

Welcome to UNIX Security

Computer security is a concept that is easy to grasp but difficult to achieve. Each day our world grows smaller via the electronic highways of the Internet. No one knows the activity that is taking place via those electronic connections. Some are gathering information for honest reasons and others for dishonest. Some are viewing art while others are surfing porn. Some are querying Microsoft as to how to fix a problem while still others are working hard at exploiting those problems for fraudulent gain. Security has become everyone's concern. We all have personal or financial information that has been placed on the Internet and which could conceivably be compromised by those who would use it for their own personal gain.

Those of us associated with the military can understand the other types of compromises, the kinds that put military personnel at risk or jeopardize military operations. No other nation relies on computers quite like America. Our military has a network of enormous size. It would take a large portion of the U.N. member nation's computer systems to equal what America has in its military alone. It is highly unlikely that an average U.S. military employee or service member knows what it takes to secure their computer network. The responsibility of doing that rests with the system administrator and network manager. Every single system administrator guards a doorway that our enemies seek to use. The information highway brings the battlefield to the doorstep of each and every LAN system that DoD operates.

Everytime we fire up a computer or network, we are inviting our enemies to find our vulnerabilities and exploit them. The information age created numerous new risks that 20 years ago didn't exist. Our computer networks fall prey to sniffing, spoofing, hacking, viruses, Trojan horses, ActiveX, CGI scripts, denial of service attacks, and buffer overflows. It takes knowledge to break into a computer network and it takes knowledge to defend against it.

The next couple of days will be spent examining some of the necessary knowledge required to secure a UNIX server. UNIX is not the most numerous or common operating system in the military's many networks, but it does provide the operating platform for some of the most important systems. Many of our tactical and strategic systems rely on UNIX as their core operating system. UNIX systems are often the ultimate target of our enemies. The systems targeted often host vast databases with critical information that our enemies want.

UNIX system administrators are not born, they are created. It requires a lot of study and dedication to even attempt to learn the large number of commands and command variants employed in UNIX. We are not going to attempt to do in a few days what takes years to accomplish. We are also not going to expect that a few days will adequately train you to defend a UNIX server attached to a military network. What we do want to accomplish is to show you that you can take a set of security procedures, an unfamiliar operating system, and still be successful.

We are going to look at a few of the security issues you will face as a UNIX system administrator and show you how to incorporate fixes. Some of the areas we'll address are:

1. UNIX commands
2. Physical security
3. Installation and initial configuration
4. Good security practices
5. Network security issues
6. Auditing
7. Available tools
8. forensics

You will be doing as many practical exercises as time permits. These P.E.s address all of the following areas:

1. Types of accounts
2. Operating System run levels
3. Key files
4. Password mechanics and policy
5. Permission (mode) settings
6. User Mode Mask (umask)
7. Detecting hidden files and directories
8. Controlling login
9. Bench marking our file system
10. Identifying files with privileged access
11. Setting system and user umask settings
12. File Access Control Lists
13. Setting password length and encryption requirements
14. Banners, core dumps, buffer overflows, and secure mounting
15. Patch and program installation
16. Identifying network services
17. Disconnecting unwanted connections
18. Capturing telnet and secure shell transmissions
19. Scheduling events to occur on a certain time or day
20. Xwindows security issues
21. Shutting down network services and ports
22. Disabling trusts
23. Network File System (NFS), Sendmail, and FTP security settings
24. File sharing vulnerabilities
25. Trojan horse programs
26. Setting up the required logging features
27. Vulnerability scanning
28. Process tracking
29. Role Based Access Control
30. Password testing
31. Forensics
32. Auditing and Monitoring

If this is your first time using UNIX, be patient, it will start to make sense. Perhaps in trying to understand UNIX you may gain a better understanding of Windows. If you aren't a current UNIX system administrator, cheer up, you could be one later.

UNIX Quiz

(Circle the correct answer)

1. T F We change the name of the root account to stop hacker brute force attacks.
2. T F System accounts are logged into like normal accounts but are used as Administrator accounts.
3. T F UNIX has no logging enabled by default.
4. T F Both the root administrator and regular users can establish trusts with other computers and users.
5. T F UNIX by default, requires the use of uppercase, lowercase, symbols and numbers in password construction.
6. T F Viruses cause a lot of problems for UNIX administrators.
7. T F If a user could run the chown command, a user could assign someone else ownership of a file.
8. T F By enabling process accounting, you will log all the commands that are run and by whom they were run.
9. T F Unnecessary services are turned off in the /etc/services file.
10. T F The /etc/ftpusers file contains the list of accounts that can utilize ftp.

UNIX Command Reference List

Command words and arguments are shown exactly as you type them. Note that most are in lowercase. They must be typed that way: UNIX treats uppercase letters as different from their lowercase counterparts. Type punctuation exactly as shown, such as hyphens or vertical bars. Control characters, typed by holding down the CTRL key while typing the character, are shown by the notation `^x`, where `x` is the character. You must finish all UNIX commands by pressing the RETURN key, which is not illustrated here. Most commands have an assortment of flags/options that are not shown. Use the man page to learn about program flags/options

Connecting from another computer

If you are using another computer on the network or on the Internet, and if that computer has a telnet command, you can use it to connect to a UNIX host. If the other computer is also running UNIX, you can connect using rlogin. If your computer offers ssh, the Secure SHell, you can also connect using ssh or slogin.

Special Characters

- ^c** Interrupts and aborts execution of the current process, which cannot be restarted.
- ^z** Suspends a process (temporarily stops it by putting it in the "background") so that you can give other UNIX commands. To resume the process, use fg (foreground) or %n, where n is the background job number. To disable ^z (make it undefined), give the command:
- ^D** Ends a telnet session
- ^]** Used to escape out of some commands or processes. Similar to ^C.
- >** Writes to the file named to the right of this symbol instead of to standard output. Thus:

```
cat myfile
```

displays contents of *myfile* at your terminal (standard output), but:

```
cat myfile > listout
```

copies it to file *listout* instead, overwriting and destroying any previous contents of the file *listout*.
- >>** Appends output to the file named to the right of this symbol. For example:

```
cat nextfile >> listout
```

appends a copy of *nextfile* to file *listout* instead of overwriting what is already in *listout*.

| Creates a "pipe" so that output from one program (on left) becomes input to the next one (on right). For example:

```
ls | lpr
```

sends output from the ls (listing) program to the lpr (printing) program.

- * Matches zero or more characters in a filename. For example, A* matches A, AbC, ARGUE, African, etc.
- ? Matches any single character in a filename. For example, A?C matches AAC, ABC, ACC, etc.
- \ Prevents the following character from being interpreted as special. For example, * makes * an ordinary character.

Getting Online Help

Online UNIX documentation consists of "man" (manual) pages that you can display or print by using the man program. For example, to display the man page for the rm (remove) program, type:

```
man rm
```

The command

```
apropos keyword
```

is used to obtain a list of man pages that contain *keyword* in their title.

Files and Directories

A UNIX filename or directory name can contain up to 255 letters, digits, and punctuation characters (best limited to period and underscore). To list names of files in a directory, type:

```
ls
```

This will not list filenames that begin with a period, unless you include the -a (all) option, like this:

```
ls -a
```

Other ls options include:

- l long form, gives file protections, size in bytes.
- F marks directory names with a slash, executable files with an asterisk.
- b forces the printing of nonprintable characters to an octal format.

Deleting a File

Use `rm` (remove) to delete files from your directory. For example:

```
rm trashfile
```

The `-i` (interactive) option to `rm` causes the system to request confirmation of each file deletion. This option may prevent accidental deletions and is useful if you are removing several files, as in:

```
rm -i *file
```

Summary of Useful Commands

The following commands are grouped alphabetically by function. Unless noted otherwise, each has a man page describing it in full. Commands marked "C shell only" are described in the `csch` man.

Access Control

<code>admintool</code>	create, assign, modify groups, users, printers, etc.
<code>exit</code>	terminate a shell (see "man sh" or "man csh")
<code>getfacl</code>	read file access control lists
<code>id</code>	identify the UID, GID, EUID, EGID
<code>logout</code>	sign off; end session (C shell and bash shell only; no man page)
<code>passwd</code>	change login password
<code>rlogin</code>	log in remotely to another UNIX system
<code>setfacl</code>	set file access control list
<code>slogin</code>	secure version of <code>rlogin</code>
<code>su</code>	switch user/super user
<code>usermod</code>	modify user account information

Communications

<code>mail</code>	send and receive mail
<code>wall</code>	send message to all users
<code>talk</code>	talk to another logged-in user (full screen)
<code>write</code>	write to another logged-in user

Programming Tools

<code>awk</code>	pattern scanning and processing language
<code>cc</code>	C compiler (xlc on ADS)
<code>crontab</code>	maintain periodic tasks
<code>csh</code>	C shell command interpreter
<code>kill</code>	kill a process
<code>make</code>	manage multipart program projects
<code>nice</code>	run command at low priority (see "man nice" or "man csh")
<code>nohup</code>	run a command immune to hangups
<code>sh</code>	Bourne shell command interpreter

Documentation

apropos	locate commands by keyword lookup
man	find manual information about commands
whatis	describe what a command is
whereis	locate source, binary, or man page for a program

Editors

emacs	screen-oriented text editor
ex	line-oriented text editor
sed	stream-oriented text editor
vi	full-screen text editor
dtpad	text editor within CDE
textedit	text editor within open windows

File and Directory Management

cd	change working directory
chgrp	change group owner
chmod	change the protection of a file or directory
chown	change owner
cmp	compare two files
compress	compress a file
cp	copy files
crypt	encrypt/decrypt files (not on ADS)
cut	display specific file information
diff	compare the contents of two ASCII files (sdiff for side by side)
find	look for files by name
grep	search a file for a pattern
ln	make a link to a file
ls	list the contents of a directory
mkdir	create a directory
mv	move or rename files and directories
pwd	show the full pathname of your working directory
rm	delete (remove) files
rmdir	delete (remove) directories
sort	sort or merge files
touch	create file
umask	change default file protections
wc	count lines, words, and characters in a file

File Display and Printing

cat	show the contents of a file; concatenate files
lpr	print a file
lprm	remove jobs from the printer spooling queue
more	display a file, one screen at a time
page	like "more", but prints screens top to bottom
tail	show the last part of a file
zcat	display a compressed file

File Transfer

ftp	transfer files between network hosts
rcp	transfer files between networked UNIX hosts
scp	secure version of rcp

Miscellaneous

alias	define synonym commands
chsh	change default login shell
clear	clear terminal screen
echo	echo arguments
setenv	set an environment variable (C shell only)
stty	set terminal options

Networks

netstat	show network status (on UTS, /usr/sbin/netstat)
rlogin	login remotely on another UNIX system
slogin	secure version of rlogin
rsh	run shell or command on another UNIX system
ssh	secure-shell version of rsh
telnet	run Telnet to log in to remote host

Process Control

(The following commands function under the C shell, bash, and ksh. They do not have separate man pages.)

bg	put suspended process into background
fg	bring process into foreground
jobs	list processes
^y	suspend process at next input request
^z	suspend current process

Status Information

acctcom	read processing accounting data
date	show date and time
df	summarize free disk space
dmesg	read message log
du	summarize disk space used
env	display current environment or run programs under modified environment
finger	look up user information
history only)	list previously issued commands (C shell, bash, and ksh
last	indicate last login of users
lastcomm	read processing accounting data
lpq	examine spool queue
ps	show process status
pwd	display full pathname of working directory
set	set shell variables (C shell, bash, and ksh only)
stty	set terminal options
w	show who is on system, what command each job is
executing	
who	show who is logged onto the system
whois	Internet user name directory service

UNIX Security

If you were to compare a properly secured UNIX system to an improperly secured UNIX system, what would you find? Most likely you would find that the secure system has an administrator that is both motivated and knowledgeable of the UNIX operating system. The unsecure system probably has an administrator that, while possibly motivated, lacks knowledge. Knowledge for all its advantages, is fleeting. What you know to be true today is often changed by events tomorrow. Information Assurance is a rapidly changing environment that demands a continuing effort by administrators to reeducate themselves. Programs and operating systems are changing every day. The more complex or user friendly a system or program becomes, the more likely a backdoor exists that beckons to the hacker. UNIX security centers around its major weakness: the superuser. Anyone who can become the UNIX superuser can take over the system. Root access remains the single point of attack on a UNIX system. UNIX was not designed to be secure. It has been around a long time and does have the necessary components to enable you to make it secure. UNIX systems might not be secure when distributed. If you purchased an approved version in accordance with the Common Criteria Listings and follow the steps outlined in the UNIX Security Technical Implementation Guide (STIG), you will be able to adequately secure your system.

Machine and Site Preparation

Security is not only about securing the computer. There are certain fundamental requirements and procedures that go hand in hand with securing the UNIX operating system.

Written Security Policy

Only by performing proper analysis and security planning can you hope to provide a secure environment for your data. Implementing security without a security policy is like going on a long journey without a road map.

1. Asset identification and evaluation
2. Threat identification and assessment
3. Vulnerability and exposures identification and assessment
4. Determine access and control requirements
5. Qualitative and quantitative risk assessment
6. Develop your plan and policies to maintain number 1 while defending against numbers 2, 3, 4, and 5.

Access Control

Access control has several objectives: Identification, Authentication, Authorization, Confidentiality, Integrity, Availability, and Accountability.

Before a user gains access to an access controlled object, the user must go through three levels of access control:

1. The user must be identified
2. The user must be authenticated based on the identification supplied

3. Once identified and authenticated, the user must be authorized for access to the object under control.

Authentication Methods

Authentication is the transfer of some information that proves you are who you say you are. Failure to properly identify users results in a breakdown of authorization and access control.

There are 3 types of authentication:

1. Something you know (password)
2. Something you have (token, smart cards, ID badges)
3. Something you are (biometrics)

UNIX Physical Security

Only the UNIX system administrator needs access to the UNIX server. Unrestricted access is the same as no security

Access to the UNIX server allows:

1. Denial of Service attack (pull the plug)
2. Disaster if the root password has been compromised.
3. Possible reboot utilizing removable media that compromises root
4. Theft of hardware or data

Common sense rules the day, if you can't get in the room, you can't affect what's in it. Or can you? Protect the circuit path as you would the machine. Protect your power sources and any other building feature that would affect operation (i.e. fire sprinklers).

Make physical security redundant, just like in yesteryear where they used moats and drawbridges and high walls to protect a castle.

Limit the Hacker's options:

1. If the keyboard, mouse, or monitor is not necessary full time, consider operating without them.
2. Physically lock the computer case
3. Adjust the boot sequence so that removable media is not bootable
4. Password protect your CMOS (1386)
5. Password protect Open Boot (3 levels - none, command, full):
6. On non-PC systems, Open Boot allows for booting the system from almost any external hard drive, SCSI device or CDROM. Various commands during the boot up sequence allows for changing of environmental variables. Secure the open boot process.
7. Maintain a list of personnel authorized entry
8. Establish key/access control.

Back up your data

Recovery preparedness is the single most important factor in recovery success.

1. What can go wrong:
 - A. Natural disasters
 - B. Man-made disasters
 - C. Inside utility failures (AC, UPS, etc.)
 - D. Hardware failures
 - E. UNIX administrator error
 - F. Documentation error
 - G. Programmer or user error
 - H. Sabotage

2. Some types of backups:
 - A. Full (regular basis) copy of the entire file system
 - B. Incremental (regular basis) copy of only those files that have been modified since the last full backup.
 - C. Day Zero (one time) copy of the original system as it was first installed

3. Secure your backups as you would your actual system. Your backups contain sensitive data. Since backup tapes contain all of your sensitive information from your system, to include user data, and passwords, they become a target for anyone who has physical access to your system.

4. Minimally, system backup and recovery procedures address the following:
 - A. Detailed backup procedures
 - D. Schedule for various backups
 - E. Update your backups whenever you update or change your system.
 - F. Ensure that everything on your system is addressed in your backup plan.
 - G. DO NOT reuse a backup tape too many times because it will eventually fail.
 - H. Restore a few files from your backup tapes on a regular basis. This ensures that you have good a backup & tapes.
 - I. Rebuild your system from a set of backup tapes to be certain that your backup procedures are complete.
 - J. Keep your backup tapes under lock and key.
 - K. Keep written records of key backup and system configuration information.
 - L. Store back-up tapes off-site whenever possible.
 - M. Encrypt back-up tapes whenever possible.

5. Keep detailed notes on:
 - A. Disk partitioning (/etc/device.tab lists your partition table and mount points)
 - B. Filesystem customizations
 - C. OS version and install options

- D. SCSI device addresses
- E. Patches
- F. Configuration Management
- G. Contingency Planning

Contingency Planning

If Information Technology operations are disrupted, mission critical functions could be lost.

1. Minimize the impact of fire, flood, civil disorder, natural disaster, or bomb threat.
2. Identify an alternate site containing compatible equipment.
3. Identify backup procedures if the primary IT operation site is disrupted.
4. Destruction or safe guarding plan in case of site evacuation. (classified sites)
5. Plan the test - test the plan.

Installation and Configuration

Securing a UNIX server starts with installation. Security should be involved in every phase of installation and system use.

Some things that should be part of a secure installation process

1. Do not be connected to the network.
2. Install the OS w/Patches
3. Secure the inetd
4. Secure the startup scripts
5. Enable logging
6. Protect against buffer overflows, root network access, system account access, sendmail, motd misuse, permissions, trusts
7. Install ssh, secure shell

A minimum install will increase security by not loading various applications.

Best Security Practices

Protect the root account

There are many methods by which root access is gained. You can log on as root. You can use the su command to become the superuser. You can use the sudo command to run some commands as the superuser. You can run SUID programs which are predesignated to run as root. You can have RBAC (role based access control) accounts that allow you to perform certain root functions. These are the honest ways to become root. Hackers utilize still more ways to gain root access.

Using su and sudo are preferred when performing administrator functions. On some systems, the superuser can not change read only filesystems, unmount a filesystem with open files, write directly to a directory or create a hard link to a directory, or decrypt passwords stored in the shadow file.

Things root can do to keep root privilege safe:

1. Secure the terminal against network root access
2. Create a working group for su access
3. Do not surf the web as root

4. Do not install unapproved software as root
5. Do not test software or programs as root
6. Do not cd around the file system, stay home
7. Create a home directory for root that does not compromise the directory structure (i.e. /root instead of /)
8. Use only full path names for commands and keep "." and ".." symbols out of path statement.
9. Do not do email as root
10. Do not operate in the GUI as root. Use single user mode.
11. Reduce the number of SUID and SGID programs
12. Use su and sudo to limit use of the root login
13. Do not run programs from directories that are group or world writeable.
14. Restrict the number of people that know the root password
15. Do not use the root account for activities that can be done with a regular account.
16. Make sure that root log files and cronjobs do not source any files not owned by root.
17. Restrict access to privileged groups

Least Privilege

Users should only have the access they require to perform their mission.

1. Users do not need elevated access to perform normal job functions
2. Applications should not run with more access than they require
3. Developers do not need administrator access
4. Keep the number of root level accounts to a minimum
5. Users should be assigned to groups that are in keeping with their level of access.

Patches

Failure to stay current with available vendor patches may leave your system vulnerable to attack. Always check with ACERT for the latest approved patches for your version of UNIX.

Use the command `#showrev -p` to list patches installed on your system.

1. Retrieve the latest patch list from your vendor (NOTE: Check ACERT advisories to be sure you have the latest "approved" patches).
2. Install patches that are recommended for your system. Some patches may re-enable default configurations. For this reason, it is important to recheck security settings AFTER installing any new patches or packages.
3. Always review the Readme files.

NOTE: Ensure that current vulnerability patches are loaded as provided by the vendor and listed in ACERT/CC advisories.

umask

In UNIX, system calls have base permissions (sometimes referred to as "default permissions") which are assigned to newly created files and directories. For directories the base permissions are (octal) 777 (rwxrwxrwx), and for files they are

666 (rw-rw-rw-). Before creating the file or directory, the base permissions are compared to a mask (the umask set by the umask command) that will "mask out" permission bits to determine the final permissions for the newly created object. To determine the effective permissions, take the binary of the base permissions and perform a logical AND operation on the ones-complement representation of the binary umask.

Here is an example of creating a file with a umask of 022. The binary representation of octal 022 is 000010010. The ones-complement simply inverts the numbers -- changes all zeros to ones and all ones to zeros, resulting in 111101101. Now, performing a logical AND with the base permissions of 666 (binary 110110110) results in 644 (binary 110100100), as illustrated here:

```
110 110 110   base permissions of 666
111 101 101   ones-complement of a umask of 022
-----
110 100 100   perform logical AND, (1 AND 1 = 1, everything else = 0)
                This converts to octal 644 which is rw-r--r--
```

With the logical AND operation, both bits must be 1 (on) for the result to be 1. Since the base permissions for files has the execute bit set to 0 (off), it is impossible to create files with the execute bit turned on. Some programs may seem to create files with execute permissions turned on, but they are actually changed after the file is created.

Instead of performing the binary operation above, you can think of the umask as "masking out" the default permissions. If the position in the umask is occupied by a "one", then whatever is in the default permission is "masked-out" of the resulting effective permission. For the previous example, you would do the following:

```
110 110 110 (666 Base permissions)
000 010 010 (022 umask value)
   X   X   (mask out wherever there is a "one")
110 100 100 (644 effective permissions of the file)
```

chmod

'chmod' (change mode) is the UNIX/Linux method of changing file permissions. Where Windows/DOS machines have one set of file permissions: Read/Write - Archive - System - Hidden and then add on User Permissions to the files and directories; *NIX breaks the permissions into three groups, user, group, & other.

When you run `ls -la` you see lines similar to this:

```
drwxr-xr-x 10 root  root 1024 Oct 20 19:56 .
drwxr-xr-x  3 root  root 1024 Sep  5 22:56 ..
drwxr-xr-x  3 root  root 1029 Oct  5 22:56 dirstuff
drwxr-xr-x  4 root  root 1029 Oct  5 22:56 dirstuff1
-rw-r--r--  5 root  root 1024 Sep  5 22:56 filestuff
```

With the exception of the first letter on each line (which indicates the file type) all the other letters in the first column of each line are the file permissions. Note: to *NIX, directories are just special files. In order to allow someone to 'traverse' the

directory tree, the user must have eXecute permissions on the directory even if they have read/write privileges.

Within each set of permissions (user, group, other) there are three permissions you can set: Read - Write - eXecute. Therefore, when you set the permissions on a file you must take into account 'who' needs access. User is sometimes referred to as owner. Other is sometimes referred to as world.

Here's a stripped down list of the options chmod takes: (for more info execute man chmod at the command line.)

```
chmod [-R] # # # <filename or directory>
```

-R is optional and when used with directories will traverse all the sub-directories of the target directory changing ALL the permissions to # # #. This is a very useful option, but must be used with extreme caution.

The #'s can be:

- 0 = Nothing
- 1 = Execute
- 2 = Write
- 3 = Execute & Write (2 + 1)
- 4 = Read
- 5 = Execute & Read (4 + 1)
- 6 = Read & Write (4 + 2)
- 7 = Execute & Read & Write (4 + 2 + 1)

Of course, you need a file name or target directory. Wild cards * and ? are acceptable. If you don't supply the -R, with the target directory, only the directory itself will be changed.

You must supply the #'s in a set of three numbers (user, group, other).

To make a file readable and writable by you, only readable for your group, and provide no access for the world, execute:

```
chmod 640 filename
```

The result would look something like:

```
-rw-r----- 9 jones user 1002 Jun 445 20:50 filename
```

SetUID (SUID)

A program that has the set-user-ID bit set will have permissions similar to:

```
- rws rwx rwx
```

Notice that the 'user' permission is rws. The s in the normal position for the x tells us that the suid bit is set. The fact that the s is lowercase also tells us the x permission still exists. If the s were an uppercase S, the x permission is gone.

When you run a program that has the suid bit set, you run the program as if you were the owner. It gives programs more privileges. An suid program gives the

person who runs it the Effective User ID (EUID) of the owner. It can also let underprivileged users function as a privileged user. It could possibly let users operate as a superuser.

The `etc/passwd` file is a good example. Root owns the file. When you change your password, you need to be able to change your password data in that file. Regular users can not modify a file owned by root unless they have write permission. The `suid` allows you to temporarily function as root. When you type `passwd`, you invoke the `/bin/passwd` program and it is both a `suid` and a `sgid` program (`-r-s r-s r-x`). You are root for as long as the program is running.

Many UNIX flavors package a large amount of SUID programs in the OS. Maybe half are necessary as usually they are functions that only root performs.

SetGID (SGID)

A program that has the set-group-ID bit set will have permissions similar to:

```
- rwx rws rwx
```

Notice that the 'group' permission is `rws`. The `s` in the normal position of the `x` means the `sgid` is set and `x` permission still exists. If the `s` were uppercase `S`, the `x` permission would be missing. The `sgid` operates similar to the `suid`, but will operate with the effective group ID (privileges) of the file's group ownership instead of the owner's permissions. `sgid` lets you run the program as if you were a member of the owner's group.

If the SGID bit is set and the group execute bit is not, the "s" will be an "S". Systems that employ mandatory file locking will show an "l" instead of "S". This prohibits execution by anyone but the owner.

Sticky Bit

Directories with the sticky bit set will have permissions similar to:

```
d rwx rwx rwt
```

Notice that the 'other' execute bit is set to 't' instead of an 'x'. The lower case `t` tells us the sticky bit is set and the `x` is still present. An uppercase `T` would tell us the sticky bit is present but the `x` is not.

If the sticky bit is set on a directory, files may only be deleted by the owner, the owner of the directory, or root. It is a feature designed for directories that are world-writable where it may not be desired to allow any user to delete files at will (`/temp` or `/tmp` directories are examples).

Make sure that `/tmp` directories are owned by root. Set permissions to `1777`. This sets the sticky bit and allows for all users to fully utilize the temp directory. (`chmod 1777 /tmp`)

Core Dump Files

Core dump files are generally world-readable and often contain sensitive information including password hashes dumped from the `/etc/shadow` file.

The operating system writes out a core image of a process when it is ungracefully terminated. The core image is called core and is written in the process's working directory (Note: this is critical. Each terminated process, such as telnet or ftp, can create separate core images). `coreadm` is the command used to specify the name and location of core files produced by abnormally-terminated processes.

Consider disabling the creation of core dump files. Core images are created when a process abnormally terminates. The `ulimit` command can be used to place limits on the user. You can limit the user's core file size, data segment size, maximum amount of CPU time, maximum number of open files, and more.

1. To prevent the creation of core files by users, add the entry
`ulimit -c 0`
to their `.login`, `.cshrc`, or `.profile`.
2. To prevent system daemons from creating core files, add the entry
`ulimit -c 0`
to the appropriate boot script (usually in `/etc/rc*.d`)
3. To prevent ANY process from creating a core dump, add the entry
`set sys:coredumpsize = 0`
to the `/etc/system` file. This may not work on i386 versions of UNIX.

Buffer Overflow

A buffer overflow condition occurs when a user or process attempts to place more data into a buffer than was originally allocated. This would normally cause a segmentation violation to occur, however, this type of behavior can be exploited to gain access, often root, to the target system.

1. PREVENT the execution of arbitrary code in the data buffer. For Solaris, to prevent the execution of instructions in the data stack, add the following entries to the `/etc/system` file: *
`set noexec_user_stack=1`
`set noexec_user_stacklog=1`
Restart the system by typing: `init 6`
2. Software patches are your best defense against buffer overflows and poor programming.

*Stack settings do not work on the X86 architecture (PC's running UNIX).

Secure Terminals

By default, on most UNIX systems, root may log on from any terminal. This allows the root account to be accessed from anywhere in the network. A malicious attacker can target the root account to compromise your system.

Disable network login for root. All unencrypted root account access must take place on the physical console. The files to check may be called /etc/ttys, /etc/default/login, /etc/securetty, or /etc/security.

In the /etc/default/login file, add: `CONSOLE=/dev/console`

Lock Workstations

If a user leaves their workstation unattended and does not lock it, unauthorized personnel could access the workstation and, possibly, the network.

Lock your workstation whenever you walk away.

Right-click the display background and select "Lock Display" from the menu pop-up box.

Some UNIX GUI's will display an icon of a "lock" on the menu bar. Clicking on this icon will automatically lock the workstation. The password to unlock the workstation is your login password.

Screen Saver Passwords

If a user leaves their workstation unattended and the screen saver is not password protected, unauthorized personnel could access the workstation and the associated network.

Implement a Screen Saver Password:

Open up your desktop controls properties section of the GUI and set the desired time for the screen to initiate the lock. There is normally a check box that needs to be checked to turn on the feature.

Set the screensaver to execute no later than 10 minutes after last activity.

Special Accounts

Special and optional accounts have caused numerous problems for system administrators. Access to these accounts gives attackers elevated privileges, which will ultimately be used to gain root access. These accounts should be disabled whenever possible. Ensure that there are no shared accounts other than root in accordance with site security policy, i.e. more than one person should not know the password to an account.

NOTE: Some systems utilize Role-Based Access Control (RBAC) to break down administrator functions to specific tasks. Accounts are then created with only the necessary privileges required for completion of the particular task.

Banners

Security warning banners notify users that they may be monitored. Lack of a banner may potentially give the impression that this system may be hacked without repercussion. A banner should be provided for both the desktop (GUI) login and the command line/network login. Individual banners may also be set up for individual network services.

1. A banner should tell the visitor who the system belongs to, that the system is being monitored, that their use of the system is agreement to their being monitored, and the regulations governing the monitoring or use of the system.
2. Create the file /etc/issue. This will serve as the standard default banner for all command line or network service access. Some systems use a file called sysbanner.
3. The /etc/default/telnetd, /etc/default/ftpd, and tcp wrappers may be used for the corresponding network service.
4. For systems using the Dtlogin login screen, modify the /usr/dt/config/Xresources file. Set the Dtlogin*greeting.labelString to the Attention Banner.

Do not use the MOTD or any program that produces a banner after the user has negotiated the login sequence.

Banner example:

```
"ATTENTION!  
THIS IS A DOD COMPUTER SYSTEM. BEFORE PROCESSING CLASSIFIED  
INFORMATION, CHECK THE SECURITY ACCREDITATION LEVEL OF THIS  
SYSTEM. DO NOT PROCESS, STORE, OR TRANSMIT INFORMATION  
CLASSIFIED ABOVE THE ACCREDITATION LEVEL OF THIS SYSTEM. THIS  
COMPUTER SYSTEM, INCLUDING ALL RELATED EQUIPMENT, NETWORKS,  
AND NETWORK DEVICES (INCLUDES INTERNET ACCESS) ARE PROVIDED  
ONLY FOR AUTHORIZED U.S. GOVERNMENT USE. DOD COMPUTER  
SYSTEMS MAY BE MONITORED FOR ALL LAWFUL PURPOSES, INCLUDING  
TO ENSURE THEIR USE IS AUTHORIZED, FOR MANAGEMENT OF THE  
SYSTEM, TO FACILITATE PROTECTION AGAINST UNAUTHORIZED  
ACCESS, AND TO VERIFY SECURITY PROCEDURES, SURVIVABILITY, AND  
OPERATIONAL SECURITY. MONITORING INCLUDES, BUT IS NOT LIMITED  
TO, ACTIVE ATTACKS BY AUTHORIZED DOD ENTITIES TO TEST OR  
VERIFY THE SECURITY OF THIS SYSTEM. DURING MONITORING,  
INFORMATION MAY BE EXAMINED, RECORDED, COPIED, AND USED FOR  
AUTHORIZED PURPOSES. ALL INFORMATION, INCLUDING PERSONAL  
INFORMATION, PLACED ON OR SENT OVER THIS SYSTEM MAY BE  
MONITORED. USE OF THIS DOD COMPUTER SYSTEM, AUTHORIZED OR  
UNAUTHORIZED, CONSTITUTES CONSENT TO MONITORING.  
UNAUTHORIZED USE OF THIS DOD COMPUTER SYSTEM MAY SUBJECT  
YOU TO CRIMINAL PROSECUTION. EVIDENCE OF UNAUTHORIZED USE  
COLLECTED DURING MONITORING MAY BE USED FOR ADMINISTRATIVE,  
CRIMINAL, OR OTHER ADVERSE ACTION. USE OF THIS SYSTEM  
CONSTITUTES CONSENT TO MONITORING FOR ALL LAWFUL PURPOSES"
```

* See AR 380-53 and AR 380-5 for banner procedures.

UNIX Network Security

Turn off unnecessary services

Unnecessary services leave a system open to attack in a variety of ways. A system should only offer services that are necessary for mission performance.

Some services often running by default are:

Port	Protocol	Service	Comment
7	tcp	echo	generally unnecessary
11	tcp	systat	unnecessary and ill advised
13	tcp	daytime	generally unnecessary
19	tcp	chargen	generally unnecessary
21	tcp	ftp	necessary only for ftp
23	tcp	telnet	necessary only for telnet
25	tcp	smtp	necessary only for incoming mail
37	tcp	time	generally unnecessary
42	tcp	nameserver	generally unnecessary
43	tcp	whois	generally unnecessary
53	tcp/udp	DNS lookup/zone	necessary only on a dns server
69	udp	tftp	generally unnecessary and dangerous
79	tcp	finger	generally unnecessary and dangerous
80	tcp	http	necessary for internet browsing
111	tcp	portmapper/rpcbind	necessary for RPC services
512	tcp	rexec	potentially dangerous
513	tcp	rlogin	potentially dangerous
514	tcp	rsh	potentially dangerous

/etc/inetd.conf

The inetd.conf file is the configuration file for the internet daemon (inetd). This is known as the super daemon. It sits in memory and waits for network connection requests. Any service that is enabled in its configuration file will be serviced. Any line not preceded by a '#' symbol is a service that is running. The first word on each line is the name of the service; the second column, the type of socket; the third column, the protocol; the fourth column, the wait status; the fifth column, which account owns the running process; the sixth column, the full path to the actual daemon program; and the last column allows for command arguments or parameters.

Disable any services that you do not need. To do this, comment out services by placing a '#' at the beginning of each line. Then enable the ones you NEED by removing the '#' from the beginning of the line. For changes to take effect, you need to restart the inetd process. If you do not need ANY services running, prevent UNIX from launching the inetd by commenting out the last line of the S72inetsvc startup script in the /etc/rc2.d directory. This is the line that specifies inetd -s.

Examine your /etc/inetd.conf file on a routine basis to verify that unnecessary services have been disabled.

START /etc/inetd.conf with logging enabled

Edit /etc/init.d/inetsvc
Add - t option to inetd command line
/usr/sbin/inetd -s -t

/etc/services

This file identifies which service is listening on which port. Comment out any ports for services that you want disabled. Place a '#' symbol at the start of each line, for the port you wish to disable. Save the /etc/services. No process restarting or rebooting is required. This file acts as a database which is referred to as network service requests are received.

Remote Commands ("R" services)

The "R" services were developed by Berkeley to provide seamless authentication between trusted hosts and users. The remote commands can allow users/attackers to circumvent password authentication via trusts. Authentication is based on IP address, TCP port, and client username. Remote shell (in.rshd) port 514 and remote login (in.rlogind) port 513 are the more dangerous remote services and should be turned off.

USE the Secure Shell (ssh) program to replace rlogin (remote login) and rsh (remote shell). Secure shell is a suite of utilities which utilize port 22 to set up public key encrypted remote access. It packages with a secure shell daemon (sshd) which handles all the connections, a secure shell program (ssh), a secure copy program (scp), a secure login program (slogin), and a secure ftp program (sftp).

rexec (remote execute)

rexec runs on port 512 and uses standard username and password authentication. Brute force attempts may go unnoticed as the rexecd performs poor logging. All communications are unencrypted. Remote Execute in the case of an invalid username responds with "login is incorrect" In the case of an invalid password it responds "password is incorrect".

- There is no access control built in to rexec.
- Disable rexec. If client applications rely upon it, figure out a migration path away and then disable it. If you can not disable rexec, consider using Secure Shell (ssh) to tunnel the program. SSH provides remote terminal access

File Transfer Protocol (ftp)

ftp has some security and management problems. It is a clear text protocol that allows users to connect to the system with a ftp provided shell. The shell normally provides access to 50 or more built in commands.

- ftp is often compromised in what is known as a bounce attack.
- While in active mode an attack can be made to look like it came from the ftp server instead of the attacker. It also allows for attacking network computers that are accessible to the ftp server but not to the attacker.
- ftp operates in two modes, active and passive

- ftp should be replaced with https, scp, sftp, ftp over a vpn, or commercial ftp versions which support encryption

/etc/hosts.equiv

The /etc/hosts.equiv file explicitly or implicitly trusts other hosts. Since trust is transitive, you may be trusting hosts not under your control. Be very careful when establishing trusts. Remote users can gain unauthorized root access to the system. Determine if the file /etc/hosts.equiv is required. If you are running certain "r" commands, such as rlogin, rsh, and rcp, this file allows other hosts to be trusted by your system. Programs such as rlogin can then be used to log on to the same account name on your machine from a trusted machine without supplying a password. Remember that trust is transitive.

IAW DA MSG DTG 050951ZMAR99, SUBJ: PROCEDURAL GUIDANCE FOR UNIX SYSTEMS, PARA 2-D: "Eliminate all rhost+, rhost++ or host.equiv to external hosts (external to the network)"

\$HOME/.rhosts

The .rhosts file is potentially a greater security risk than hosts.equiv, as one can be created by each user. A .rhosts file containing two plus signs "+ +", results in all hosts and all users being allowed to access the system without having to log in. The first plus sign indicates "all hosts" and the second one means "all users". Unlike /etc/hosts.equiv, .rhosts allow for root login. IAW AR 25-2, Section 3-3c(2)(j): "... users will not: Share personal accounts and passwords or permit the use of remote access capabilities..." Delete all .rhosts files.

Samba (SMB)

SMB is the Microsoft file and print sharing protocol. Samba makes a UNIX server look like an NT server by enabling it to run SMB. Vulnerabilities have been discovered in all versions of Samba, especially for Linux running on Intel platforms. Exploits may allow unauthorized remote users to obtain root access. NOTE: Samba loads NetBios on your UNIX computer.

Network File System (NFS)

NFS allows transparent access to files and directories of remote systems as if they were stored locally. NFS exhibits many vulnerabilities. In the worst case, intruders gain unauthorized root access from a remote host. Sharing system related file systems can occur when NFS is misconfigured. Weak authentication is used. Requests can be spoofed or sometimes proxied through the local portmapper.

1. DO NOT use NFS if you do not need to export file systems.
2. Disable NFS and related services, i.e. mountd, statd, and lockd.
3. Firewall protect your NFS server and block port 2049 on the firewall
4. Keep up on NFS security patches
5. Share information as "read only" whenever possible.
6. Use strong authentication in the /etc/nfssec.conf file

Be aware that you implicitly trust the security of the NFS server to maintain the integrity of the mounted files. A “web of trust” is created between hosts connected to each other via NFS. That is, you trust the security of any NFS server you use.

Domain Name Service (DNS)

DNS is one of the few services that is almost always required and running on an organization’s Internet perimeter network. BIND (Berkeley Internet Name Domain) is the name of the program which provides the relational database capabilities which make DNS possible. A flaw in BIND will almost surely result in a remote compromise. In a typical BIND attack, intruders erase the system logs, and install tools to gain administrative access.

1. SANS rates unpatched BIND servers as the #1 security problem on the Internet.
2. Disable/remove BIND from any system that isn’t used as a DNS server.
3. Ensure that the version of BIND you are using is current and patched for security-related flaws. BIND has had over a dozen security advisories in the last 4 years. Current versions of BIND do not have the identified vulnerabilities.

Log Files

Logs often harbor trace evidence of a crime. Logs are a good starting point to piece together what occurred on a computer system or network. Unfortunately, log files have an inherent vulnerability in that they may be forged and/or give misleading information. Securing the various log files on a UNIX system can mitigate this vulnerability.

Important log files:

lastlog	logs each user’s most recent successful login time, and possibly the last unsuccessful login
sudo	logs use of the su command
utmp	records each user currently logged in
utmpx	extended utmp
wtmp	provides a permanent record of each time a user logged in and logged out; also records system shutdowns and startups
wtmpx	extended wtmp
syslog	a host-configurable, uniform system logging facility
loginlog	records bad login attempts (after 5 tries)
vold.log	logs errors encountered with the use of external media
xferlog	logs ftp access
messages	records output to the system console and other messages generated from the syslog
acct or pacct	records commands run by every user (accounting)
.history	keeps a record of recent commands used by the user (not available in all shells)

All servers, as a minimum, will have auditing turned on and reviewed for the following activities:

1. all logins and attempts
2. all service connection requests
3. all ftp connections
4. all super user (root) connections, and requests

Consider installing a PC or other machine as a network log server.

Regularly monitor logs for successful and unsuccessful su attempts. Capture repeated login failures. Create file /var/adm/loginlog. This file does not exist by default. After 5 unsuccessful attempts on Solaris, an entry is made in the log. (This can be changed in the /etc/default/login file.)

```
#ls -l /var/adm/loginlog. If it does not exist, as root create it:
```

```
#touch /var/adm/loginlog
#chmod 600 /var/adm/loginlog
#chgrp sys /var/adm/loginlog
```

Ensure that permissions of all audit logs are set to 640 or more restrictive.

Syslog consists of 4 files:

1. syslog() - an application program interface (API)
2. logger - a UNIX command used to add single line entries to a system log
3. /etc/syslog.conf - configuration file used to control the logging and routing of system log events.
4. syslogd - system daemon used to receive and route system log events from the syslog() and logger program.

Syslog Priorities and Sources

The daily/weekly requirement to review log files is complicated by the sheer volume of information. Care should be taken to view log information that is relevant and important on a daily basis while lesser log entries can be viewed on a weekly basis.

Log information is affixed a priority:

- | | |
|------------|---|
| 1. Emerg | most immediate messages like system shutdown |
| 2. Alert | system conditions require immediate attention |
| 3. Crit | critical system conditions exist |
| 4. Err | other system errors |
| 5. Warning | warning messages |
| 6. Notice | notices requiring attention at a later time |
| 7. Info | informational messages |
| 8. Debug | messages for debugging purposes |

Log information has an originating facility:

- | | |
|---------|---|
| 1. user | (default) - generated by user processes |
| 2. kern | generated by kernel processes |
| 3. mail | generated by e-mail processes |

4. daemon generated by system daemons
5. auth generated by authorization programs
6. lpr generated by printing system
7. news generated by usenet news system
8. uucp generated by uucp system
9. cron generated by cron and at
10. local 0 thru 7 generated by up to eight locally defined categories
11. mark generated by syslog itself for time stamping logs

UNIX Tools

Pluggable Authentication Module (PAM)

PAM enables the authentication mechanism to be extended beyond UNIX passwords and to be both "stackable" and "tailorable" on a host- or application-basis using other authentication mechanisms like s/key, kerberos, and smart cards. Rules can be written such that a user must pass multiple authentication schemes to access high-security servers. In addition to authentication, PAM enables administrative customization of account management and session management. PAM allows you to change your authentication methods and requirements on the fly, and encapsulate all local authentication methods without recompiling any of your binaries.

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

1. Use a non-DES encryption for your passwords. (making them harder to brute-force decode)
2. Set resource limits on all your users so they can't perform denial of service attacks (number of processes, amount of memory, etc.)
3. Enable shadow passwords on the fly
4. Allow specific users to login only at specific times and places
5. Create a password history file
6. Write your own PAM to lockout accounts after 3 failed login attempts.
7. Disable .rhosts lookup
8. Port over Linux modules to Solaris and other UNIX platforms.
9. supports login, dtlogin, passwd, su, rlogind, telnetd, ftpd, ssh, & pop

Simple Watcher (Swatch)

Consider running an automatic log monitor such as Swatch. The Perl Compiler is required for installation. Swatch is a program that is designed to monitor system activity and filter log files. This package monitors and scans log files for pattern matches specified by the system administrator and takes action as specified by the system administrator. Swatch's configuration file identifies what the program looks for and what it does when it locates a pattern match. Swatch can react in three different ways to triggers: email; page; execute scripts. (The STIG refers to Swatch as "Simple Watchdog").

Tripwire

Tripwire is a utility that scans a set of designated files and directories, computes a digital signature, then compares the digital signature/fingerprint to a signature previously generated and stored in a database. Differences, including additions and deletions, are flagged and logged. When used regularly, it enables a system administrator to rapidly spot any changes to files and directories.

Tripwire has 9 levels of security descriptors for each file or directory it monitors. It can monitor files that can not be changed, binaries with the SUID/SGID bit set, read only binaries, configuration files, logs, directory permissions/ownership, members of the trusted computing base, kernel processes, and dynamic kernel processes. Tripwire watches file sizes and computes checksums of files to produce signatures that shouldn't change.

Tripwire is a good tool for keeping Trojan horses off of your system. It protects you from unauthorized persons toying with your critical data files. You establish a policy that consists of variable and rule definitions. The Tripwire policy tells tripwire what files to examine, what types of information to look for, and when to alert you to changes. Tripwire utilizes a site and local pass-phrase to encrypt tripwire policies, databases, and configuration files to keep them from being tampered with.

sudo

sudo is a program designed to allow a sysadmin to give limited root privileges to users and log root activity. The basic philosophy is to give as few privileges as possible but still allow people to get their work done. sudo is configured in the sudoer file. The commands or directories from which commands can be run are identified by the individual user allowed to run them. The sudolog records activity.

TCP_wrapper

TCP Wrapper is high speed, low drag protection that does not require additional communications between the client and server. It serves as a stateless firewall, applying rules prior to allowing connections. New releases of some OS's already come with TCP Wrappers incorporated into a xinetd.conf file for use by the xinetd program.

1. This software provides logging and access control for most network services.
2. TCP Wrappers provides additional logging; a banner; reverse DNS lookup; and access control.
3. Enable PARANOID mode. This is usually enabled by default.
4. Deny all hosts by putting "all:all" in the /etc/hosts.deny file and explicitly list trusted hosts who are allowed access to your machine in the /etc/hosts.allow file.
5. Wrap all TCP services that you have enabled in /etc/inet/inetd.conf or /etc/inetd.conf, or other appropriate files. For example, to "wrap" telnet:
 - a. Original entry:
telnet stream tcp6 nowait root /usr/sbin/in.telnetd in.telnetd
 - b. Modified entry:
telnet stream tcp6 nowait root /usr/sbin/in.tcpsd /usr/sbin/in.telnetd
6. Consider wrapping any udp services you have enabled. If you wrap them, then you will have to use the "nowait" option in the /etc/inet/inetd.conf file.

IPSec

IPSec is a security architecture for the IP protocol. It is a layer 3 (network layer) security solution that is an Internet standard and is not dependent on any particular encryption or authentication algorithm or operating system. IPSec is transparent to upper-level protocols and applications. IPSec provides authentication, encryption, integrity, and replay protection.

1. IPsec uses two protocols to provide security at the IP level: Authentication Header (AH) and Encapsulating Security Payload (ESP).
2. IPsec has two modes: transport and tunnel modes
3. IPsec may be used to configure a VPN (virtual Private Network)

npasswd

Use `npasswd` as a replacement for the default UNIX `passwd` command. `Npasswd` (new password) is a replacement for the system `passwd` command that incorporates a password checking system (word lists) that refuses poor password selections. It also incorporates the `crack` utility to eliminate easily guessed passwords from being used. This program reduces the chance of users choosing poor passwords.

Secure Shell

Secure Shell (`ssh`) is a program used to log into another computer over a network, to execute commands in the remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels. It is a suggested replacement for `telnet`, `rlogin`, `rsh`, `ftp`, and `rcp`.

Secure Shell can be used to tunnel remote X sessions, do secure network backups, and remote administration.

USE the Secure Shell (`ssh`) program to replace `telnet`, `rlogin` (remote login) and `rsh` (remote shell). Secure shell is a suite of utilities which utilize port 22 to set up public key encrypted remote access. It packages with a secure shell daemon (`sshd`) which handles all the connections, a secure shell program (`ssh`), a secure copy program (`scp`), a secure login program (`slogin`), and a secure ftp program (`sftp`).

Use `sftp` or `scp` to replace standard `ftp`. `ftp` is a clear text protocol which provides over 50 built-in commands to its user.

NOTE: IAW DA Msg 050951ZMar99, all unencrypted root account access must take place on the physical console. Secure Shell is an ACERT recommended commercial product and can be obtained at <http://www.ssh.org>.

There is also a free version of SSH available at <http://www.openssh.com>.

Penetration Testing

One of the best tests of your security is to break into your system. Hacking your own system can identify backdoors that may exist for the hacker. Start your penetration testing with the simplest methods first. Use common tools readily available on the Internet. Analyze the access control infrastructure for vulnerabilities and single points of failure. Any weaknesses identified by the penetration testing should be fixed immediately. NOTE: Make sure that the testing is authorized and conducted with management's knowledge. *Penetration testing is not authorized once the system has been connected to the network.*

Discovering a Break-In

Log Files

Examine log files for connections from unusual locations or other unusual activity. For example, look at your last log, process accounting, all logs created by syslog, and other security logs.

If your firewall or router writes logs to a different location than the compromised system, remember to check these logs also. Note that this is not foolproof unless you log to append-only media; many intruders edit log files in an attempt to hide their activity.

SetUID & SetGID Files

Look for setuid and setgid files (especially setuid root files) everywhere on your system. Intruders often leave setuid copies of /bin/sh or /bin/time around to allow them root access at a later time. The UNIX find(1) program can be used to hunt for setuid and/or setgid files. For example, you can use the following commands to find setuid root files and setgid kmem files on the entire file system:

```
#find / -user root -perm -4000 (-print as required)
#find / -group kmem -perm -2000 (-print as required)
```

Note that the above examples search the entire directory tree, including NFS/AFS mounted file systems. Some find commands support an "-xdev" option to avoid searching those hierarchies.

For example:

```
#find / -user root -perm -4000 -print -xdev
```

Another way to search for setuid files is to use the ncheck(8) command on each disk partition. For example, use the following command to search for setuid files and special devices on the disk partition /dev/rsd0g:

```
#ncheck -s /dev/rsd0g
```

System Binaries

Check your system binaries to make sure that they haven't been altered. We've seen intruders change programs on UNIX systems such as login, su, telnet, netstat, ifconfig, ls, find, du, df, libc, sync, binaries referenced in /etc/inetd.conf, and other critical network and system programs and shared object libraries. Compare the versions on your systems with known good copies, such as those from your initial installation media. Be careful of trusting backups; your backups could also contain Trojan horses.

Trojan horse programs may produce the same standard checksum and timestamp as the legitimate version. Because of this, the standard UNIX sum command and the timestamps associated with the programs are not sufficient to determine whether the programs have been replaced. The use of cpm, MD5, Tripwire, and other cryptographic checksum tools is sufficient to detect these Trojan horse programs,

provided the checksum tools themselves are kept secure and are not available for modification by the intruder. Additionally, you may want to consider using a tool (PGP, for example) to "sign" the output generated by MD5 or Tripwire, for future reference

Monitoring Programs

Check your systems for unauthorized use of a network monitoring program, commonly called a sniffer or packet sniffer. Intruders may use a sniffer to capture user account and password information.

cron & at Jobs

Examine all the files that are run by cron and at. We've seen intruders leave back doors in files run from cron or submitted to at. These techniques can let an intruder back on the system (even after you believe you had addressed the original compromise). Also, verify that all files/programs referenced (directly or indirectly) by the cron and at jobs, and that the job files are not world-writeable.

Unauthorized Services

Check for unauthorized services. Inspect `/etc/inetd.conf` for unauthorized additions or changes. In particular, search for entries that execute a shell program (for example, `/bin/sh` or `/bin/csh`) and check all programs that are specified in `/etc/inetd.conf` to verify that they are correct and haven't been replaced by Trojan horse programs.

Also check for legitimate services that you have commented out in your `/etc/inetd.conf`. Intruders may turn on a service that you previously thought you had turned off, or replace the `inetd` program with a Trojan horse program.

Password File

Examine the `/etc/passwd` file on the system and check for modifications. In particular, look for the unauthorized creation of new accounts, accounts with no passwords, or UID changes (especially UID 0) to existing accounts.

Configuration Files

Check your system and network configuration files for unauthorized entries. In particular, look for '+' (plus sign) entries and inappropriate non-local host names in `/etc/hosts.equiv`, `/etc/hosts.lpd`, and in all `.rhosts` files (especially `root`, `uucp`, `ftp`, and other system accounts) on the system. These files should not be world-writeable. Furthermore, confirm that these files existed prior to any intrusion and were not created by the intruder.

Hidden or Unusual Files

Look everywhere on the system for unusual or hidden files (files that start with a period and are normally not shown by `ls`), as these can be used to hide tools and information (password cracking programs, password files from other systems, etc.). A common technique on UNIX systems is to put a hidden directory in a user's account with an unusual name, something like `'...'` or `'.. '` (dot dot space) or `'..^G'`

(dot dot control-G). Again, the find program can be used to look for hidden files, for example:

```
#find / -name ". " -xdev (-print as required)
```

```
#find / -name ".*" -xdev | cat -v (-print as required)
```

Also, files with names such as '.xx' and '.mail' have been used (that is, files that might appear to be normal).

Local Machines

Examine all machines on the local network when searching for signs of intrusion. Most of the time, if one host has been compromised, others on the network have been, too. This is especially true for networks where NIS is running or where hosts trust each other through the use of .rhosts files and/or /etc/hosts.equiv files. Also, check hosts for which your users share .rhosts access.

Create a UNIX Response Toolkit

When investigating an incident, it is vital to have trusted commands on hand. Create a CD or floppy disk with the following tools:

```
ls, dd, des, file, pkginfo, find, icat, lsof, md5sum, netcat, netstat, pcat, perl, ps, strace, strings, truss, df, vi, cat, more, gzip, last, w, rm, script, hash, modinfo, lsmmod, ifconfig
```

Protect the Evidence

When an incident is suspected, preserve the evidence by making a duplicate of the evidence media. Perform the investigative steps on the restored image. Freezing the scene is key to preserving evidence. Evidence has differing levels of volatility. It can be in places such as the processor registers, kernel data structures in memory, swap space, network data structures and counters, user process memory and stacks, file system buffer cache, the file system itself, etc.

Conducting a UNIX Investigation

1. Review all pertinent logs
2. Perform key word searches
3. Review relevant files
4. Identify unauthorized user accounts or groups
5. Identify rogue processes
6. Check for unauthorized access points
7. Analyze trust relationships
8. Maintain a chain of custody on any evidence collected.