for race conditions in file accesses *http://seclab.cs.ucdavis.edu/papers/bd96.ps* Written by Matt Bishop.
- Attend the SANS course on writing secure code taught by Matt Bishop. It is very highly rated by those that have attended. See *http://www.sans.org* for the upcoming conference in October.
- Bishop's papers on writing secure `setuid` programs and other documents from him on writing secure code *http://seclab.cs.ucdavis.edu/~bishop/secprog.html* Some of that goes back to 1986 – Whew.
- Techniques for Trusted Software Engineering *http://seclab.cs.ucdavis.edu/~devanbu/icse98.pdf*
- *Practical UNIX & Internet Security* by O'Reilly and Associates contains a chapter on writing secure `setuid` programs.

Some of these links (Specifically the Microsoft one) was pulled from the recent bugtraq summary which can be found at *http://geek-girl.com/bugtraq/1998_3/0215.html* My first *Secure Programming* section appeared in May, '98.

13.3 Preventing Buffer Overflows

Buffer overflows are an attempt at exploiting a bug in a software program that does not allocate enough space to store data in a buffer, then using this to write past the end of the buffer, on top of other memory space, outside of its normal stack area.

To quote Aleph 1 <*aleph1@underground.org*>,

> "smash the stack" [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind.

You can find more information and the rest of this article at *http://www.2600.com/phrack/p49-14.html* titled "Smashing The Stack For Fun And Profit" written by Aleph 1.

You can find a full description of the problem at *http://l0pht.com/advisories/bufero.html* titled "How to Write Buffer Overflows".

There is a fully indexed page of this information available at *http://reality.sgi.com/nate/machines/security/*

Solar Designer, the well-known Linux kernel hacker, has several times provided secure solutions to otherwise non-secure issues. His *secure-linux* document and selection of kernel patches helps to prevent such things as:

- Non-executable user stack area, which make buffer overflows more difficult to exploit.
- Restricted links in `/tmp`, which prevents non-root users from creating hard links to files they don't own.
- Restricted pipes in `/tmp`, which disallows writing to pipes not owned by the user in directories with sticky bit set.
- Restricted `/proc` prevents non-root users from seeing statistics for processess other than their own, as well as making information about existing network connections unavailable.

His document also explains the drawbacks to using these patches. You can find the patch, and more information at *http://www.false.com/security/linux/index.html.*

StackGuard, a library and compiler technique, attempts to minimize the effects of these problems by fixing it at the library level, rather than at the individual program source level, with only a minimal performace penalty. You can find more information on this at *http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/*

14 Incident Response: Before, During, and After

Incident response must be highly organized in order to be effective. You must have a response plan developed ahead of time. There are six stages of activity in a formal incident response. This is a brief description of these stages. For a more involved description, contact CERT at *http://www.cert.org/*

14.1 Preparation

Organizations should outline the objectives for incident handling. You must determine the minimal acceptable level of security controls for systems and networks, and implement these controls. Ensuring that all systems and network components have at least a minimum level of security often prevents incidents from becoming widespread. Security response team should be available 24 hours per day.

The preparation stage of incident response should entail the installation and testing of software that will be used when an incident occurs. Intrusion detection software can be very effective, for example, as can software that verifies the integrity of programs and data, such as tripwire, or the Red Hat package manager. Waiting to obtain useful software can be a costly mistake.

You can find more information on detecting signs of intrusion at CERTs Security Improvement site *http://www.cert.org/security-improvement/modules.html*

14.2 Detection

It is important to be able to recognize suspicious signs that an incident has occurred. Becuase detection must often rely on extremely subtle signs, the use of incident-detecting software is a standard practice in systems security efforts. System accounting logs are usually a dependable source of information about possible incidents, if they are configured correctly initially. In many organizations, the system administrator is required to inspect each system's log data on a daily basis. There are also programs available that can scan log data for the most common types of exploits, and provide a summary for the administrator to track and investigate.

Every detail about a possible intrusion should be recorded. Preserving as many details as possible will help the incident response team to understand how the attack occurred and how it affected the victim system.

Most organizations now employ the use of firewall systems to increase the security of the internal networks. Examining the firewall logs can lead to intrusion attempts. There are also programs available to process firewall logs, and produce a report of possible successful and failed intrusions.

Be sure to see the *Intrusion Detection* section of this document as well.

14.3 Containment

The third stage of incident response is containment. Once you have realized there is an attack going on, you need to be sure it does not spread further to other systems, or produce further damage to your system. Spotting a security compromise under way can be a tense undertaking. How you react can have large consequences. Hasty actions can cause more harm than the attacker would have.

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 setting 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 are normally 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 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 intruders site.

If you can't deny all people from the same site as the intruder, locking the users 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 backdoors).

After you have done one of the above (disconnected 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 and get back in. Perhaps using a different account, and/or from a different network address.

The first priority of containment is to determine what is at risk if the incident spreads. If at all possible, a backup of the existing status of the machines should be made. There are several reasons for this, including keeping evidence of an attack for legal reasons, as well as keeping the data in the event the exploit deletes data.

Containing a network attack is often a matter of shutting the system down, which is in many cases, the safest response. If the system contains sensitive information, you might consider disconnecting the system from the network, booting to single user mode, or configuring the firewall to deny incoming requests.

In some cases, allowing an attacker to continue is an effective way to track the attacker's actions. Obviously, this should only be done with prior arrangements being made with the incident advisory group. Several people have tracked intruders in the past, and written their reports for others to learn from. Consider reading "UNIX Backdoors" or "Protecting Your Site By Breaking Into It".

If you are able to determine what means the attacker used to get into your system, you should try and 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 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 here *http://www.caldera.com/tech-ref/security/*. Red Hat has not yet seperated their security fixes from bugfixes, but their distribution errata is available at *http://www.redhat.com/errata* It is very likely that if one vendor has released a security update, that most other Linux vendors will as well.

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.

14.4 Eradication

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?

The fourth stage of incident response includes getting rid of the problem. Obviously in order to eradicate the problem, you need to know where the source of the problem is. Generally it is difficult to find the exact cause of the exploit.

Network intrusions are generally more difficult to eradicate, because attackers can use any system on a network to launch an attack on other addressable systems. Network-based exploits may require patches to the operating system, or routers on the network, which will take time to find and fix.

An excellent document describing what steps to take upon finding out you've been compromised is available by CERT at *http://www.cert.org/tech_tips/root_compromise.html*

14.5 Restoration

The fifth stage of intrusion detection is rebuilding, and coming back online after the exploit. Restoration entails returning a system to its normal operational status, or ensuring that the system and the data are exactly as they were before the incident occurred. Ensuring that every aspect of the system is the same as before a security incident occurred is typically a labor-intensive activity. It requires that the integrity of every file in the compromised system be examined and restored.

For this reason, administrators typically backup important data, and reinstall the operating system from CDROM. Performing an integrity check or restoring the services from a backup is only the first step. The response team should then verify the integrity of services with nonproduction data in a test environment before they are allowed to resume in production mode.

The first thing is to assess the damage. What has been compromised? If you are running an Integrity Checker like Tripwire you can make a tripwire run and it should tell you. 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 and then wiping your disk(s) and reinstalling, then restoring your user files from backups and your config files. This will insure that you have a new clean system. If you have to backup 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.

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 to, 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.)

14.6 Follow Up

The final stage of incident response is the follow-up involved in reviewing the incident that has transpired and the action taken to handle it.

This should be done to make sure not only that it won't happen again, but also to see if the procedure can be improved. Total cost of the incident in terms of employee time spent, loss of critical data, legal costs, computer time, etc, should be evaluated. This information should be compiled into a document to be used to determine what the risks of loss in the future, similiar incidents will be.

It is also highly recommended that CERT be notified, and the proper documentation be completed, in order to prevent others from being afflicted by the same attack. CERT will be more than willing to help you with your attempt at finding the intruder.

You should report the attack to the admin contact at the site where 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. If you decide to report the intrusion, which is strongly advised, you should be ready for the types of questions that will be asked. Disclosure of the information is not mandatory, and most documents allow you to specify which information can be disclosed and which cannot. To be prepared for the types of questions that will be asked, you should document the working condition of your systems, including host information, administrative contact for that network, and the type of incident (probe, scan, prank, scam, email spoofing or bombardment, sendmail attack, etc)

14.7 Additional Information

Incident response is still a relatively new subject, and has not been studied as extensively as other subjects, such as contingency planning and risk analysis. However, there is a great deal of resources available to help in such circumstances. Two organizations, CERT/CC and FIRST are well-equipted to help in such events.

You can find the appropriate documents for reporting security exploits near the end of this document.

15 Security Sources and Tools

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.

There is an Appendix section in "`Building Internet Firewalls`" that discusses some of the more popular and useful security tools. Have you gotten the hint yet that this is a book you really should purchase? :)

COAST, the Computer Operations, Audit, and Security Technology project at Purdue University is the place to go for security tools. You can find these archives at *http://www.cs.purdue.edu/coast/hotlist/* They typically are very well organized, and have a clear description of what the tool does, as well as a keyword search.

Security tools are typically categorized in several different categories. These include network and host scanners to tell you which services are available on your system and possibly any exploits associated with those services, authentication tools, analysis tools to analyize your machines and log files both before an after an attack, service filtering tools such as firewalls and service monitors, and general utilities.

Linux systems, and Linux vendors, realize the benefits of many of these programs, and as a result many of the preventitive security tools have already been incorported into your distribution. Don't use this as an excuse for not going through what is available, and making sure it is configured properly.

15.1 Network Scanners and Auditing Tools

There are a number of different software packages available that do port and service based scanning of machines or networks. SATAN and ISS are two 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 it can, and tries to determine what service is running there. Based on this information, you could find out the machine is vulnerable to a specific exploit on that server.

If when you run these network scanning and auditing tools, you find egregious security exploits, you should rethink your approach. These tools are not a one-stop security solution, and don't assume that if they don't find a problem, it doesn't exist.

15.1.1 Security Administrators Tool for Analyzing Networks (SATAN)

SATAN is a port scanner with a web interface. SATAN was written by Dan Farmer and Wietse Venema, and released in 1995. It was based on known vulnerabilities (mainly those from CERT Advisories), but hasn't really been updated since then by the original authors.

It can be configured to do light, medium, or strong checks on a machine or a network of machines. It has been known in the past to crash machines when doing a heavy scan. This isn't a bug in SATAN; rather, it's a poorly configured machine. It's a good idea to get SATAN and scan your machine or network, and fix the problems it finds. *http://www.trouble.org/~zen/satan/satan.html*

15.1.2 Security Administrator's Integrated Network Tool (SAINT)

Perhaps not as well known, but nevertheless useful, is a package called SAINT, which is also a security analyzer. Quoting from the README:

SAINT is the Security Administrator's Integrated Network Tool. In its simplest mode, it gathers as much information about remote hosts and networks as possible by examining such network services as finger, NFS, NIS, ftp and tftp, rexd, statd, and other services. The information gathered includes the presence of various network information services as well as potential security flaws – usually in the form of incorrectly setup or configured network services, well-known bugs in system or network utilities, or poor or ignorant policy decisions. It can then either report on this data or use a simple rule-based system to investigate any potential security problems. Users can then examine, query, and analyze the output with an HTML browser, such as Mosaic, Netscape, or Lynx. While the program is primarily geared towards analyzing the security implications of the results, a great deal of general network information can be gained when using the tool - network topology, network services running, types of hardware and software being used on the network, etc.

However, the real power of SAINT comes into play when used in exploratory mode. Based on the initial data collection and a user configurable ruleset, it will examine the avenues of trust and dependency and iterate further data collection runs over secondary hosts. This not only allows the user to analyze her or his own network or hosts, but also to examine the real implications inherent in network trust and services and help them make reasonably educated decisions about the security level of the systems involved.

You can find the latest version at *http://www.wwdsi.com/saint/*, although it seems to be a relatively new product, but developing rapidly. It claims to be a work derived from SATAN, and includes many enhancements.

15.1.3 Rhino9 Auditing Tool

The security research group Rhino9 have released their auditing tool There are also several tools available for Linux that allow you to:

- Kernel check (from linux.kernel.org)
- Checks for all unpasswored accounts
- List every suid program on the system
- Tell you then which suid programs are possibly vulnerable
- Check for a sniffer (PROMISC mode)
- Permissions check on some vital directories
- Inetd backdoor scanner (bind a shell to a port)
- Search the entire system for .rhosts files
- Check showmount exports
- List all current listening ports

- Tells how to disable anonymous ftp (if found)
- Trusted xhost check
- X11amp check
- Check for TCP wrapped ports
- All one text file (uuencoded)
- Now mails you the security check results

The tool seems to be still a little immature, but variation is always a good idea.

15.1.4 Internet Security Scanner (ISS) and System Security Scanner (S3)

Internet Security Systems has written two proactive security tools, both of which run on Linux. ISS has been a long-time supporter of Linux, and make some very useful tools.

The System Security Scanner allows the security professional to *"proactively seeking internal system vulnerabilities. S3 is a comprehensive host-based security assessment and intrusion detection tool which i dentifies and reports exploitable system weaknesses. S3 assesses file permissions and ownerships, network services, account setups, program authenticities, operating system configurations and common user-related security weaknesses such as guessable passw ords to determine current security levels and to identify previous system compromises. With the majority of information security breaches perpetrated by insiders, this avenue of assessment is of vital importance in assuring the protection of an organization's information."*

It is commercially supported, currently checks for 413 vulnerabilities, and available for Linux.

15.1.5 Abacus-Sentry

Abacus-Sentry is a commercial port scanner from *http://www.psionic.com.* Look at it's home page on the web for more information.

15.2 The Art of Port Scanning

Fyodor *<fyodor@dhp.com>* wrote "The Art of Port Scanning" in Volume 7, Issue 51 of Phrack Magazine, in September 1997, which discusses the various types of port scanning that can be done, and a source code program at the bottom that can be used to find out what services a host is offering, using a variety of port scanning techniques. It is available at *http://www.2600.com/phrack/p51/* or at Fyodor's site *http://www.dhp.com/~fyodor/nmap/*

15.2.1 Detecting Port Scans

There are some tools designed to alert you to probes by Satan and ISS and other scanning software, however, liberal use of TCP wrappers and making sure to 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.

You should read the *Phrack* magazine document listed in the previous section to understand the types of port scans that can be performed on your systems.

A properly configured implementation of TCP Wrappers can do a great deal towards catching an intrusion attempt, and even warn the administrator of the break-in. See the *Host Security* section for specific examples of TCP Wrappers usage.

Gabriel, a tool specially designed to detect SATAN scans and attacks, but can also detect other types of scans, probes and system attacks, can be found at COAST.

15.3 Incident Response Contacts

- Australian Computer Emergency Response Team (AUSCERT) *http://www.auscert.org.au/* mail contact *auscert@auscert.org.au* call +61 7 3365 4417
- **Computer Emergency Response Team/Coordination Center** was developed by Carnegie Mellon University and The Defense Advanced Research Projects Agency (DARPA) to handle security incidents and issue security alerts, as well as assisting other response teams to set up operations. They possess a large body of knowledge about incident response, as well as current exploits, and contacts for law enforcement. They can be reached at *http://www.cert.org* and *cert@cert.org* and by phone at (412) 268-7090.
- **The Forum of Incident Response and Security Teams** was established by the National Institute of Standards and Technology (NIST), the Department of Energy, NASA, CERT/CC, and several other agencies to provide a forum in which teams can easily communicate with each other and share collected knowledge about incident response. You can contact them at *http://www.first.org* and by email at *first@first.org* and by phone at (301) 975-3359.
- **Computer Incident Advisory Capability (CIAC)** *http://ciac.llnl.gov/* mail contact *ciac@llnl.gov* or call +1 (510) 422-8193
- **Defense Information Agency Center for Automated System Security Incident Support Team (ASSIST, for DoD sites)** *http://www.assist.mil/* mail contact *assist@assist.mil* or call +1 (412) 357-4231
- **Federal Computer Incident Response Capability (FedCIRC)** *http://fedcirc.llnl.gov/* mail contact *fedcirc@fedcirc.nist.gov* or call +1 (412) 268-6321
- **Federal Bureau of Investigation - National Computer Crime Squad**Washington Office: (703) 762-3160Boston Office: (617) 223-6056

  Depending on the City which the bureau is located, such a crime as email threats will be handled by either the Violent Crime Unit, or NCCS.
- **The German Research Network Computer Emergency Response Team (DFN-CERT)** *http://www.cert.dfn.de* mail contact *dfncert@cert.dfn.de* or call +49-40-5494-2262
- **NASA Incident Response Center (NASIRC)** *http://www-nasirc.nasa.gov/nasa/index.html* mail contact *nasirc@nasirc.nasa.gov* or call +1 (800) 762-7472
- A full list of European CERTs can be found at: *http://www.cert.dfn.de/eng/csir/europe/certs.html*

15.4 Vendor Information

There is a vendor FAQ available at *http://www.iss.net/vd/vendor.html* where you can find general vendor information. The Linux-specific contacts are as follows:

- Caldera *http://www.caldera.com/tech-ref/security/* email to *support@caldera.com*
- Red Hat *http://www.redhat.com/* mail to *redhat@redhat.com*
- Debian *http://cgi.debian.org/www-master/debian.org/sec.html* mail to *security@debian.org*
- General Linux Security *http://www.aoy.com/Linux/Security/* mail to *security-alert@redhat.com*
- Can someone please send me the correct SuSe security list address?

15.5 Mailing Lists

These are the classic security-based mailing lists available to the general public. You've probably seen the addresses for these lists a thousand times by now, but here they are again, all in one place.

Subscription requests for most lists is performed by sending "subscribe listname" in the body of the message. Be sure to keep your subscription information, so you know the proper way to unsubscribe.

- BUGTRAQ Full Disclosure List *listserv@netspace.org*
- Best Of Security *best-of-security-request@cyber.com.au*
- CERT Advisories *cert-advisory-request@cert.org*
- WWW Security *www-security@ns.rutgers.edu*
- Security List FAQ *http://www.iss.net/vd/mail.html*
- Firewalls Discussion List *majordomo@lists.gnac.com* with "subscribe firewalls" in the body of the message.
- Computer Incident Advisory Committee *majordomo@tholia.llnl.gov* with "subscribe ciac-bulletin" in the body of the message
- Linux-specific Security Issues *linux-security-request@redhat.com* with "subscribe linux-security" in the message Subject.
- Linux-centric Software Audit List *security-audit-subscribe@ferret.lmh.ox.ac.uk* and using "subscribe" in the body of the message. You can find the FAQ for this group at
- List of Security lists *http://www.faqs.org/faqs/computer-security/secmaillist/*

15.6 General References

- The COAST archive has a very large number of Unix security programs and information: *http://www.cs.purdue.edu/coast/hotlist/*
- Rootshell.com is a great site for seeing what exploits are currently being used by crackers: *http://www.rootshell.com/*
- The Linux security WWW is a good site for Linux security information: *http://www.aoy.com/Linux/Security/*
- Infilsec has a vulnerability engine that can tell you what vunerabilities affect a specific platform: *http://www.infilsec.com/vulnerabilities/*
- A good starting point for Linux Pluggable Authentication modules can be found at *http://www.kernel.org/pub/linux/libs/pam/*
- The Linux *SunSITE* archives contain quite a few contributed programs that are able to be used on Linux. You can find them at *ftp://sunsite.unc.edu:/pub/Linux/system/security*

15.7 Books - Printed Reading Material (Works Referenced)

There are a number of good security books out there. This section lists a few of them. In addition to the security specify books, security is covered in a number of other books on system administration. There really is an incredible amount of information on security available on the Internet; it's just a matter of finding it.

- *Building Internet Firewalls*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**Deborah Russell & G.T. Gangemi, Sr. 1st Edition July 1991 ISBN: 0-937175-71-4
- **Linux Network Administrator's Guide**Olaf Kirch 1st Edition January 1995 ISBN: 1-56592-087-2
- **PGP: Pretty Good Privacy**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

16 Glossary

Listed here are a few of the most common terms used most frequently, yet may not be familiar to some users. See the NET-3-HOWTO for further networking information, and the excellent 3Com Network Glossary for a great online glossary, available at *http://www.3com.com/nsc/glossary/index.htm*

There is also a security-oriented glossary available at *http://www.securityinfo.com/glossary.html* that will be useful.

- **Host** - A computer system attached to a network
- **Firewall** - A component or set of components that restricts access between a protected network and the Internet, or between other sets of networks.
- **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 strongs walls, room for extra troops, and the occasional useful tub of boiling hot oil for discouraging attackers.
- **Dual-homed Host** - A general-purpose computer system that has at least two network interfaces.
- **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). accomplish packet filtering, you set up a set of rules that specifiy 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 on to real servers, and relay answers back to clients.
- **Denial of Service** - A denial of service attack is when an attacker consumes the resources on your computer for things it was not intended to be doing, thus preventing normal use of your network resources to legimite purposes.
- **Secure Logging** - (3Com Glossary) A method whereby an audit trail of system activity is received from a bastion host and placed in a secure location.
- **Buffer Overflow** - Common coding style is never to allocate buffers "large enough" and not checking for overflows. When such buffers are overflows, 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.
- **Spoofing** - 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 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 of Phrack Magazine, available at *http://www.2600.org/phrack/p48-14.html*.
- **Authentication** - The property 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.
- **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 desire to deny ever having sent that data.
- **Set User-ID (suid) / Set Group-ID (sgid)** - Set User ID and Set Group ID files are files that everyone can execute as either it's owner or group privileges. Typically, you'll find root suid files, which means that regardless of who executes them, they obtain root permission for the period of time the program is running (or until that program intentionally relinquishes these privileges). These are the types of files that are most often attacked by intruders, because of the potential for obtaining root privileges. Buffer overflows (see glossary entry) have been a common method of attempting to obtain root permission using suid root files. You can find more information on both `setuid` and `setgid` in the *File System Security* section of this document.
- **Digital Certificate** Also called Digital IDs, is one solution to the electronic identity problem. A digital ID is a digitally-signed statement from a trusted source that attests to the identity and public key of a person, or computer, much the same way that a driver's license is an ide
- **Firewall** By this time, you must have some idea what a firewall is and does, and there is a multitude of information available on the Internet on firewalls. To briefly quote *"Building Internet Firewalls"* - "An Internet firewall is more like a moat of a medieval castle than a firewall in a modern building. It serves multiple purposes:

  - It restricts people to entering at a carefully controlled point
  - It prevents attackers from getting close to your other defenses
  - It restricts people to leaving at a carefully controlled point

  An Internet Firewall is most often installed at the point where your protected internal network connects to the Internet.

  All traffic coming from the Internet or going out from your internal network passes through the firewall. Because it does, the firewall has the opportunity to make sure that this traffic is permitted to pass through, as defined by the security policy at your site."

- **Confidentiality** - Information is accessible only to authorized users, with unauthorized individuals prevented from accessing or eavesdropping on the information.
- **Integrity** - Information traversing the network is protected from being erroneously changed by authorized or unauthorized users. This typically demands encryption mechanisms for security.

17 Frequently Asked Questions

17.1 Is Linux a secure Operating System?

For you to determine whether *you* think Linux is a secure operating system, there are a few pieces of information you should be aware of before making your decision:

- Linux, as well as other popular freely available operating systems, have thousands of people scrutinizing each line of code, not only for possible exploits, but also to further audit its level of security. Closed operating systems have only a small staff of people to determine its level of security.
- UNIX, and UNIX-like operating systems such as Linux, have an established background to work from, and despite Linux's relative youth, it is still older than many commercial operating systems.

17.2 Loadable modules versus compiled kernel?

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 himself load 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.

17.3 How can I securely use the Apache Front Page Extensions?

Answer: Well, you've taken the first step, and admitted that it has been a cause of exploit in the past. There have been several papers written on it's insecurity. Their new version, FrontPage 98 apparently hasn't gotten much better. Read this article, for an interesting overview of the issues at hand *http://www.mr.net/~fritchie/frontpage.html*

Also, the *Microsoft FrontPage Security Hell* is available here *http://www.worldgate.com/~marcs/fp/*

Instructions for installing and configuring, as well as building awareness, is available at the *FrontPage Awareness Site* available at *http://frontpage.netnation.com/*

17.4 How do I know whether my machine is secure?

Answer: Well, the obvious answer is to read and follow the procedures outlined in this document.

Assuming you have done that, and you are not currently aware of one of your machines already being exploited, and perhaps less obviously, download some of the exploits from *http://www.rootshell.com* and see if they work on *your* machine.

17.5 What are the arguments for *ipfwadm(8)*?

Answer: This information is covered in the Firewall-HOWTO, as well as in the *Firewalls and Border Patrol* section of this document. You should keep in mind that the *ipfwadm(8)* command is specific to the 2.0 release of the kernel. Version 2.2 will feature a much improved firewall, called IP Chains. You can find more information on IP Chains in the Firewalls section of this document.

17.6 Where do I find the most secure version of program XYZ?

Answer: The best thing you can do here is to check your Linux vendor's errata for any preconfigured packages in their updates archive for your current distribution.

You should also already be subscribed to one of the informational security mailing lists, or at least the announce list from your vendor, describing the procedure for finding the proper updates.

17.7 Logging in as root from a remote machine always fails.

Answer: See the section on 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. Don't forget, potential intruders have time on their side, and can run automated programs to find your password.

17.8 How do I enable shadow passwords?

How do I enable shadow passwords on my Red Hat 4.2 or 5.x system?

Answer: 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 `/usr/bin/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 are 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.

17.9 How can I enable the Apache SSL extensions?

Answer:

1.Get the latest version of SSLeay from *ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL*

2.Build and test and install it!

3.Get the latest version of the Apache source

4.Get Apache SSLeay extensions from *ftp://ftp.ox.ac.uk/pub/crypto/SSL/apache_1.2.5+ssl_1.13.tar.gz* `<ftp://ftp.ox.ac.uk/pub/crypto/SSL/apache_1.2.5+ssl_1.13.tar.gz>`

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 *http://www.replay.com* which has many pre-built packages, and is located outside of the United States.

17.10 How can I securely manipulate user accounts?

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

- The `pwconv(8)` and `unpwconv(8)` programs can be used to convert back and forth between shadow and non-shadowed passwords.
- The `pwck(1)` and `grpck(1)` programs can be used to verify proper organization of the passwd and group files.
- The programs `useradd(8)`, `usermod(8)`, and `userdel(8)` can be used to add, delete and modify user accounts. The programs groupadd, groupmod, and groupdel will do the same for groups.
- Group passwords can be created using `gpasswd(1)`.

All these programs are "shadow-aware" – if you enable shadow it will use `/etc/shadow` for password information, otherwise it won't.

See the respective man pages for further information.

17.11 How can I password protect specific HTML documents?

Answer: 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*

17.12 My Set-User-ID shell script does not work!

Any shell script set to run as root can gain unauthorized root access to ordinary users. It is therefore disabled in the kernel from operating. The details about how one can break a setuid shell script are posted regularly on Usenet. Set-User-ID programs are one of the most common methods of intrusion, especially by means of buffer overflow.

See the UNIX FAQ for more information on how this actually works.

18 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.

It is unfortunate that this document does not discuss some other issues, such as the legal ones. While this certainly affects an administrator, it does not uniquely affect a Linux system administrator. The legal issues are real ones. Luckily, there is an incredible amount of information on this topic available elsewhere already.

Some things you should always be sure to do when taking on a security project:

- Use layered security. Don't rely on all one security mechanism, such as your firewall, for securing your entire site.
- Encrypt as much data as possible. Clear text passing on the wire is a great opportunity for a cracker to intercept and use to his advantage.
- Do not rely on basic authentication. Utilize the tools that are available, including SSH, S/Key, Kerberos, as well as TCP Wrappers for host authentication, etc.
- Have someone check your work. What you may consider secure, another may see an obvious hole in your strategy. Balance of power.
- Be aware of your environment. Is syslog still working? Does it seem like there is an abnormal load on your machine?
- Keep it as simple as possible. A simple solution is far easier to keep secure than a difficult one.
- Be proactive. Staying in tune with the current events in security, and improving technology is key to protecting your network.
- Employ the "Keep It Simple, Stupid" Methodology (KISS) at first
- Multiple gateways is bound to be less secure than trying to manage just one. Make sure you know where all your points of entry are.
- Install only network services required by your users, and only after evaluation of their potential for security problems.
- Perform regular scans of the status of your network. Use the freely available tools, such as SATAN, ISS, and CRACK to check your systems integrity.
- Relax. Check out the Linux Penguin's Page *http://www.vni.net/~kwelch/penguins/*

Keeping up-to-date with the flood of security topics can be an overwhelming task. Take a piece at a time, and prioritize what needs to be done. Choosing the obvious holes to fix first is a good start.

Remember, just because you have all the latest software updates installed, does not mean your machines are secure. There will always be new software exploits, as well as uneducated users who choose poor passwords. Continual inspection and attentiveness is required.

19 Thanks To

Special thanks to Antonomasia <*ant@notatla.demon.co.uk*> for repeated reviewing of this document, as well as a great source of information for bouncing off of ideas, general improvements, and suggestions.

I also appreciate all the work the developers have put into their security area of expertise, and providing us with an outstanding alternative to the expensive, proprietary solutions we would otherwise have to endure.