United States Army Signal Center and Fort Gordon

Fort Gordon, Georgia 30905-5144

School of Information Technology

Information Assurance Division

System Administrator/Network Manager Security Course

Week One

Student Handout

(Unix Security)

14 June 2005

Table of Contents

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 conceiveably 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 battlefront 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

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

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 csh man.

Access Control

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

Communications

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

Programming Tools

awk	pattern scanning and processing language
cc	C compiler (xlc on ADS)
crontab	maintain periodic tasks
csh	C shell command interpreter
kill	kill a process
make	manage multipart program projects
nice	run command at low priority (see "man nice" or "man csh")
nohup	run a command immune to hangups
sh	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	list previously issued commands (C shell, bash, and ksh only)
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 (I386)
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 (pg 13 & PE 8)
3. Secure the inetd (pg 20 & PE 13)
4. Secure the startup scripts (PE 1)
5. Enable logging (pg 23 & PE 17)
6. Protect against buffer overflows (pg 17), root network access (below), system account access (pg 32 & PE 13), sendmail (PE 8 , PE 9 & PE 14), motd misuse (pg 19), permissions (starting on pg 13), trusts (pg 22)
7. Install ssh, secure shell (pg 27 & PE 10)

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
-------------  perform logical AND, (1 AND 1 = 1, everything else = 0)
110 100 100    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 a 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.

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

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 and 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.

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 misleadning infomation. 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
sulog	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 shear 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. <u>Original entry:</u>
    ```
    telnet stream tcp6 nowait root /usr/sbin/in.telnetd in.telnetd
    ```

    b. <u>Modified entry:</u>
    ```
    telnet stream tcp6 nowait root /usr/sbin/in.tcpd /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, lsmod, 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.

NOTE: WHEN USING DTPAD, IF YOU ADD A LINE TO THE BOTTOM OF A FILE, MAKE SURE YOU PRESS THE ENTER KEY AT THE END OF THE LINE. SOLARIS PROCESSES TEXT FILES ONE-LINE-AT-A-TIME AND USES THE CARRIAGE RETURN AT THE END OF EACH LINE AS A DELIMITER. IF YOU DO NOT PRESS ENTER AT THE END OF THE LAST LINE, THE LINE MAY BE IGNORED.

PE 1 - Introduction

This Practical exercise is designed to familiarize you with UNIX, UNIX files and file structures, and the operating system in general. In the following steps, you will learn about, view and examine files for structure and key security entries. The commands you type will be printed in bold text. Do not type those that are bold and underlined. Type the commands as you see them.

1. It is time to login to your computer. You should see a box with a warning banner greeting you. Type your Log on ID. Your Log on ID is: **root**

2. After entering you Log on ID, you need to supply a password. Your password is **student**

3. The system should boot up in the Common Desktop Environment (CDE). It has a purple background and has the word "Solaris" tiled all over it. There may be windows open in the desktop area. Close them by right-clicking on the title bar and selecting "close".

4. After you have closed all the windows, right-click anywhere on the desktop and a popup menu will appear. Select "Tools". A new popup menu will appear: select "Terminal". This opens a terminal (shell) window on your desktop. This is where you will be doing the majority of your work. <u>*All PEs are designed to function correctly with the basic Bourne shell. Using any other shell for these PEs will not work.*</u>

5. You are required to shut down the system at the end of the day; and may have to do so for some other reason. To shut down, you must be logged in as root.

   –DO NOT Press the reset button
   –DO NOT Press the power button
   –DO NOT pull the power cord

   Improper shutdown will result in the loss of data or the requirement to run a file system check (fsck -y).

   shut down by entering the proper command at a shell prompt

   init 0, **shutdown**, (**sync**, **sync**, **halt**), **halt**

   reboot the system by entering the command at the shell prompt

   init 6(soft boot), **reboot**

6. What constitutes a UNIX system? This simple definition will suffice:

   A set of enabling technologies first developed at AT&T that have been incorporated into several legally distinct but closely related operating systems, each of which can be considered to be a UNIX system. If it looks like UNIX, operates like UNIX, runs common UNIX utilities and programs, and is developed with UNIX as a model, it is UNIX.

7. On any UNIX system, there are 3 types of users:

Root

- has complete control over the complete system
- can access all files
- the only account that can execute certain commands
- has the UID of 0 (zero)

Normal users

- any user that can login
- have a home directory in which they can create and manipulate files
- can not perform many system level functions
- usually a human being

System users (System accounts)

- system users don't log in
- accounts that are used for specific system purposes
- The nobody account is an example (nobody handles http requests)
- System accounts are created during install and without passwords.

8. UNIX has various run levels that you can choose from. Each run level (0 through 6 and S/s) is tied to one or more directories containing script files. Run levels 2 thru 5 may vary between UNIX and Linux distributions.

0	terminates the operating system; safe to power down
1	single user, admin/maintenance level.
2	multi-user, no NFS services
3	multi-user including NFS (default)
4	not used by most flavors
5	safe for automatic power down if supported
6	shuts the system down and reboots to default level (soft boot)
S or s	single user mode with all file systems mounted

• The command “who -r” will show you the run level
• To invoke a run level, type “init #” where “#” is the run level

9. Check your default run level. When you boot into a run level, the system executes the scripts for that run level.

   a. Run the "who -r" command to find your default run level. Type: **who -r**. Run level 3 actually executes the scripts in both the rc2.d and rc3.d directories. Notice in the run level list in step 8 that the only difference between run level 2 and 3 is that run level 3 starts up the NFS (Network File System). Let’s take a look at what one of these directories look like:

   b. Change to the directory for run level two. Type: **cd /etc/rc2.d**

   c. Run the "ls" command to see the directory contents. Type: **ls -al**

d. Notice the contents of the directory. Some scripts start with a capital "K" and others start with a capital "S". Those scripts starting with a "K" will kill processes. Those with an "S" start up a service or process. These are known as your "start up" scripts. To stop a script from running, rename it. Example: S72inetsvc could be renamed to _S72inetsvc. This would effectively turn off this script. Attackers, if they gain access to your system, will more than likely add scripts or modify existing scripts. Understanding what scripts should be in the /etc/rc* directories is crucial. All startup/shutdown scripts should be readable and executable only by root (UID = 0). Examine all "S" files in /etc/rc2.d and /etc/rc3.d (or similar directories). Any files that start unneeded services should be renamed (be sure the new names DO NOT start with a capital "S").

e. Run the "cd" command to go back to your home directory. Type: **cd**

10. Let's find out where our default run level is set up.

    a. View the /etc/inittab file. Type: **cat /etc/inittab**

    b. Find the line that reads "is:3:initdefault". The 3 refers to the default run level. It should match what you saw when you ran the "who -r" command. Changing this setting would change the default run level.

    c. The /etc/inittab:

        1) Defines which processes should be started up at boot time.

        2) Defines and controls the known run levels

        3) This is one of the system’s most important configuration files and should have permissions which will protect it: owned by root/sys with minimum permission levels of -rw-r--r-- (644).

        4) Composed of entries delimited by a new line:

<u>id</u>:<u>rstate</u>:<u>action</u>:<u>process</u>

id	one or two characters to uniquely identify an entry
rstate	define a run level (0-6)
action	key words on how to treat the process specified in process field (respawn, wait, once, boot, etc)
process	command to be executed (any legal shell text)

11. Let's look at the file that contains our user information.

    View the /etc/passwd file. Type: **cat /etc/passwd**

<u>login-ID</u>:<u>password</u>:<u>user-id#</u>:<u>group-id#</u>:<u>user-info</u>:<u>home-dir</u>:<u>shell</u>

login-id	2-->8 characters containing lower case alphabetic characters and numbers. (usermod -l)
password	13 character encrypted password (including 2-char salt); if empty,

	login does NOT prompt for a password; if 1-->12 characters, NO password will ever match; an "x" indicates that the password is stored in the shadow file.
user-id#	UID, numerical ID for the user, should be between 0 and 60000 (usermod -u); "0" user-id indicates the root user; any account with the user-id of zero will be treated as root; there should only be one root account
group-id#	GID, numerical ID for the group that the user belongs to, should be between 0 and 60000. (usermod -g)
user-info	User's real name, etc. *(GECOS) (usermod -c)
home-dir	Path to the directory the user is logged in to. Specifies the location of the user's (usermod -d)(passwd -h) home within the operating system. The user is placed here by the login program. For a normal user, this directory should be owned by the user.
shell	The user's initial shell program. If one is not listed, a default shell will be assigned. The default shell if this is empty is /bin/sh. (usermod -s) *Every account must have an authorized valid shell assigned to it in order to operate on the system.*

a. Guest accounts: Verify that NO guest accounts exist in either the /etc/passwd or /etc/shadow file. Note: Most systems come preconfigured with guest accounts. If you cannot remove the account, disable login to the account by assigning /bin/false or /dev/null as the shell.

b. Distributed Authentication (Naming Information System NIS/NIS+): Improper setup of NIS and/or NIS+ have been the cause of several exploits against UNIX systems. Users could have access to the /etc/passwd and the /etc/shadow files. If you do not need to have a distributed database of all objects in your network, then do not use NIS/NIS+.

   DO NOT run NIS or NIS+ if you don’t really need to.
   If NIS functionality is required, use NIS+ if possible.

c. Examples of some shells:

Valid Shells	File Name	Prompt
Bourne Shell (1)	/bin/sh	$
C Shell (2)	/bin/csh	%
Tenet C Shell (3)	/bin/tcsh	>
Korn Shell (4)	/bin/ksh	$
Bourne Again Shell (5)	/bin/bash	$
any shell		# (you are ROOT)

False Shells	File Name	Prompt
	/bin/false	n/a
	/dev/null	n/a

d. Restricted Shells. Assigning a restricted shell to a user account severly curtails the user’s ability to perform certain functions:

1) The user can't change the current directory.

2) The user can't change the value of the PATH environmental variable (even though they own their .profile).

3) The user can't use command names containing slashes (meaning they can't start commands at root).

4) The user can't redirect output with > or >>

5) If the user attempts to interrupt the restricted shell while it is processing the $HOME/.profile, the shell will immediately exit.

Restricted Shells	File Name	Prompt
Restricted Bourne Shell	/usr/bin/rsh	$
Restricted Korn Shell	/usr/lib/rksh	$

e. There are also shells designed for remote connections and secure connections. These types of shells are not assigned to a user but act as utilities to gain network access. Examples are remote shell or secure shell.

f. Shells can be valid and still not be authorized for login. Root can create a /etc/shells file and can list in it the shells authorized for login. If a /etc/shells file exists, all shells not listed in it will function similar to a false shell with regards to login. NOTE: /dev/null, /bin/false, or /bin/true can be used to disable user accounts, but WILL NOT be listed in the /etc/shells file. If you place them in the /etc/shells file, ftp access is enabled for the accounts with a false shell.

g. Make sure that users who should no longer have access are removed from all the systems to which they had access. Accounts inactive for 45 days should be disabled. (IAW AR 25-2 guidelines inactive accounts WILL be disabled after 45 days of inactivity.)

12. Most current versions of UNIX/Linux/BSD have implemented password shadowing. The permissions on this file are more restrictive than those on the /etc/passwd file. This prevents anyone other than root from reading the file or making a copy of the file and then attempting to crack user passwords. Let's look at the file that contains our password information.

View the /etc/shadow file. Type: **cat /etc/shadow**

<u>login-ID</u>:<u>password</u>:<u>lastchg</u>:<u>min</u>:<u>max</u>:<u>warn</u>:<u>inactive</u>:<u>expire</u>:<u>flag</u>

login-id	The login-id as it appears in the /etc/passwd file
password	13 character encrypted password (including 2-char salt); (passwd -l to lock); (passwd -d to delete); NP indicates no password is valid; *LK* indicates the account is locked until the superuser sets a password
lastchg	# of weeks from Jan 1, 1970 to the last password change
min	Minimum # of days required between password changes (passwd -n).
max	Maximum # of days the password is valid (passwd -x).

warn	# of days in advance to warn the user that the account password is going to expire (passwd -w).
inactive	# of days of inactivity allowed for the user (usermod -f)
expire	Absolute date after which the login may no longer be used (passwd -f) (usermod -e).
flag	Currently not used

a. The login-id's must match the /etc/passwd file entries. These two files contain a list of all your accounts by login-id.

b. System accounts have passwords of "NP", for no password; or, "*LK*" for locked. (In fact, any entry in this field other than a valid password hash will lock the account.)

c. Let's lock an account. Type: **passwd –l nobody**

d. View the /etc/shadow again. Type: **cat /etc/shadow**

e. Notice that the "nobody" account is now locked.

f. To unlock an account, run the command in the same format without the "-l" switch. Do not unlock the “nobody” account. When you unlock an account, you will have to provide a password for the account unlocked. (To unlock it without having to provide a password, you will have to edit the file with a text editor and manually delete the *LK* entry.)

13. Lets look at the file that lists groups and group memberships. Only members of multiple groups are listed in this file. If an account belongs to only one group, the /etc/passwd file will show that group.

    On most UNIX systems, the *wheel* group is used to indicate users who have access to the su command which invokes superuser privileges. Never add members to the wheel group natively (don't make it their primary group in /etc/passwd). Only add to the wheel group in the /etc/group file.

    a. View the /etc/group file. Type: **cat /etc/group**

    b. Format is: <u>groupname</u>:<u>password</u>:<u>group id</u>:<u>user list</u> separated by commas

14. Now let’s view the file that establishes some of the more important security settings -- /etc/default/login. Type: **cat /etc/default/login**

CONSOLE	control where root can login
PASSREQ	determine whether passwords are required
PATH	initial path after logging in
SLEEPTIME	time to sleep in between failed login attempts
SUPATH	initial path when su-ing to root. ENSURE that "." is NOT in root’s search path.
SYSLOG	if ‘yes’, all root logins are recorded by syslog

TIMEOUT	time in seconds before abandoning a login session
UMASK	default login umask
RETRIES	number of times before the login process exits
SYSLOG_FAILED_LOGINS	how many failed login attempts allowed prior to recording failed login messages, 0 means 'log all'

a. Find the line that reads CONSOLE=/dev/console. This entry controls root login. This setting will allow root login at the server but not over the network. It is called "securing the terminal". Setting CONSOLE=/dev/null will disable all root logins and ensure complete protection of the root account. SU access would be the only root access. Entries set to CONSOLE=   [empty] will also disable login. Always assume that the root password has been compromised. All unencrypted root account access must take place on the physical console. Programs such as secure shell (ssh) should be used for remote login, ensuring that passwords are not transmitted in the clear. That is why we restrict root access to local only. (In Linux, make sure all the ptys in the /etc/ttys file are set to unsecure. This stops the rlogin/telnet connections.)

b. Find the line that reads PASSREQ=YES. This entry ensures that your users will be authenticated via passwords.

c. Find the line that reads SYSLOG=YES. This entry ensures that root logins are recorded.

d. Find the entry #UMASK 022. This entry is one of the places that your system umask may be assigned. The pound sign, #, at the beginning of the line indicates that it is commented-out. The computer will ignore everything on this line. Normally this indicates that the system umask is set in the /etc/profile.

e. Find the entry that reads #RETRIES=5. This entry is commented out so that it does not get read by the system. Retries must be set to 3 IAW AR 25-2. It specifies that all systems will be configured to limit logon attempts to 3 tries by timing out or disabling them. Retries will reset the session but will not lock out the account. This entry needs to be changed to 3 and the comment (#) mark removed. If you are using LDAP, "3 tries and out", the native Solaris 8 client LDAP does not support account lockout. A 3rd party tool like PAM must be used to lock out accounts.

   LDAP = lightweight directory access protocol
   PAM = pluggable authentication module

f. Find the entry reading "SYSLOG_FAILED_LOGINS". This variable determines how many failed login attempts are detected before a failed login message is logged. The default is 5. A setting of "0" would log all failed attempts.

15. Type: **cat /etc/profile**. This file is used to set system-wide login parameters. All users, except those with a restricted shell, receive these values unless their account settings override it for their login session.

    a. Establish the umask setting

    b. Establish the path

c. Display MOTD or check for mail

d. Design the look of the prompt

16. Towards the bottom of the file there is an entry for umask. The default for Solaris 9 is 022. This value will be shared by all who log on the system unless they set their own unique umask in their ~/.profile, ~/.login, or ~/.cshrc file.

17. Let's look at the file that allows us to set up a password policy within UNIX. Type: **cat /etc/default/passwd**

    a. Notice that it only has three options, two of which are not even assigned. The PASSLENGTH refers to the character length of the password. The Army standard calls for 10. You would have to adjust this with a text editor. The MAXWEEKS refers how long the password is good before it expires; 150 days/21 weeks for systems. The MINWEEKS establishes the earliest time when you would be able to change your password, 90 days/13 weeks for systems.

    b. This file does not control password content nor address security measures found in Windows systems. Weak, easily guessed passwords allow access to accounts by unauthorized personnel. Passwords of insufficient length increase password cracking tool's efficiency. You must turn to 3rd party tools for those needs.

18. At the heart of the operating system lies the kernel. It is the compiled code that controls how the system manages system memory, the file system, disk operations, and processes. The kernel stays resident in memory at all times. Many of the things that the kernel controls, are directly targeted in denial of service attacks. As the kernel is compiled, changing how the kernel operates can only be accomplished by recompiling the kernel with new instructions or by placing entries in the /etc/system file. The /etc/system file is the one file that directly communicates to the kernel and allows you to alter settings that were precompiled. It operates similar to the Windows registry.

    a. Some of the more common uses for the/etc/system file are:

        1) Control the maximum number of user processes
        2) Set the maximum number of users
        3) Set NFS fileserver security
        4) Set the maximum number of groups per user
        5) Set stack sizes
        6) Set logging on for attempts to exploit via buffer overrun
        7) Set file descriptor limits (stndin, stndout, stnderr) (0<,1>,2>)
        8) Control use of chown by users
        9) Set maximum number of queued commands

    b. View the /etc/system file. Type: **cat /etc/system**

    c. A couple of example entries are:

        1) set nfssrv:nfs_portmon 1 (this allows nfs requests via ports 1 - 1024 ONLY)

        2) set rstchown=0 (this allows all users to change ownership on files)

19. Let's examine permissions and other key file information. The best way to do that is with the “ls” command and various flags. (-a) all files; (-l) long listing; (-b) nonprinting characters; (-R) recursive; (-u) access time.

    a. View the contents of the "/" directory. Type: **ls -alb**

    b. The format of the resulting file listing is:

       file type (-,d,c,b,l,s,p)
       owner's permission (r,w,x/s/S)
       group's permission (r,w,x/s/S/l)
       other's permission (r,w,x/t/T)
       access control list if any (+)
       links to the file
       owner
       group owner
       file size in bytes
       date saved or modified
       file name
       file linked to with file type "l"

    c. The file types are:

SYM	TYPE	OCTAL
-	normal file such as a text or database file	10
d	Directory	04
c	character device like a printer. Represent a specific kind of device that communicates with hardware, character by character	02
b	block device like a disk or CDROM. Represents a specific kind of device that communicates with hardware in units known as blocks. UNIX systems usually have peripheral devices attached to them. These devices may be involved with I/O terminals, printers, modems, disks, tapes, and may have specialized functions. The UNIX paradigm for devices is to treat each one as a file, some with special characteristics. UNIX devices are represented as inodes, identical to files. The inodes represent either a character or block device.	06
l	symbolic link. links to files in different filesystems. It allows the same file to have different names. Links are normally rwxrwxrwx but do not grant access unless the original file does. Anyone can link to anything.	12
s	socket. a general purpose interprocess communication mechanism. They allow processes that are not running at the same time or on the same machine, to exchange information. Pairs of sockets manage communications between different processes on your machine. You can read or write data to a socket just like to a file.	14
= or p	FIFO. A FIFO is a First In, First Out buffer which is a special kind of named pipe. It allows unrelated programs to exchange information	01

d. The permissions are:

r	read
w	write
x	execute
s/S	set UID/set GID
t/T	sticky bit

e. Read permission on a file (-r--r--r--) gives you permission to copy or view. Read permission on a directory (dr--r--r--) allows you to view directory contents. Read has a binary value of 4.

f. Write permission on a file (--w--w--w-) lets you edit the file contents and write permission on a directory (d-w--w--w-) allows you to delete files in the directory. *Write permission for the other's category is the most dangerous permission to allow.* Write permission for others is most often targeted by umask settings. Write has a binary value of 2.

g. Execute permission on a file (---x--x--x) allows you to execute the file and on a directory (d--x--x--x), to access files within the directory. Execute has a binary value of 1

h. Set UID (SUID) When you run a program that has the SUID set, you run the program as if you were the owner. It gives the program more privileges. An SUID program gives the person who runs it the Effective User ID (EUID) of the owner.

It lets underprivileged users function as a privileged user. By predesignating programs to run as root, a user could operate as a superuser. The SUID bit has a binary value of 4. Example:

- --s --- ---	(uppercase "S" in case of no underlying "x").
- r-s --x --x	has a value of 4511. Lower case ‘s’ as the owner still has execute permission.
- r-S --x --x	has a value of 4411. The S is uppercase as the owner does not have execute permission.

i. Set GID (SGID) permission allows the executer of a program to run it as if they belonged to the group that owns it by assigning them an effective GID of the owning group. -----s--- (uppercase "S" (or an "l") in case of no underlying "x"). If the system supports mandatory file locking, an l (lowercase L) may appear if the SGID is set and the group executable is removed. This will lock the file and prevent execution by both the group and others categories. The SGID bit has a binary value of 2. Example:

- r-x r-s r-x	has a value of 2555. The s is lowercase as the group has execute permission.
- r-x r-S r-x	has a value of 2545. The S is uppercase as the group has no execute permission.
- r-x r-l r-x	has a value of 2545. The l indicates that the system supports mandatory file locking and the group does not have execute permission.

j. Sticky bit permission, set on directories, allows deletion of files if your UID matches the file owner UID, directory owner UID, or root UID. --------t (uppercase "T" in case of no underlying "x"). The sticky bit has a binary value of 1. Example:

d rwx rwx rwt	has a value of 1777. The t is lowercase as the others category has execute permission.
d rwx rwx rwT	has a value of 1776. The T is uppercase as the others category has no execute permission.

20. Permissions are changed via the command “chmod”. Permissions are known as the file's mode and chmod stands for “change mode”. Only a file's owner or root may change file permissions. Either numbers or alpha/character entries are acceptable when assigning permissions.

Alpha/Character syntax:

```
Syntax: chmod [-Rfh] [aguo] [+-=] [rwxstl] filelist

Separate multiple permission settings by a comma:

Example: chmod u+rw,go+r "file"
```

a	All	r	Read
g	Group	w	Write
u	Owner [User]	x	eXecute
o	Other		
-	Remove	s	SUID/SGID
+	Add	t	Sticky Bit
=	Replace	l	Mandatory file locking

Numeric syntax:

Example: chmod 0644 or chmod 644

1000	Sticky Bit	0777	rwxrwxrwx
2000	SGID	0666	rw-rw-rw-
3000	SGID/Sticky Bit	4555	r-sr-xr-x
4000	SUID	1777	rwxrwxrwt
5000	SUID/Sticky Bit	6745	rwsr-Sr-x or rwsr-lr-x
6000	SUID/SGID		
7000	SUID/SGID/Sticky Bit		

Positional Values:

Octal		$0 \rightarrow 7$			$0 \rightarrow 7$			$0 \rightarrow 7$			$0 \rightarrow 7$	
Binary	4	2	1	4	2	1	4	2	1	4	2	1
Symbol	s	s	t	r	w	x	r	w	x	r	w	x

Special bits share the *execute* position in the appropriate area:

	SUID			SGID			Stick Bit		
Special Bits →			s			s			t
Permission Bits →	r	w	x	r	w	x	r	w	x
If all values are set →	r	w	s	r	w	s	r	w	t

21. Lets examine a file that is both a Set UID and a Set GID file.

    a. View permissions for the /etc/passwd **ls –l /etc/passwd**.

    b. Notice that the owner of the file is "root" and the group is "sys". Neither owner, group, or other has anything but read authority. This means that only root may edit this file as root has explicit write capability on all files. This also means that in order to change your password, you must involve root.

    c. View the permissions of the program used to change passwords **ls –l /bin/passwd**.

    d. Notice that the permissions are -r-sr-sr-x. Both the Set UID and Set GID are turned on. It is owned by "root" and "sys". This means everyone can run this program ("other" executable) and they will be assigned the *effective GID of sys* and the *effective UID of root*. While the /bin/passwd program is running, the system will see the person that runs it has having both root and sys authority. This is what allows you to change your password even though you lack the access.

22. Umask stands for "user file-creation mode mask". More simply put, when you create a file (or directory) it receives default permissions that can be masked out. The default permission for a file is 666 (rw-rw-rw-) and for a directory is 777 (rwxrwxrwx). These permissions do not meet our "least access" requirements. The "others" and "group" categories may be receiving permissions that are above normal requirements. A umask setting addresses this problem. The purpose is to assign an appropriate umask value that will mask out the permissions that could potentially create problems. Normally a system umask setting of 037 or 077 will serve this purpose. Very seldom does a group member or member of the "other" community need permission to read, write, or execute a program by default. The root administrator should have a umask setting of 077 to protect any files that root may create.

    How umask works:

Permission	Mode Setting	UMASK Setting
No Access	0	7
Execute	1	6
Write	2	5
Read	4	3
Full Control	7	0

Solaris text file default			
666	r w -	r w -	r w -
Umask			
022	- - -	- w -	- w -
Effective			
644	r w -	r - -	r - -

Directory:			
Permissions	7	7	7
Umask	0	3	3
Effective	7	4	4
File:			
Permissions	6	6	6
Umask	0	3	3
Effective	6	4	4

Permissions	7	7	7
Umask	7	5	5
Effective	0	2	2
Permissions	6	6	6
Umask	7	5	5
Effective	0	4	4

23. Below are some file permissions. Remember that the first character from the left identifies the file type; the next three, the owner's privileges; the next three, the group's privileges; and the last three, the other's privileges.

(r = 4, w = 2, x =1)

(--s------ = 4***) (-----s--- = 2***) (--------t = 1***)

Place the correct octal number next to each permission listed. Output can be 3 or 4 digits. Ignore the file type. (Example: -rw-rw-rw- = 666)

a.	dr--r--r--	444	b.	br--r--r--	____________
c.	-rwxr-xr-x	____________	d.	lrwxrwxrwx	____________
e.	-r-xr-xr-x	____________	f.	cr--------	____________
g.	-r-sr-x--x	4551	h.	drwx---r-x	____________
i.	brwxrwxrwx	____________	j.	srwxrwxrwx	____________
k.	---S--l--x	____________	l.	dr----S--T	____________
m.	crw-rw-rw-	____________	n.	-r-sr-sr-x	____________

24. Solve for the umask results. Example: 666 (text file) 027 (umask) = 640

	Default	umask	results (octal)
a.	666	002	664
b.	666	046	___________
c.	777	046	___________
d.	777	011	___________
e.	666	011	___________
f.	666	033	___________
g.	777	077	___________
h.	666	055	___________
i.	777	055	___________
j.	666	013	___________

25. Write out the following permission values. Example: chmod 555 = r-xr-xr-x

a. chmod 1555 =	_r-xr-xr-t_______	b. chmod 6775 =	_________________
c. chmod 2744 =	_________________	d. chmod 4501 =	_________________
e. chmod 0444 =	_________________	f. chmod 611 =	_________________
g. chmod 4040 =	_________________	h. chmod 1111 =	_________________
i. chmod 7000 =	_________________	j. chmod 3111 =	_________________

26. If you finish early, spend some time getting acquainted with the UNIX system. Examine the directory structure or desktop until everyone is ready to discuss the PE.

FROM THIS POINT FORWARD, IT IS THE STUDENT'S RESPONSIBILITY TO OPEN A TERMINAL WINDOW WHEN APPROPRIATE TO FACILITATE ENTERING COMMANDS AT THE COMMAND LINE PROMPT. STUDENTS WILL NOT BE TOLD TO OPEN A TERMINAL WINDOW BEFORE BEING TOLD TO TYPE A COMMAND.

End of PE

PE 2 - Hidden Files/Directories

The purpose of this PE is to familiarize you with hidden file names and how to detect them. UNIX doesn't typically create files with non-printing characters unless there happens to be a catastrophic system crash. If you discover hidden files, you can bet that someone is attempting to hide files on your system. The operative question would be "Why?"

Commands to type are written in bold. Only type what is in bold. To type ^E, hold down the control key and press the letter e at the same time.

1. Check your directory and make sure you are in the root directory (/). Type: **pwd.** If you are in a directory other than "/", type: **cd.**
2. Make a directory with a non-printing character, control-E. Type: **mkdir ^E**
3. Change to that directory. Type: **cd ^E**
4. Make a new directory. Type: **mkdir ^E**
5. Change to that directory. Type: **cd ^E**
6. Make a new directory. Type: **mkdir ^E**
7. Change to that directory. Type: **cd ^E**
8. Create a file. Type: **touch hacktool**
9. Create another file. Type: **touch ^B**
10. Go back to the root directory. Type: **cd**
11. Use the **ls** command to see your ^E directory. Can you see it? _______
12. Use the **ls-a** command to see your ^E directory. Do you see it? _______
13. Use the **ls -al** command to see your ^E directory. Do you see it? _______
14. Does the **ls -alb** command show you the ^E directory? _______
15. What does the name of the ^E directory print out as? __________________
16. Try and find the file you created. Type: **find / -name hacktool**
17. What is the location for the hacktool file? ___________________________
18. Write the proper syntax to delete the file using the rm command (rm /directory/filename).

    ____

19. Now try and find the ^B file using syntax similar to step 16. What is its location?

____

20. Using your answer in step 19, write the proper syntax that will delete (rm) the ^B file. What is the syntax?

____

21. Lets check the computer for any and all hidden nonprinting characters that have been used in the creation of file or directory names.

    Type: **ls -laR | grep '\\'** (see page 5 for an explanation of the | char)

    Note: The first non-character symbol in the command is the vertical bar (pipe). The symbols in ' ' are back slashes.

    What files or directories were found? "None" is an acceptable answer.

____

____

22. Type: **ls -labR | grep '\\'**

    What files or directories were found? What is strange about these files? Why were the files not displayed in the previous command? Notice that only the -b flag for ls will display nonprinting characters. Grep couldn't find the "\" until –b forced it to print.

____

____

23. Using the table below, identify the character that is defined by the octal characters in the file name. Omit any that you created earlier.

Decimal	Octal	Hex	Character	Remark
0	000	00	CTRL-@	NUL (Null prompt)
1	001	01	CTRL-A	SOH (Start of heading)
2	002	02	CTRL-B	STX (Start of text)
3	003	03	CTRL-C	ETX (End of text)
4	004	04	CTRL-D	EOT (End of transmission)
5	005	05	CTRL-E	ENQ (Enquiry)
6	006	06	CTRL-F	ACK (Acknowledge)
7	007	07	CTRL-G	BEL (Bell)
8	010	08	CTRL-H	BS (Backspace)
9	011	09	CTRL-1	HT (Horizontal tab)
10	012	0A	CTRL-J	LF (Linefeed)

24. Now lets find the absolute pathname to the file. Lets search for .\007.

    Type: **find / -name .^G**

    Note: To create the ^G, press the CTRL key and the G key simultaneously.

    Where is the file?

    ____

25. Now let's delete the file so that this anomaly is no longer present on your system.

    Type: **rm /var/preserve/.^G**

26. Is the use of hidden non-printing characters an effective way of hiding files and directories from most administrators?

    ____

End of PE

PE 3 - umask & Permissions

The purpose of this PE is to familiarize you with the UNIX file & directory structure, and to reinforce good security practices.

1. Type: **cat /etc/default/login**

   Do you see the entry “CONSOLE=/dev/console”?
   What restriction does this put on the root account?

   ____

   ____

2. Let’s verify the system is only using authorized shells. If not, modify the system to do so. First, determine if an /etc/shells file exists. Type: **ls –l /etc/shells.** Is the system requiring the use of authorized shells (does the /etc/shells file exist)?

   ____

   ____

3. Find the valid shells. Type: **ls –l /bin/*sh** (or ls –l /usr/bin/*sh). The UNIX shells you are looking for are <u>sh</u>, <u>csh</u>, <u>ksh</u>, and <u>rksh</u> (possibly remsh, bash, jsh, depending on vendor).

   How many shells did you find? ___________

   Do you see the shells listed in the question? _________

4. Create the /etc/shells file. We’ll use dtpad -- it’s a text editor. If you want, you can use the vi editor. Type: **dtpad /etc/shells** (you may have to type the absolute path to get the dtpad command to work. If so, type: /usr/bin/dtpad /etc/shells.)

   Confirm the creation of the new file if prompted.

   Add the following entries on separate lines: **/bin/csh**, **/bin/ksh**, **/bin/rksh**, **/sbin/sh.**

   (example)
   /bin/csh
   /bin/ksh
   etc.

   Save the file and close the editor.

5. Should you add the false shells (/bin/false and/or /dev/null) to the /etc/shells file? Why or why not?

   ____

____

6. Lets look at the account names and the shells that are assigned to them. Type: **cut -d: -f1,7 /etc/passwd** (this shows the first and seventh column of the /etc/passwd file).

   Which accounts have an authorized shell now that you have created an /etc/shells file? What should you do to the accounts which do not have an authorized valid shell identified?

   ____

   ____

7. Let's test to see if the /etc/shells file works. Log out of CDE. CDE is the graphical desktop program we are using. (select the Exit button in the Frontpanel toolbar).

8. Log in as labuser1 (password is **student1**). Verify that CDE is the selected Desktop. Solaris has two. CDE is the one we've been using. Open Windows is the other.

   Were you able to login as labuser1? Why?

   ____

9. Log in as root. Add the Bourne (/bin/sh) shell to /etc/shells file so that the labuser1 account can be utilized later.

   Type: **dtpad /etc/shells**

   Add the following entry on a separate line: **/bin/sh**

   Save the file and close the text editor.

10. UNIX doesn't normally create files without an identifiable owner or group. The discovery of such files either indicates that there is a major problem with your operating system, administrators are not removing user files when they delete user accounts, or someone is attempting to hide files.

    Let's determine if the system has any files that may imply the system has been compromised by finding all files and directories that do not have an identifiable owner or group (orphaned files).

    Type: **find / -nouser –o –nogroup**

    (the script matches all file UID and GIDs to existing accounts).

11. What files or directories did you find that do not have an identifiable owner or group ? If none were found, state so.

    ____

    ____

12. World-writeable files and directories are normally created as a matter of convenience; however, attackers will not overlook the obvious. Reconfiguration of system files such as those located in /etc/rc*.d could give attackers root access. Especially important are system initialization files, system configuration files, and user startup files. As an administrator, you should be concerned about the unrestrictive permissions that are set by default in a typical UNIX system.

    The permission that will cause the greatest security problem is the Write permission. You need to know which files and directories have the Write permission set for Group, but in particular for Other.

    We'll search all the type -, b, c, p files for other's write permission and then stick the results into a file called "wfile".

    Type:

    find / \(-type f –o –type b -o –type c -o -type p \) –perm –o+w > wfile

    Note: When you redirect output to a file, it does not display on the screen.

13. This script searches the same file list for group write permission and appends to the file.

    Type:

    find / \(-type f –o –type b -o –type c -o -type p \) –perm –g+w >> wfile

14. This script searches all directories for group write permission and sends the results to a file called "wdir".

    Type: **find / -type d –perm –g+w > wdir**

15. This script appends to list all the directory names that have others write permission.

    Type: **find / -type d –perm –o+w >> wdir**

16. Now let's look and see what our command did for us. The first two commands created a file (wfile) that contains a list of all the files that have either Write permission for group or Write permission for other. The last two commands created a file (wdir) that contains a list of all the directories that had either Write permission for group or Write permission for other.

    Type: **cat wfile**

    What is the content of the "wfile" file (describe)? Are the file names listed one per line?

    ____

    ____

17. Let's count the number of lines in the "wfile" file. Type: **wc -l wfile** (the result of this command provides the number of lines in the file).

    How many files have the 'Write' permission set for Group or Other ? Do the default file permissions leave potential holes in the system?

    ____

    ____

18. Type: **cat wdir**

    What is the content of the "wdir" file?

    ____

    ____

19. Let's count the number of lines in the "wdir" file. Type: **wc -l wdir**

    How many directories have the Write permission set for Group or Other? Do the default directory permissions leave potential holes in the system?

    ____

    ____

20. You've learned about the "Set User Id – SUID" and "Set Group Id – SGID". Let's see if your system has either of these types of files. We'll search the computer by searching for the permission octal values.

    a. Type: **find / -type f -perm -004000 > suidfiles**

    b. Type: **find / -type f –perm -002000 > sgidfiles**

    c.Write the proper syntax that will find all of the directories with the "sticky bit" set.

    ____

    d. Write the proper syntax that will find all of the files that have both the SUID and SGID bits set.

    ____

21. Attackers will take advantage of a program or process while it is performing a privileged operation. This usually involves timing the attack to abuse the program after it enters a privileged mode but before it gives up its privileges. A vulnerability that allows attackers to abuse this window of opportunity is called a "race condition." A good practice for reducing this vulnerability is to reduce the number of SUID/SGID binaries on the system. Let's count the number of lines (and, therefore, the number of suid files on the system) in the suidfiles file. Type: **wc -l suidfiles**

How many files have the SUID? ____

What are the security implications of having the SUID bit set indicate?

____

____

22. Remove the set-user-id (SUID) or set-group-id (SGID) bit from processes that do not need it. Most systems ship with more preset suid files than are required. Many of these programs are only accessed by root. This is a point of failure on all standard UNIX machines. Virtually every attack is based around the concept of gaining root access on a machine. Once this is achieved an intruder can literally do anything with your system. Check the output of step 20a. (**cat suidfiles**). Does the file "/usr/bin/admintool" exist on the list? Write the proper syntax using the "chmod" command that will turn off the SUID bit in /usr/bin/admintool.

    ____

23. Let's count the number of lines in the "sgidfiles" file.

    Type: **wc -l sgidfiles**

    How many files have the SGID? ____

    Is this a concern? ____

    Which should cause you the most concern, SUID or SGID? ______________

24. Let's investigate another vulnerability associated with SUID.

    a. Type: **cp /usr/bin/ksh /export/home/labuser1**
       This makes a copy of the korn shell and puts it into the labuser1's home directory.

    b. Type: **ls –l /export/home/labuser1/ksh** and make note of the current

       permission setting:______________________.

    c. Type: **chmod 4555 /export/home/labuser1/ksh**

    d. Type: **ls –l /export/home/labuser1/ksh** and make note of the new

       permissions:________________________

    e. What changed on the permissions? ______________________________________

       ____

    f. Type: **telnet localhost**

    g. Log in as labuser1 (password student1).

h. Type: **/export/home/labuser1/ksh**. What happened? What does this mean?

____

____

i. Processes that have a SUID or SGID will execute as either the owner of the file or the group that is assigned to the file. Files that are executed as root are particularly dangerous. Type: **id**. What is your user id? What is your effective user id? Could this be a security issue?

____

____

25. Type: **exit** to quit the korn shell session.

26. Type: **exit** again to quit the telnet session.

27. As a System Administrator, you are concerned about security and want to make permissions more restrictive. UNIX uses a umask setting to determine file permissions for newly created files and directories. Umask identifies the permissions that you do not want assigned to a file or directory. Let's see how the umask affects the creation of files and directories. Type: **umask**.

    What is your current umask setting? How does this setting affect the permissions of newly created files?

    ____

    ____

28. Let's set a system default umask. Normally we would want to ensure that the umask value for each user is set to 037 or a more restrictive value, such as 077. On an operational system, the umask value for root should be set to 077. Check .profile (for sh, ksh) or .login (for csh) for setting the umask value. Don't universally grant "other" permissions. The first way to set the system umask is to edit the file /etc/default/login and modifying the umask line to read "umask 037" or "umask 077". The second and more popular option is to edit the file /etc/profile by modifying the umask line to read umask "umask 037" or "umask 077". The /etc/profile umask value will override the /etc/default/login setting.

    Type: **dtpad /etc/profile**

    Locate the umask setting and change it to **umask 077**.

    Save the file and exit the editor.

29. Type: **touch foobar**

30. Type: **ls –l foobar**

What are foobar's permissions? Were they effected by the umask setting in the /etc/profile? Why?

____

____

31. Log out of CDE and log back in as root.

32. Type: **touch foobar2**

33. Type: **ls –l foobar2**

    What are foobar2's permissions? Was the umask setting in the /etc/profile used to create these permissions?

    ____

34. Log out of CDE, then log back in as labuser1. NOTE: Be careful to verify CDE is the selected Desktop.

35. Type: **umask**

    What is labuser1's current umask setting? What does this indicate about the umask setting in the /etc/profile file?

    ____

    ____

36. Type: **pwd**

    Are you in labuser1's home directory? ____________________________________

37. Type: **touch file1**

38. Type: **ls –l file1**

    What permissions does file1 have? Was the umask setting in the /etc/profile file utilized to set the file permissions?

    ____

39. Change labuser1's umask so that all files you create have the following permissions:

    owner = read, write
    group = read, write
    other = read

    Type: **umask 002**

40. Type: **touch file2**

41. Type: **ls –l file2**

    What are the permissions? Are they different from file1?

    ____

42. Users can set their own umask. This will override system defaults for the user. Hackers can take advantage of files that have weak permission settings. Let's set the labuser1 umask so that labuser1 always get the permissions labuser1 wants on login vice the system defaults (e.g. labuser1 won't have to type the umask command at every login). labuser1 wants everyone to be able to read and write the files labuser1 creates.

    Type: **dtpad .profile**

    Editing .profile allows you to change your operating environment. Create a new line and add: **umask 000**. Save the file and exit the editor.

43. Log out of CDE and log back in as labuser1.

44. Type: **touch file3**

    Type: **ls –l file3**

    What are the permissions? Are they different from file1 and file2? Which umask (/etc/profile, .profile) was used to create file3? Is this good security?

    ____

45. What could you do to stop a user from creating a .profile umask setting?

    ____

46. What would the umask setting in the ~/.profile need to be to create a resulting directory permission of:

    owner = read, write, execute
    group = read
    other = read

    ____

NOTE: When changing the default umask for root, be mindful of the final result.

End of PE

PE 4 - Access Control Lists

The purpose of this PE is to acquaint you with file access control lists (FACL). FACLs allow the administrator to set finer controls than are allowed by the default UNIX permissions. Failure to set FACLs on critical files and directories may leave those files and directories exploitable by malicious attackers. If a file or directory has an FACL set, a '+' will appear as the last character (far right) following the permissions. FACL are only supported by the UFS and are best used to control access to binaries which have the suid or sgid bit set.

1. Make sure you're in your home directory (/). If you're not, use the cd command to move there.

2. Create a file to assign an access control list to. Type: **dtpad myfile**

   Enter some text, save the file, and close the editor.

3. Check the file for assigned permissions. Type: **ls –l myfile**

   What are the permissions? ____

4. Gove everyone permission to the file. Type: **chmod 777 myfile**

5. Verify the new permissions. Type: **ls –l myfile**

   Are the permissions "full access" for owner, group, and other? ________________

6. Set up your access control list so that labuser2 has no privilege on the file.

   Type: **setfacl –m user:labuser2:--- myfile**

7. Verify the new access control list. Type: **getfacl myfile**

   What is labuser2's effective permission? ___________________________________

   Who has access to this file? ____

   If there, what are the group permissions? __________________________________

8. Type: **su labuser2**

   Type: **cat myfile**

   Could labuser2 read the file? _______

   Type: **exit**

9. Type: **cat /etc/passwd**

   Is labuser2 a member of the same group as the owner of the file (root)? _______

   If so, labuser2 can still access the file because of group membership.

10. Type: **usermod -g staff labuser2** to change labuser2's group to staff.

    Type: **cat /etc/passwd**

    You should see that labuser2 is now in a different group.

11. Type: **getfacl myfile**

    The group still has rwx permissions, but labuser2 is not in that group.

12. Verify that the file is reporting the access control list by checking permissions.

    a. Type: **ls –l myfile**

    b. Have the permissions changed from step 3? ______________________________

    c. Has a plus sign appeared at the end of the permissions to indicate an access control list? ____

13. Type: **su labuser2**

    Type: **cat myfile**

    Can labuser2 access the file now? ________

    Remember, group permissions can circumvent your efforts to block user access.

End of PE

PE 5 - Banners

The purpose of this PE is illustrate for the student how to establish banners. Login banners are a requirement for DoD systems. A good banner will convey the identity of the computer system, the fact that monitoring is taking place, use of the system is permission to be monitored, and any regulation governing the system use or the security of the system. We are going to create a couple of banners that a user will see when connecting to our system via telnet. These are sometimes referred to as "command line" banners.

(Refer to page 18 for a discussion of banners.)

1. Create a login banner for command line logins.

   Type: **dtpad /etc/issue**

2. Enter the following text: **is this my banner?** and press **ENTER**.

   Save and close the editor

3. Type: **telnet localhost**

4. Do you see your banner? __________

   Exit telnet by typing: **^D**

5. You will now create a specialized banner for telnet sessions only.

   Type: **dtpad /etc/default/telnetd**

6. Enter the following text: **BANNER="Hey Einstein, "** and press **ENTER**.

7. Save and close the editor.

8. Type: **telnet localhost**

9. Do you see both banners? _________

   Exit telnet by typing: **^D**

End of PE

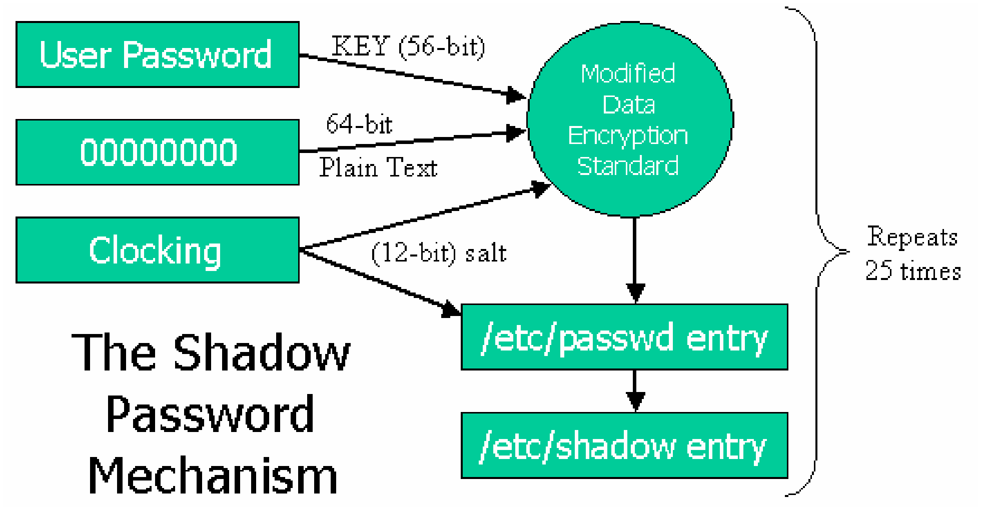
PE 6 - Passwords

You sit down at your computer. You enter in your user name, and then you enter your password. Magically, you are taken into your account. What is actually happening behind the scenes? How does your computer store passwords, and how does it keep them secure? This exercise will show you how to adjust for the Army's more stringent password encryption requirements. First, you will examine the default configuration utilizing a 6 character password using the default DES (56 bit) encryption algorithm. Then, you will upgrade the system to a 128 bit encryption method employing a 10 character password.

To begin with, a basic understanding of cryptography is necessary. There are three fundamental techniques of encryption: symmetric key-based algorithms, such as block ciphers and stream ciphers; asymmetric key-based algorithms, such as public key encryption; and hash ciphers, such as SHA and MD5. Most encryption schemes are based around one or more of these.

UNIX passwords are stored using one-way hash ciphers. The encryption is fairly simple, a fact that has its merits and its disadvantages. On the one hand, the relative simplicity means the encryption is performed quickly, so logging-in happens immediately. On the other, the simplicity means an encrypted password is easy to crack. An advantage of a one-way hash is that your password is not stored anywhere on your computer. Rather, a hash, (an encrypted string of characters) is stored in the password file (/etc/passwd, by default). The hash is formed when you first set the password. The system encrypted it using a set mathematical formula known as the hash function. The system knows how it hashed the sequence of characters that is your password, so every time you log on, the system encrypts what you have just typed using the same hash function, and compares the encrypted results to the encrypted password.

The password hash is easy to crack...that's a pretty big downside. In the classic configuration of UNIX systems, there isn't any way to protect the integrity of our passwords. The password file, /etc/passwd, is world readable. It has to be, because of other crucial information that's stored in it. Since it's world readable, anyone who's got read access to your computer can view your password file and run cracking programs against it.



To enhance security, the shadow password system was developed. When using shadow passwords, the computer stores non-sensitive information in the world-readable

/etc/passwd file, and stores the hashed passwords in the /etc/shadow file, which is [or should be] readable only by root. This way, not just anyone can read the file. Only those with root access can read the shadow file: if a hacker already has root access, there's no need to worry about the password files.

1. Make sure you are logged in as root. Type: **cat /etc/shadow**

2. You should see the user list starting with root and ending with labuser2. The second column (each column is separated by a ":") contains the encrypted password hash value. The first two characters are the salt value and the last 11 are the password hash. These password hashes were created using the Data Encryption Standard. This is the default and is only 56 bit encryption. 128 bit encrypted password hashes will be longer than 13 characters. Solaris comes with two other methods of encrypting passwords: *md5* and *blowfish*. md5 passwords start with $1$. DES is encrypted in a 64 character alphabet which does not include the $ character. DES uses a 2 character, 12-bit salt value which is the first two characters in each hash. MD5 uses a 12 character, 64-bit value. MD5 starts the salt with $1$ and ends it with $. Sun's version starts off with $md5$. The $ symbols are ignored by the algorithm. Its salt looks like $md5$blahblah$. Blowfish uses a 16 character, 128-bit salt and looks like this: $2a$blahblahblah$. Both MD5 and blowfish provide a baseline of 128 bit encryption.

3. You will now create a new user account and go through a series of password changes. Type: **useradd labuser3**

4. Verify that labuser3 has been created. Type: **cat /etc/passwd**

5. Check the status of the labuser3 password. Type: **cat /etc/shadow**

6. Do you see that the labuser3 account is locked? __________________________

7. Unlock the user account and give it a password. Type: **passwd labuser3**

8. When prompted for the password, type: **student3**

9. Repeat **student3** a second time to set the password. The computer should indicate that the password is now set for labuser3.

10. Now login as labuser3. Type: **exec login**

11. When prompted for a username, type: **labuser3**

12. Provide the password of: **student3**

13. You are now logged in as labuser3 (note the '$' prompt). Change the password for labuser3. Type: **passwd**

14. Enter the existing password: **student3**

15. When prompted for a new password use a short password. Type: **stuff**

16. Why didn't "stuff" work? ____

17. What is the current length requirement? ________

18. What should the requirement be? ________

19. Press **ENTER** until you are returned to the $ prompt.

20. Now set the password to 10 character length. Open a new terminal window and notice that it is running as root (note the '#' prompt). Type: **dtpad /etc/default/passwd** in this new terminal window.

21. Highlight the number 6 and replace it with 10. Make sure the phrase looks like **PASSLENGTH=10**. Save your changes and close the editor.

22. Click back inside the border of the terminal window running as labuser3 (the one with the '$' prompt). Type: **passwd**

23. Enter the existing password: **student3**

24. When prompted for a new password use a short password again. Type: **again**

25. Why didn't "again" work? ____

26. What is the current length requirement? __________________________________

27. Try a longer password. Type: **weloveUNIX**

28. What was wrong with this? ____

29. Lets try again. Type: **i82loveUNIX**

30. Did it work? ____

31. You have the length of the password fixed, now you'll increase the encryption strength. Click within the boundaries of the terminal window running as root (with '#' prompt). Type: **cat /etc/security/crypt.conf**

32. Standard DES encryption is built in. Look at the additional supported versions of encryption. What are the supported versions of encryption (last 3 lines of the file)?

    ____

33. Now type: **dtpad /etc/security/policy.conf**

34. You should see that at least 4 to 5 lines in the file do not start with a # symbol. Find the line that reads: CRYPT_ALGORITHMS_ALLOW=1,2a,md5. This identifies which encryption modules are available to you. It should match what you saw in the /etc/security/crypt.conf file.

35. Locate the line that reads: #CRYPT_ALGORITHMS_DEPRECATE=__UNIX__

36. Remove the # symbol in front of this line:

    CRYPT_ALGORITHMS_DEPRECATE=__UNIX__

    This will instruct the kernel to ignore the built-in "__UNIX__" algorithm which is the 56-bit encryption you are replacing.

37. Go to the bottom of the file and locate a line that reads: CRYPT_DEFAULT=__UNIX__

38. Place a # symbol in front of this line: **#** CRYPT_DEFAULT=__UNIX__

39. Enter on a new line: **CRYPT_DEFAULT=md5** & press **ENTER**

40. This entry sets up sunmd5 as the new standard for the system.

41. Save this file and close the editor.

42. Labuser3 has a 10 character password and a 13 character password value. Look once more at the /etc/shadow file. Type: **cat /etc/shadow**

43. Do the password hash values for labuser2 and labuser3 look similar in

    length? ______________

44. Click back in the terminal window that is running as labuser3 ('$' prompt). Change the password once more. Type: **passwd**

45. Enter the existing password: **i82loveUNIX**

46. When prompted put in a new password. Type: **uloveUNIX2**

47. Repeat it again. Did the password change? ______________________________

48. Click back in the root window ('#' prompt) and look at the /etc/shadow again. Type: **cat /etc/shadow**

49. Does the password hash format for labuser3 look significantly different

    from the one for labuser2? ____

50. Which encryption algorithm was used for labuser2 and which one for labuser3?

    ____

51. If labuser2 or labuser1 changes their password, which hashing algorithm

    will be used to create the password hash? ______________________________

You've just upgraded your system's encryption scheme to 128-bit encryption.

End of PE

PE 7 - Core Dumps/Buffer Overflows/Automount

The purpose of this PE is to familiarize the student with three of the more common configuration settings affecting system security: core dumps, buffer overflows, and auto-mounting file systems.

1. Core Dumps:

   a. A core dump is a file containing the contents of memory at the time a program or operating system crashed. It can be loaded into a specialized program called a "debugger" to get some idea of what the program was trying to do when it crashed.

   b. In the early days of computers, memory was made of small doughnut-shaped soft-iron rings with very small wires passing through them. These small rings were called cores. The term "core" became synonomous with "memory". So, a "core dump" is a file which contains a copy of all the ones and zeroes copied directly from memory so as to make an exact copy of what was in memory at the time of the crash.

   c. Core dumps can have clear text passwords and keys in them. Hackers look for core dump files (files with "core" in the name). Password hashes and IP addresses's are gained through doing this. Solaris has a utility called coreadm which allows the administrator to manage these sensitive files. There is also a /etc/coreadm.conf file.

   d. The coreadm program shows dump process settings New systems create dumps with permissions set to 600. Core dumps are created either global (in the case of an operating system crash) or by process. Check your system settings. Type: **coreadm**

   e. The dumpadm application shows your dump configuration. To see your dump configuration, type: **dumpadm**

   f. A big part of the core dump problem stems around content. By eliminating content, we can eliminate that part of the problem. To prevent any process from creating a core dump, modify the /etc/system file. The /etc/system file allows us to modify kernel activity without recompiling the kernel. Type: **dtpad /etc/system**

      Go to the bottom of the file and enter on the last line:

      set sys:coredumpsize=0

      Press **ENTER**, save and close the editor.

2. Buffer Overflows:

   a. Now lets deal with the issue of buffer overflows. a buffer overflow is an anomalous condition where a program somehow writes data beyond the allocated end of a buffer in memory. Buffer overflows usually arise as a consequence of a bug. One ramification of the overflow is that valid data can be overwritten as a result. Buffer overflows are also a commonly exploited computer security risk—since program control data often sits in the memory areas adjacent to data buffers, by means of a buffer overflow condition the computer can be made to execute arbitrary (and potentially malicious) code that is fed to the buggy program as data. Buffer overflows are the most common attacks you'll see these days. They take advantage of bad programming. Once again, we'll use the /etc/system file to modify our systems behavior. Type: **dtpad /etc/system**

b. Go to the bottom of the file, below the coredumpsize entry and add the following entries on two new lines of the /etc/system file:

```
set noexec_user_stack=1
set noexec_user_stacklog=1
```

NOTE: If you are using a Intel-based version of UNIX, close the file without saving. This version of UNIX does not have the program variable required to perform this function. If you are using hardware specifically designed for UNIX, save and exit the editor.

c. The stack = 1 entry disables the execution of code directly on the stack by forcing a core dump on any buffer overflow attempt. The second command causes any attempt to run code from the stack to be logged. This fix could possibly stop some legitimate programs from running properly. Some programs need to be able to execute code in the stack. The stack is a reserved area of memory used to keep track of a program’s internal operations, including functions’ return addresses, passed parameters, and so on. A stack is usually a LIFO (last in/first out) data structure. The last item added is the first item used.

3. Auto-Mounting File Systems

a. UNIX and UNIX-like operating systems assign a device name to each device, but this is not how the files on that device are accessed. UNIX creates a virtual file system, which makes all the files on all the devices appear to exist under one hierarchy. This means, in UNIX, there is one root directory, and every file existing on the system is located under it somewhere. Furthermore, the UNIX root directory does not have to be in any physical place. It might not be on your first hard drive - it might not even be on your computer. UNIX can use a network shared resource as its root directory.

b. To gain access to files on another device, you must first inform the operating system where in the directory tree you would like those files to appear. This process is called “mounting” a file system. For example, to access the files on a CD-ROM, informally, one must tell the operating system "Take the file system from this CD-ROM and make it appear under the directory /mnt". The directory given to the operating system is called the *mount point* - in this case it is /mnt. The /mnt directory exists on all UNIX systems, and it is intended specifically for use as a mount point for temporary media like floppy disks or CDs. It may be empty, or it may contain subdirectories for mounting individual devices. Generally, only the administrator (i.e. root user) may authorize the mounting of file systems.

c. UNIX-like operating systems often include software and tools that assist in the mounting process and provide it new functionality. Some of these strategies have been coined "auto-mounting" as a reflection of their purpose.

d. In many situations, filesystems other than the root need to be available as soon as the operating system has booted. All UNIX-like systems provide a facility for mounting filesystems at boot time. System administrators define these filesystems in the configuration file fstab, which also indicates options and mount points.

e. In some situations, there is no need to mount certain filesystems at boot time, although their use may be desired thereafter. There are some utilities for UNIX-like systems that allow the mounting of predefined filesystems upon demand.

f. Removable media have become very common with microcomputer platforms. They allow programs and data to be transferred between machines without a physical connection. Two common examples include CD-ROMs and DVDs. Utilities have therefore been developed to detect the presence and availability of a medium and then mount that medium without any user intervention.

g. By default, most file systems are mounted with Read/Write permissions for everyone, as well as allowing the ability to create SUID files. To protect your system, file systems should be mounted as Read-Only, and no SUID. Be very careful when using the mountall or umountall commands.

h. Now we'll check to see if our mounted drives are being mounted with the "nosuid" option. It is normally easy to figure out if automounting is occurring. If you insert a CD into the CDROM drive and the file manager creates a file window for you, automounting is happening. Check the Volume Manager daemon to see which file is responsible for handling the loading of the various mounted volumes. Type: **cat /etc/vold.conf**

   Do you see where the removable media mount program (rmmount) is called

   on to handle mounted media? ____

i. Check the rmmount.conf file: **cat /etc/rmmount.conf**

j. Can you locate the “nosuid” setting? ____________________ What file systems

   are supported? ____
   (file systems usually end in fs)

k. Is the computer being protected against homemade hacking programs that

   utilize the SUID file setting? ____

End of PE

PE 8 - Packages & Patches

This PE will help teach the student how to identify the software packages that have been loaded on the system and how to install new ones. This PE will also teach the student how to identify the patches that have been installed on the system and how to update the system with new patches.

Before beginning the PE, ensure that you have the CDROM provided by the instructor. The CDROM contains the following packages: 114134_0.zip; 114137_0.zip; jetpkg.bz2; sol_9_x8.zip.

1. Use the package information command: **pkginfo**

2. You should see a long scrolling list of packages that are loaded to your system. Pick one and look at its individual information:

   Type: **pkginfo –l SUNWdtdst**

   This command should display the information about the desktop application package. Each individual package has its own pkginfo file that can be queried for information.

3. Take a look at the patches loaded on your system by using both the showrev and patchadd commands

   showrev –p
   patchadd –p

   Both commands show that no patches have been loaded.

4. Insert the cdrom the instructor gave you. It will automount to your system. Open up the file manager if cdrom content does not display on the desktop. (right click on the desktop, select files, and then file manager). The cdrom will mount as either CDROM0 or New

   Access the cdrom: **cd /cdrom/cdrom0**

5. List the contents: **ls**

   Some are patches and some are software packages. All are compressed with some sort of encryption.The files “114134_0.zip” and “114137_0.zip” are patches for mail and sendmail

6. Make the default patching directory: **mkdir /var/spool/patch**

7. Copy the patches over to the default directory:

   Type: **cp /cdrom/cdrom0/114* /var/spool/patch**

8. Change over to the patch directory: Type: **cd /var/spool/patch**

9. Unzip the patches. Type:

   unzip 114134_0.zip
   unzip 114137_0.zip

10. Install the patches. Type:

    patchadd 114134-01
    patchadd 114137-04

    You should see the patches load after each patchadd command.

11. Run either **patchadd –p** or **showrev –p** to see the current list of installed patches. You should see the two patches you just loaded.

12. Go back to the root home directory. Type: **cd**

13. Double check that you are in the / directory. Type: **pwd**

14. List the cdrom components. Type: **ls /cdrom/cdrom0**

15. You will now install a package to enable the Solaris Jump Start imaging program. First we need to copy the package to the default package directory. Type:

    cp /cdrom/cdrom0/jetpkg.bz2 /var/spool/pkg

16. Change to the /var/spool/pkg directory and uncompress the file. Type:

    cd /var/spool/pkg
    bunzip2 /var/spool/pkg/jetpkg.bz2

17. Do a listing to see if the unzip was successful. Type: **ls**

18. Add the package. Type:

    pkgadd –d /var/spool/pkg/jetpkg

    A list of packages will display, press **ENTER** to accept all. Answer **yes** to all questions.

19. Run the package info command. Type: **pkginfo**

    Check for the packages you just added. You should see packages like JetEXPLO, JetFLASH, JetSAN, JetSDS, and JetVTS. Congratulations you have installed software

20. Now lets upgrade our overall encryption package. Type:

    cp /cdrom/cdrom0/sol_9_x8.zip /var/spool/pkg

21. Now unzip the package. Type:

    Unzip /var/spool/pkg/sol_9_x8.zip

    This creates a directory of sol-9-x86-crypto

22. Lets view the packages available. Type:

    ls /var/spool/pkg/sol-9-x86-crypto/Encryption_9/i386/Packages

    You'll see a list of 3 packages that deal with encryption.

23. Install the packages. Type:

    pkgadd –d /var/spool/pkg/sol-9-x86-crypto/Encryption_9/i386/Packages

    You see a choice of 3, press **ENTER** to select all 3. Answer **yes** to each prompt as you install all 3 When you've installed all 3, hit the "**q**" key to stop the install loop.

24. Use the pkginfo utility to find out what we installed. Type:

    pkginfo

    You should see SUNWcrman (Solaris Crypto Man Pages), SUNWcry (Crypt utilities), and SUNWcryr (Solaris Root Crypto) have been added to the package list.

    You have just installed software updates to improve current encryption levels and to enable stronger encryption in programs such as IPSec.

25. Use the cd command to get back to the / directory. Type: **cd**

End of PE

PE 9 - Remote Sessions/Ports/Services

This PE will familiarize the student with identifying remote connections, disconnecting remote sessions, identifying ports and services that are open and some of the dangers of leaving unnecessary ports open and unnecessary services running. Some commands will be typed within a TCP session.

This PE will have you "telnet" to various ports and observe some of the disadvantages of leaving them open when certain services are running on them.

1. Break down into groups of two. Each partner should telnet to their partner's workstation. Type: **telnet** (**partner ip**)

   example: #telnet 147.51.217.151

   Were you greeted by your partner's banner(s)? ___________________________

2. Login as **labuser2** with a password of **student2**.

3. Change to root with su and root's password. Type: **su root**

4. Use the netstat command to determine the connections on your partner's computer. The netstat daemon will report network connections, routing information, and statistics. Remember, you are viewing your partner's computer connections. This will show all of the tcp protocol activity. Type: **netstat –a –P tcp**

5. You should see a lot of ports listening or in use. Find the line where telnet and your IP or machine name are listed. One is the connection from you. The other is your partner connecting to you.

6. Find the process that is your connection to your partner. The in.telnetd daemon makes the inbound connection possible. Search for processes that involve it. Type: **ps –ef | grep in.telnetd**

7. The first number listed, from the left, is the process id of the telnet connection you are using to connect to your partner. Kill this process. Type: **kill -9 (PID)** (example: # kill -9 488)

8. Did you just get notification that your connection was disconnected by the

   foreign host? _____________________________ You just disconnected yourself.

9. Check to see if the process still exists. It should be gone from both machines.
   Type: **ps –ef | grep in.telnetd**

10. Use netstat to see if the connection is still there. The connection you will be viewing is that of your own workstation. You are no longer connected to your partner.
    Type: **netstat –a –P tcp**

11. The connections should be gone or in a TIME_WAIT status. A TIME_WAIT status indicates that your tcp connection ended ungracefully. Your computer is still waiting for the command to close the connection. It will time out eventually. When you used kill -9,

you forced the connection to close on your partner's machine. No command was ever sent back to your machine. The connection was left open on your machine.

12. Check to see what ports and services are open. The following set of commands will show services and ports that are open on your computer. Type (in order):

    netstat –a > file1
    netstat –an > file2
    sdiff file1 file2

13. Look for your connection to your partner. It should no longer be listed as being in a TIME_WAIT status. It should be gone.

14. Look at the IPv6 TCP section and notice the services on the left with the corresponding port number near the center of the line. There are services open expecting a connection from the outside. Notice the listening status. If you browse the entire list you'll see the UDP: IPv4, UDP: IPv6, TCP: IPv4, and TCP: IPv6 ports and services.

15. Go look at what can be learned from the FTP port of 21.

    Type: **telnet localhost 21**

    Solaris 9 users will also type:
    USER labuser1 (press **ENTER**)
    PASS student1 (press **ENTER**)

    a. What information does the system give you once you connect to port 21?

       ____

    b. Type: **help**. 1. You should see a bunch of commands that are not meant for public consumption. Manipulation of these commands enables hackers to take over systems.

    c. Type: **quit**

16. Look at what can be gained by going to port 13. Type: **telnet localhost 13**

    It won't let you connect but it will give you some information. What did you learn?

    ____

17. Look at what can be gained by going to port 79. This is the port for finger. Type: **telnet localhost 79**

    Press **ENTER** twice. What did you learn?

    ____

18. Look at what port 25 has to offer. This is the port for SMTP. Port 587 is the submission port for sendmail and works similar. Type: **telnet localhost 25**

a. What did you learn?

____

b. Type: **help**. You see see some commands that you could use to manipulate email.

c. Type: **vrfy labuser1**

d. Type: **vrfy labuser2**

e. Type: **vrfy labuser8**

f. You'll notice that in steps c and d, you found out about labuser 1 and 2's email accounts. Step e indicated that there wasn't a labuser8 on the system. This is one of the ways hackers verify or identify valid users on a system.

g. Type: **quit**

19. Take a look at another unnecessary port. Type: **telnet localhost 7**

    a. Press **ENTER**

    b. Type: **quit**

    c. See how the port repeats what it gets. This is the echo service. It doesn't serve much purpose and is liable to utilize large amounts of process time when given large amounts of data to repeat.

    d. Type: **^]** (Control right-bracket)

    e. To get out of the telnet prompt ">", type: **^D** (Control D)

20. Another seemingly harmless port is 19. Type: **telnet localhost 19**

    a. Seems like you've tapped into a treasure trove of useless data. This is the character generator service. You should notice that the CPU meter on the bottom right is starting to show red. What do you think would happen if you redirected this output to your port 7?

    ____

    b. To stop the data flow, type: **^C**

    c. Type: **^]**

    d. To get out of the telnet prompt ">", type: **^D**

    If you are not back to the # prompt, you can try various combinations of the above control characters or close and reopen your window.

21. Let's summarize what you've learned from these unnecessary ports by matching the correct fact with the port that you visited.

    Choices: Port 7, 13, 19, 21, 25, 79

    a. What port told us what time zone the computer was in? ________________

    b. What port told us who was logged on? _______________

    c. What port repeats data sent to it? ____________________________________

    d. The port that tied up our processor was? ______________________________

    e. The port that helped us determine if the computer was a mirror site

       was? _____________

    f. The port that told us the version number of an often hacked program was? _____________

    g. The port that told us the machine name was? ___________________________

    h. The port that told us our operating system was? ______________________

    i. The port that helped us id all the users was? _________________________

    j. What two ports could be used to tie up resources and create a denial of

       service attack? ____

22. Lets take a look at the traffic on the wire utilizing the "snoop" utility that is provided with Solaris 8/9. Snoop is a tool that really has no basis for existence as System Administrator's should not be analyzing traffic in this manner. Utilize your partnership from earlier in the PE to complete these steps. To start, both partners need to capture some network traffic.

    a. Type: **snoop**

       After a slight delay you should see quite a bit of packet information. This is being generated by your classmates' computers. Your network card is capturing all broadcast traffic on the network.

    b. Type: **^C** to stop the flow of data. Take a couple of seconds to familiarize yourself with the format.

    c. Partner B now should ping another computer in the room. Type: **ping –s (some IP)** ("some IP" represents an IP number, ping –s sends that IP a ping each second, do not type "(" or ")")

    d. Partner A will now capture the traffic between the two computers in a file called cap.

       Type: **snoop –o cap (partner B IP) some IP**

Example: snoop –o cap 147.51.217.151 147.51.217.166

(this captures the traffic between your partner and the IP they pinged in a file named cap)

e. Partner A should see a counter on the command line increase numerically. When Partner A has received a comfortable amount of packets (over 40), both partners need to type: **^C**

23. Partner A can now examine the packets.

    Type: **snoop –i cap –p 0,40**

    (this allows you to view the first 40 packets of the file cap)

    You should see the ping (ICMP) packets between the two machines. You may see a few packets of another type. Do you see the echo requests and echo replies?

24. Partner A can now examine the details on any packet. This script looks at the 16th packet. You may substitute the number 16 for any of your packet numbers.

    Type: **snoop –i cap –v -p16** (Notice the nice breakdown of the packet information)

25. If you like, reverse your roles so both partners can capture traffic.

26. Let's see if a regular user can use snoop to spy on fellow users.

    a. Both partners type: **exec login**

    b. Enter **labuser1** for login and press **ENTER**

    c. Enter **student1** for the password and press **ENTER**

    d. Type: **snoop**

    e. Did it work or were you denied permission? ____________________________

27. Check out your permissions, type: **ls –l /dev/iprb0**

    a. What was the outcome? Is this a link? _________________________________

    b. Check the target of the link:

       Type: **ls –l /devices/pci@0,0/pci8086,244e@1e/pci8086,50@2:iprb0**

       (Link targets will vary from system to system.)

    c. What are the permissions on the actual network adapter? ______________

28. Snoop needs to be deleted or disabled. Is it disabled for user use?

    ____

29. Return to the root account -- Type: **exec login**

30. Type: **root** and press **ENTER**.

31. Type: **student** and press **ENTER**.

32. The terminal window will close and you will be back at root.

End of PE

PE 10 - SSH

The following steps will take the you through a demonstration of how Secure Shell (SSH) protects your communications from prying eyes (snooping). The PE is designed around working in groups of 3. Two (partners A and C) will be setting up telnet sessions and SSH sessions. The third team member (partner B) will be the man in the middle, snooping the traffic between the other two. Partners need to perform their respective tasks in the specified order to avoid incorrect results.

WHEN YOU SEE “partnerA-IP” OR “partnerB-IP” OR “partnerC-IP”, SUBSTITUTE THE IP ADDRESS OF THE APPROPRIATE PARTNER. DO NOT TYPE ‘partnerA-IP’, etc. THE IP ADDRESSES FOR THE CLASSROOM ARE WRITTEN ON THE CHALKBOARD.

1. Partner B: Type: **snoop –o cap partnerA-IP partnerC-IP** to set up the traffic intercept. Snoop will scan for traffic between the two IPs and put the captured packets in a file called “cap”.

   Example: snoop –o cap 147.51.217.151 147.51.217.153

2. Partner A: Type: **telnet partnerC-IP** to connect to partner C.

3. Partner A: Log in using the **labuser2** account (password **student2**).

4. Partner A: You are going to type a message to partner C. Type: **who** to identify the root login at either pts/4 or pts/5.

5. Partner A: Type: **write root pts/4** or **pts/5**

   Example: write root pts/4

   The prompt will be a line without a prompt.
   On this line, type: **UNIX is a wonderful operating system**

   Press **ENTER** and then **^D**

6. The message should print in partner C’s terminal window. A packet counter should be continuously incrementing on partner B’s screen. Partner B: Type: **^C** and stop the capturing of packets.

7. Partner B can view the packet history by reading the file called “cap”.

   Type: **snoop –i cap** (This displays the contents of “cap” file.)

   Partner B should see a printout of all the packets that A and C exchanged. Notice that on the right you see the ASCII values of some of the packets.

   Do you see the message partner A typed? ___________________________________

   Do you see the characters from the message echoed back to partner A (from

   partner C)? _______________________

Can you see the password “student2” laid out vertically in packet sequence? ________

8. Partner B: Pick a packet at random and plug its number into the following script:

   Type: **snoop –i cap –v –p**(**packet #**) (This analyzes the selected *packet #.*)

9. Do you see the breakdown of the packet? __________________________________

10. Partner A: Type: **exit** and then press **ENTER** to disconnect from partner C.

11. Partner B can now set up capturing again, this time between partner C and partner A. Their roles will be reversed.

    Type: **snoop –o cap2 partnerC-IP partnerA-IP**

12. Partner C will now set up a secure shell connection to partner A.

    Type: **su labuser2**

    Type: **ssh** (**partnerA IP**)

    If prompted about the RSA key, ignore it by typing **yes** and pressing **ENTER.**

13. When prompted for the password, Partner C will type: **student2**

14. Partner C, type: **who**

15. Partner C, You are going to type a message to partner A. Type: **who** to identify the root login at either pts/4 or pts/5.

    Type: **write root pts/4** or **pts/5**

    The prompt will be a line without a prompt. On this line type: **UNIX is a lot better than the Windows operating system!**

    Press **ENTER** and then **^D.**

16. The message should print on partner A’s screen. The packet counter on partner B’s screen should be continuously incrementing. Partner B: Type: **^C** to stop the capture.

17. View the packet history by reading the file called cap2.

    Partner B, Type: **snoop –i cap2**

    Partner B should see a little bit different output than before. The packet capture doesn’t show any of the ASCII value text and there is additional sequence and coding numbers.

    Do you see any of the previous ASCII values? _____________________

18. Partner B can pick a packet at random and plug its number into the following script.

    Type: **snoop –i cap2 –v –p**(**packet #**)

19. Do you see the breakdown of the packet? _____________________________ Notice that the previous telnet capture showed plaintext telnet lines that are now gone.

20. Partner C, Type: **exit** and press **ENTER** to disconnect from partner A.

21. What do you answer when asked the question, "What is the benefit of running secure shell rather than telnet?"

    ____

    ____

End of PE

PE 11 - cron

The purpose of this PE is to familiarize the student with UNIX's main scheduling utility. "cron" is a chronological scheduler that allows you to run programs or scripts unattended.

Cron is the scheduling utility of choice on UNIX systems. If you use it, you have “cron jobs” pending or in the works. To use cron, you edit the cron table using crontab with one of the following flags:

-e (edit)
-l (list)
-r (remove)
-d (kills all cronjobs)

1. Make sure your system date is correct. Type: **date**

2. If the date and time is incorrect, you can reset it by typing: **date nnddhhmm**

   nn = month
   dd = day
   hh = hour
   mm = minutes

3. Check the date again by typing: **date**

4. Open up the cron table for editing by typing: **crontab -e**

   The format for a cronjob is

   minute, hour, date, month, day, file or script

   minutes are counted from 0 to 59
   hours are counted from 0 to 23
   date is counted from 1 to 31
   months are counted from 1 to 12
   days are counted from 1 to 7

   Separate entries with a space. Multiple entries are allowed in each field.

5. On the last line of the cron jobs table, create a new line to run a script that hunts for user created ~/.rhosts files. Make the script so that it runs every day at 10:55am and saves to a file in the root directory called "trusts".

   Type: **55 10 * * * find / -name .rhosts > /trusts**

6. Click on "file" and select "save (needed)".

7. Close the editor. Your script should now be set to run at 10:55am everyday.

8. Use of the cron utility is controlled through cron.deny and cron.allow files. Take a look at the cron.deny file.

   Type: **cat /etc/cron.d/cron.deny**

   Are any of the normal users listed? ___________________________________

9. Lets add labuser1 and labuser2 to the cron.deny file.

   a. Type: **dtpad /etc/cron.d/cron.deny**

   b. The users listed cannot use cron. On lines by themselves, add both labuser1 and labuser2 to the bottom of the list.

   c. Click on "file" and select "save (needed)" to save the list with changes.

10. Double-check your changes. Type: **cat /etc/cron.d/cron.deny**

11. A cron.allow file is not normally created. Check for its existence.

    a. Type: **cat /etc/cron.d/cron.allow**

    b. A cron.allow file is usually used when all users are included in the cron.deny file. The cron.allow would allow only the users listed to use cron. The system reads the cron.allow file prior to the cron.deny file. Do not be surprised if you do not have one by default.

End of PE

PE 12 - X-Windows

Hackers can exploit your X-Windows sessions in various ways. The major problem with X is that its security model is an all or nothing approach. Once a client is granted access to an X server, pandemonium is allowed. X clients can capture the keystrokes of the console user, kill windows, capture commands for display elsewhere, and remap the keyboard to use nefarious commands no matter what the user types. Most problems stem from a weak access control paradigm.

Pair up the students so that they are in groups of two. Select one to be partner A and one to be partner B. All references to partnerIP will reflect the actual IP number of the partner's machine.

1. Each student type: **xhost +** (plus "+" sign).

2. Notice what the computer tells you about access control lists. What does it indicate to you?

   ____

   ____

3. Each student type: **xhost -** (minus "-" sign).

4. Notice what the computer tells you about access control lists. What does it indicate to you?

   ____

   ____

5. Each student type: **xhost +partnerIP**

   "example: #xhost +147.51.217.157"

6. Each student type: **xhost**

7. Who is listed in your access control list for the Xserver?

   ____

8. Each student type: **xhost -partnerIP**

   "example: #xhost -147.51.217.157"

9. Each student type: **xhost**

10. Who is listed in your access control list for the Xserver?

    ____

11. Each student type: **xhost +**

12. Partner B, start a program on the computer. Type: **xclock &**

13. Partner A, type: **xlsclients –a –l –display partnerB_IP:0**

    "example: xlsclients -a -l -display 147.51.217.155:0"

14. Record the names of the clients (open windows) on your partner's machine.

    ____

    ____

    ____

15. All the windows that are open on partner B's computer have an ID (0x??????????). What is the window ID for the window named xclock?

    ____

16. Partner A, Type: **xkill –id** (xclock ID from question 15) **–display partnerB_IP:0**

    "example: #xkill -id 0x40000040 -display 147.51.217.155:0"

17. What happened on your partner B's display?

    ____

    ____

18. Partner A, type: xterm –display partnerIP:0

    "example: #xterm -display 147.51.217.155:0"

19. What happened to the partner B's screen?

    ____

    ____

20. Partner B, click on the xterm window and Type: **cat /etc/hosts** and press **ENTER**. Did the IP in the hosts file match the IP of partner A or

    partner B? __________________________

21. Partner B, Type: **reboot** and press **ENTER**.

22. What happened?

    ____

23. What does this tell you about the potential danger of xhost files?

____

____

24. When communicating with your partner's screen were you ever prompted for a password? Did you have to utilize a .rhosts or hosts.equiv file? Who does the "xhost +" command allow Xserver access to?

____

____

____

IAW HQ DA SAIS-IAS message DTG R 050951Z MAR 99, disable the xhost command. Xwindows uses TCP ports 6000-6063. Blocking these ports will also stop outside access.

DO NOT permit Xwindows access from arbitrary hosts. To secure the system, remove all instances of the 'xhost +' command from the system-wide Xsession file, from user .xsession files, and from any application programs or shell scripts that use the X window system.

If you must allow access to your X server, specify each server by IP address. Keep in mind that any user on that server can connect to your X server and snoop away. DO use xauth to replace xhost. If you use ssh, the xauth is automatic. READ the manual pages for xauth and Xsecurity. Use this information to set up the security level you require.

USE the X magic cookie mechanism MIT-MAGIC-COOKIE-1 or better. With logins under the control of xdm, you can turn on authentication by editing the xdm-config file and setting the DisplayManager*authorize attribute to true. When granting access to the screen from another machine, use the xauth command in preference to the xhost command.

MIT-MAGIC-COOKIE-1: Shared plain-text "cookies"
XDM-AUTHORIZATION-1: Secure DES based private-keys
SUN-DES-1: Based on Sun's Secure RPC system
MIT-KERBEROS-5: Kerberos version 5 user-to-user

End of PE

PE 13 - Network Services

The purpose of this PE is to familiarize the student with the UNIX default network services & explain the procedures to safeguard network access

1. First, locate the daemons that allow for the use of the remote "r-" commands. Type: **find /usr/sbin –name "in.r*" > rcmds**. (This creates a file named "rcmds" and places in it a list of all commands starting with "in.r".)

2. Type: **cat rcmds**

3. Which r-command daemons would you want to delete or disable (see page 21)?

   ____

   How would you do this?

   ____

4. The greatest vulnerabilities in your system are network services. You need to know which services are currently running on your system.

   Type: **dtpad /etc/inetd.conf** (Most of these services are unnecessary.)

   a. Any line not preceded by a '#' symbol is a service that is running on this computer.
   b. The first word on each line is the name under which the service runs.
   c. The second column tells you the type of socket.
   d. The third column tells you the protocol.
   e. The fourth column tells you the wait status.
   f. The fifth column tells you the UID that it runs as.
   g. The sixth column tells you the full path to the actual daemon program.
   h. The last column allows you to specify arguments.
   i. What are some of the services that are currently running by default?

      ____

      ____

   j. Under what names do in.rshd, in.rexecd, and in.rlogind run?

      ____

      ____

5. Disable the telnet service only. To disable telnet, comment it out by placing the '**#**' symbol (without the single-quotes) at the beginning of the line. Note that there are two lines beginning with telnet. Only one of these is lowercase like the service name. Additionally, the first telnet line already begins with '#'. Save the file and exit the editor.

NOTE: Commenting out a service in the inetd.conf DOES NOT automatically terminate the service. You must stop and restart the inetd daemon so the new configuration file is initialized.

6. Next, you will terminate the inetd[aemon] and then restart it. To do this, you must first determine the inetd processor ID (PID) using the ps command.

   Type: **ps –ef | grep inetd**

   Write down the PID of inetd. The PID is found in the second column of the line which ends with the /usr/sbin/inetd filename. Of the two columns of numbers, it will be the left one.

7. Type: **kill –HUP PID** (where PID is the PID of the inetd from step 6). HUP is the hang-up argument which stops and restarts a service.

   For example: kill -HUP 169 or kill -HUP 170

8. Try to start telnet: Type: **telnet localhost**

   Write down the message received.

   ____

   (Note: If you did not receive an error message when the telnet command was executed then the kill command executed in step 7 did not work correctly. To ensure the inetd has read the updated /etc/inetd.conf type: kill –9 (PID of inetd);, then type inetd –s. This will restart the inet daemon using the modified configuration file. Try step 8 again. A word of caution: using kill –9 is NOT the recommended way to restart processes.)

9. In addition to the /etc/inetd.conf, you can edit the file that specifies which service runs on a specific port. The /etc/services file lists the available ports by service name, port number/protocol.

   Type: **dtpad /etc/services**

10. Comment out the telnet service. Add the '**#**' symbol to the beginning of the line that contains the service you want to disable. When done, save the file and exit the editor.

11. Try to start telnet. Type: **telnet localhost**

    Write down the message received. Is the message different from the message received when the services had simply not been started?

    ____

12. Just as in Windows, UNIX also has the ability to establish trust relationships. However, it is more dangerous in UNIX because every user can establish a trust. First, let's determine if the administrator has set up a trust.

    Type: **ls –l /etc/hosts.equiv**

Does the file exist? When is a hosts.equiv file allowed? (Page 22)

____

____

13. In Step 12, we addressed the fact that the system administrator may establish trusts. UNIX will also allow ANY user to establish trusts with other users, either internal or external. This is done through the creation of a .rhosts file. Are there any .rhosts files on your system?

    Type: **find / -name .rhosts**

    Do any .rhosts files exist on the system? What would it mean if a .rhosts file did exist (page 22)? Name 2 methods to increase security from the vulnerability associated with .rhosts files?

    ____

    ____

    ____

14. Switch user to the labuser1 account. Type: **su labuser1**

15. Now, let's try to compromise a default system account.

    Type: **rlogin -l lp localhost**

    (NOTE: The lp account does not have a password. When asked for a password, abort the rlogin attempt by typing a **^**D.)

16. Were you able to log in as lp? Why?

    ____

    ____

17. Type: **exit**. This should put you back into 'root.' (Note the # prompt).

18. Create a .rhosts file in the lp home directory.

    Type: **dtpad /usr/spool/lp/.rhosts**

    Enter the following line as the first line:

    + +

    (The line should be a plus sign followed by a space, followed by a plus sign, followed by pressing the **ENTER** key.)

    Save the file and close the editor.

19. For a .rhosts file to work, the directory owner must have at least read permission on the file. Does the lp account have read permission for the .rhosts file? If not, why not?

    ____

    Type either **chown lp /usr/spool/lp/.rhosts** to change ownership of the file to lp, or use **chmod -o+r** to add read permissions to others.

20. Now, let's try to compromise the lp account. Type: **su labuser1**

21. Now try to rlogin to the lp account. Type: **rlogin -l lp localhost**

22. Were you able to log in? (use the **id** command). Why is use of a .rhosts file a security risk? What does the + + mean (page 22)?

    ____

    ____

23. Type: **exit** (to quit the rlogin session.)

24. Type: **exit** (to quit the su labuser1 session.)
    Remain logged in as root. (NOTE the # prompt.)

25. Let's disable the lp account login by giving it a false shell.

    Type: **cat /etc/passwd**

    Does the lp account have an authorized shell? Should you be able to login as lp?

    ____

    ____

26. Type: **usermod –s /bin/false lp**

27. What did you just do?

    ____

    ____

28. Type: **cat /etc/passwd** What's different about the lp account?

    ____

    ____

29. Type: **su labuser1**

30. Now try to rlogin as lp. Type: **rlogin –l lp localhost**

    Were you able to login now? (use the id command) Why or why not? (Look carefully at the messages from the rlogin).

    ____

    ____

31. Type: **exit** (this will put you back at root)

32. Remove the false shell from the lp account (hint: **usermod -s /bin/sh lp**)

33. Edit the pam.conf file: **dtpad /etc/pam.conf**

    Locate the lines for rlogin and rsh. Place a # symbol in front of each line that has the rhosts reference. They should look like this when you are done if using Solaris 9 (Solaris 8 is slightly different):

    ```
    #rlogin        auth   sufficient     pam_rhosts_auth.so.1
    #rsh           auth   sufficient     pam_rhosts_auth.so.1
    ```

34. Save the file and close the editor.

35. Type: **su labuser1**

36. Try and rlogin again as you did in step 30. (hint: **rlogin -l lp localhost**)

    What was different this time as compared to step 30? Why do you think you were prompted for a password?

    ____

    ____

37. How can system accounts be secured from .rhosts logins?

    ____

    ____

38. Type: **exit** (to quit the su labuser1 session.)

End of PE

PE 14 - Network Settings

The purpose of this PE is to familiarize the student with default network settings and necessary security adjustments.

1. NFS is the Network File System. With NFS, a UNIX/Linux user can share files with other users. The security involved is often weak. Check the default settings on your system.

   Type: **cat /etc/nfssec.conf**

   What is the default security setting? ____________________________________

   The present setting only requires a request to have been generated from the proper system and user. These can easily be spoofed. Stronger settings are available. Some additional options are: none; dh (Diffie Hellman); Kerberos. Selecting dh would greatly increase security by requiring DES encryption. Kerberos is normally supplied only in the client form.

2. Files are shared out by placing them in the distributed file system table (/etc/dfs/dfstab). Files we connect to are identified in the virtual file system (/etc/vfstab). Take a look at the default distributed file system. Nothing should be shared by default. Type: **cat /etc/dfs/dfstab**

3. Sendmail is another important network service that runs on a majority of UNIX systems. Many vulnerabilities relate to buffer overflow and input validation attacks. In addition, sendmail allowed attackers to pipe commands directly to sendmail for execution. It is also possible to gain privileged access. Sendmail is a daemon that runs at root level. Older versions of sendmail allow spammers to relay mail through your system. Newer versions are found at www.sendmail.org. When a person connects to port 25, sendmail has a habit of broadcasting its version number. This is something best left for the hacker to guess. Sendmail is an often hacked program and it does no good to share version information with the world.

   Type: **telnet localhost 25**

   Does sendmail advertise the sendmail version info? ______________________

   You do not have a prompt anymore and you are in the sendmail shell. To exit the sendmail shell, type: **quit**

4. Let's remove the sendmail version info from public access. Edit the sendmail configuration file.

   Type: **dtpad /etc/mail/sendmail.cf**

   Click on “edit” and select “find/change”. Search for “Smtp”. This will send you to the line for the SmtpGreetingMessage. **Delete everything to the right of the ‘$j’ entry**. This is what causes the version to be broadcast. The next time the system boots up, the sendmail version will not be broadcast.

   A couple of internal sendmail commands allow you to find user account information and in the end, allow you to fingerprint a mail server’s client list. The commands in question are “vrfy” and “expn”. We need to disable those also. At the bottom of the /etc/mail/sendmail.cf file (still open in the editor), add:

O PrivacyOptions=novrfy,noexpn. (press **ENTER** after adding the line).

The next time the mail program is started, the vrfy and expn options will not be available. Save your work in the editor and close the test editor.

If you do not need Sendmail running, turn it off and remove or rename the sendmail startup script /etc/rc2.d/S88sendmail. This will stop sendmail from listening to the network for incoming mail. If you must run sendmail, do not run it as root. Build a chroot environment and run it as a non-privileged user. Sendmail is not required to listen to the network to send mail. You can set up cron so that sendmail will service the queue of outgoing messages on a schedule.

5. Another widely exploited service is FTP. It transmits passwords in the clear, and could give attackers root access to your system. If you do not have a need to run ftp on your server, then it should be disabled. The problem is that file transfers are a way of life for most IT personnel. The FTP protocol is well known and widely used. UNIX systems provide a way of controlling who is using FTP. Check the FTP access list by viewing the ftpusers file.

   Type: **cat /etc/ftpd/ftpusers**

   What users on this system can use FTP? Hint: compare the list against the /etc/passwd list of users.

   ____

   At a minimum, make sure that the root Administrator and the system accounts are listed. This is a list of those accounts that *can not* connect to the system to use FTP. Disable FTP if you are not an FTP server. FTP can be disabled by commenting out the ftp line in the /etc/inetd.conf file. Additional steps can be taken by commenting out port 21 in the /etc/services file. Utilize SSH (secure shell), SHTTP (secure http), VPNs, or other encrypted versions of FTP before relying on the version packaged with the operating system.

6. Let's test access to ftp. At the # prompt, type: **ftp localhost**

   When prompted for Name(localhost:root), enter: **labuser1**

   Enter the appropriate password: **student1**

   At the ftp> prompt, type: **help** (you should see a wide range of commands)

   Exit your ftp session by typing: **quit**

7. Assign a false shell to the labuser1 account:

   Type: **usermod -s /bin/false labuser1**

   Type: **ftp localhost**

When prompted for a name, type: **labuser1**
Type the appropriate password: **student1**

What happened?

____

Type: **quit**

8. Now add the /bin/false shell to the /etc/shells file. **dtpad /etc/shells**

   Add /bin/false on a new line, save the file, and exit the editor.

   At the # prompt, type: **ftp localhost**

   When prompted for a name, type: **labuser1**

   Type the appropriate password: **student1**

   Did the results from step 7 repeat? What happens when you put a false shell in the /etc/shells file?

   ____

   Type: **quit**

   Remove the /bin/false entry from the /etc/shells file and reassign /bin/sh as labuser1's shell. (hint: dtpad /etc/shells) (hint: usermod -s /bin/sh labuser1)

9. Finger is a service that gives us the ability to find out information about other users on other systems. Its very nature is intrusive. It is an unnecessary service that can be turned off.

   The Finger daemon can leak sensitive system information like usernames, home directories, and login patterns. Finger can provide a would-be intruder with a lot of information about your host. Other services such as rusers and netstat may give out similar information. To disable fingerd, put # in front of the line that starts with fingerd in the inetd.conf file.

   First, lets use the usermod command to alter some information about labuser2. Type: **usermod -c "(your name) loves UNIX" labuser2**

   Use finger to retrieve labuser2's account information.

   Type: **finger labuser2@localhost**

   Did you see that the name column says you love UNIX? ____________________

   Lets see where the source info comes from for a finger search.

   Type: **cat /etc/passwd**

Do you see your entries recorded in labuser2's information? ____________

10. Now shut down the finger service. On your own, edit the /etc/inetd.conf file to turn off finger, stop and restart the inetd. If you are having trouble doing this, refer to PE 13 (page 89).

    Test the finger service by executing **finger labuser2@localhost**

    Did you successfully stop the finger service? ____________

    Could you stop other unnecessary network services in a similar manner? ___________

11. Does the finger service still exist for internal users? _____________

    Test it: Execute **finger labuser2**

    How would you disable the command for use by internal users?

    ____

End of PE

PE 15 - NFS

The purpose of this PE is to expose the student to the fundamental weakness of NFS. NFS allows for the accessing or sharing of files over the network. This PE has the student setting up a simple file share, modifying the file, and checking the output for security issues.

This is a two partner PE and both partners will need to be logged in as root on their respective machines. Whenever you see the reference to three x's (xxx), it refers to the last octet of the IP address.

1. Both partner A and B, create a directory: Type: **mkdir /testnfs**

2. Partner A, create a file in the /testnfs directory:

   Type: **dtpad /testnfs/nfsfile**

   Put some text in the file, save it, and close the editor.

3. Partner A, create a host table entry: Type: **dtpad /etc/hosts**

   On a new line, create an entry for your partner's machine. Make sure you press **ENTER** after you make the entry, save the file and close the editor.

   Example: 147.51.217.171 ws217171

4. Partner A now needs to share the file out by making an entry in the distributed file system table.

   Type: **dtpad /etc/dfs/dfstab**

   Create a share entry which gives partner B access. Place the cursor on the beginning of the first free line and make the following entry:

   share –F nfs –o rw=ws217xxx,root=ws217xxx –d "nfs test" /testnfs

   Press **ENTER** at the end of the line; save the file and close the editor.

5. Partner A now needs to enable the share:

   Type: **/etc/init.d/nfs.server start**

6. Now, partner A will set up the capturing of packets to analyze network activity. The first IP to use is Partner B and the second is Partner A.

   Type: **snoop –o nfscap 147.51.217.xxx 147.51.217.xxx**

7. Partner B can now check to see if the share is available. You will use the dfshares command to check Partner A's machine for the share.

   Type: **dfshares 147.51.217.xxx**

8. Partner B should see the share. Before the file can be accessed, it must first be mounted to a directory on Partner B's system.

   Type: **mount –F nfs –o rw 147.51.217.xxx:/testnfs /testnfs**

9. Let's see if it allows for editing.

   Partner B, Type: **dtpad /testnfs/nfsfile**

   Make some sort of modification to the file that Partner A originally created. Once you've done that, save the file and close the editor.

10. While this transaction was taking place, a counter on Partner A's machine should have been recording packets captured. Partner A needs to stop the capture of packets by pressing **^C**. Partner A should be able to open the file and see the new changes.

    Type: **cat /testnfs/nfsfile**

11. Partner A may now view the captured packets . Type: **snoop –i nfscap**

    This will give you a read out of the captured packets. Remember the number of the very last packet captured as we will now look at the packets and their hexadecimal and ascii data. Lastnumber -20 refers to your last packet number and subtract 20 from that. Your screen buffer will not hold all of the packet data so we have to be selective. You may have to play with the packet numbers to view the packet data.

    Type: **snoop –i nfscap –v –x 0 –p**(**lastnumber-20**)**,**(**lastnumber-10**)

    (Example: snoop –i nfscap –v –x 0 –p82,92)

    Scroll up through the packets until you see the packet that corresponds to the modification that Partner B did to Partner A's text file when Partner B saved the file. You'll be looking at the lines starting with NFS.

    The Ether header has the packet number, packet size, and time stamp. The IP header has the source and destination IP numbers. The TCP header has the source and destination port numbers. The RPC header has the user credentials like UID. The NFS header has the file handle number and the packet data payload.

    The packets include a file handle value. This is the number used to keep track of files within NFS. You will also notice that port 2049 is used either as source or destination in each packet. Notice is that information shared in this manner is totally visible to anyone with the ability to capture the packet. This is a clear text file sharing capability.

    NFS should be avoided if possible. Use sftp or scp to transfer or share files. There are a few security options that can be set with NFS that we didn't set. The default mode in Solaris 9 is read only. We had to specifically give root write access for this PE. Normally we would also want to specify the –nosuid option when we mount the directory on the receiving end. Both the share and mount commands allow for specifying authentication mechanisms like Kerberos or DES.

End of PE

PE 16 - Hacked Root

This PE looks at how the root administrator, through account settings, could allow access to an otherwise unprivileged user.

Gaining root access is the primary goal of every hacker trying to get into your system. Root has absolute control over your box. If an attacker gains root access, it's no longer your system. This account should be protected at all costs. Files not owned by root, if run by root, may compromise root access or perform functions that are unwanted.

1. Type: **cat /etc/passwd**

   Which accounts utilize password authentication? Can you determine if an account has a password with the /etc/passwd file? Why?

   ____

   ____

2. Type: **ls –l /etc/passwd** What are the permissions on this file?

   ____

3. Type: **cat /etc/shadow**

   What is the difference between the /etc/passwd and /etc/shadow files?

   ____

   ____

4. Type: **ls –l /etc/shadow**

   What are the differences in permissions? What implication does this have for security? Is shadowing important?

   ____

   ____

5. We've mentioned certain files that determine a user's operating environment, e.g. .profile, .login, and .cshrc. Let's see how these files can affect the execution of programs in a user's environment.

   Type: **ls –la /**

   Does the 'root' account have a .profile, .login, or .cshrc file in its home directory? Why? Should the root account have these types of files?

   ____

   ____

6. Logout of CDE and log back in as labuser1

7. Typically, .profile, .login or .cshrc files have a modification to the PATH statement. Let's see how the PATH environment variable can be a security vulnerability.

   To exploit this vulnerability we need to create a script file. This script file will execute the ls utility with the appropriate argument(s), make a copy of the ksh shell and print two messages on the terminal window.

   Type: **cd /tmp**

   Type: **dtpad ls**

   Add the following lines to the file :

   ```
   #!/bin/sh
   cp /usr/bin/ksh /tmp/.secret_ksh 2>/dev/null &
   chmod 4755 /tmp/.secret_ksh 2>/dev/null &
   echo "Hello world"
   echo "This could be a bad program"
   /usr/ bin/ls $*
   ```

   Save the file and close the editor. Use the following command to change permissions on the file to make it executable.

   Type: **chmod 755 ls**

8. Verify that the script works by executing it.

   Type: **/tmp/ls**

   On the terminal the following should be displayed:

   A. the statement "Hello World" (without "")
   B. the statement "This could be a bad program" (without "")
   C. the files in the /tmp directory should be displayed

9. Minimize the footprints you have left on the system.

   Type: **rm .secret_ksh**

10. Logout of CDE and log back in as root.

11. You have just taken over as System Administrator of an existing system. You would like to configure your environment on log in. Since the bourne shell is your startup shell you will create a .profile file.

    Type: **cp /etc/skel/local.profile /.profile**

    Type: **cat .profile**. Does the .profile have a 'path' statement? If so, is there anything in that path that could be considered a vulnerability?

____

____

Let's modify the .profile to look first at the current directory for executables.

Type: **dtpad .profile**

Move the **.** to the front of the PATH statement.
Originally the line looks like: PATH=/usr/bin:/usr/ucb:/etc/:.
Change to **PATH=.:/usr/bin:/usr/ucb:/etc**

Save the file and close the editor.

12. It is standard practice as an administrator to periodically review the contents of the /tmp directory.

    Logout of CDE and log back in as root.

    Type: **cd /tmp**

    Type: **ls -al**

    What were the results of the running the script? Look carefully at the files that exist in the /tmp directory. Were any new files recently created ? If so, which ones?

    ____

    ____

    What would be the result if labuser1 were to login and execute the command /tmp/.secret_ksh?

    ____

    ____

13. Let's try to fix the vulnerability.

    Type: **dtpad /.profile**

    Modify the file so that the path statement does not have the ".". Save the file and close the editor.

14. Logout of CDE and log back in as root.

    Type: **cd /tmp**

    Type: **ls -a**

    Which ls command did you execute (the one in /tmp or the standard system call)? Does the .secret_ksh file exist ? What security vulnerability does the .secret_ksh file present?

____

____

15. As you have seen in previous exercises, the su command (switch user), allows root to change into any user account without a password. It also allows any user to become another user if they know that user's password. This includes becoming root. Unfortunately some versions of UNIX will allow ANY user to execute the su command. From a security point on view, this is NOT a good idea. Some versions of UNIX have a group called "wheel" which restricts the execution of su to only its members. It is totally unacceptable to allow unrestricted access to su.

    Let's restrict access to the 'su' command.

    Type: **ls –l /usr/bin/su**

    Who is the current owner? What is the current group? What are the file permissions? Who can execute this file? Who does it execute as (effective user id)? Is this a security vulnerability?

    ____

    ____

    ____

16. Type: **su labuser1** (prompt changes to $)

    Type: **/usr/bin/su**

    When requested, enter the root account password.

    Was the switching user to root successful? ______________________________

    Type: **exit** (to quit the su session.)

    Type: **exit** (to quit the su labuser1 session)

17. Change the permissions on the su command such that only the owner and group can execute the file

    Type: **ls –l /usr/bin/su**

    Type: **chmod 4550 /usr/bin/su**

    Type: **su labuser1**

    Type: **/usr/bin/su**

    Was switching user to root successful? Why or why not?

____

____

Type: **exit**

18. Create a special group that will be utilized to identified users that can use the su command.

    Type: **admintool &**

    Create a new group called 'susers.' Click on 'Browse,' select 'Groups', then click on 'Edit,' then 'Add.' Accept the default group ID, make labuser1 a member of this group. Exit Admintool.

19. Type: **chgrp susers /usr/bin/su**

20. Type: **ls –l /usr/bin/su**

    Did your changes take effect? (Is the group name different?)

    ____

21. Test the new group.

    Type: **su labuser2**

    Type: **/usr/bin/su**

    Can you execute the 'su' command? Why?

    ____

    ____

    Type: **exit**

22. Type: **su labuser1**

    Type: **/usr/bin/su**

    When prompted, enter the root user password.

    Can you execute the 'su' command? Why?

    ____

    Type: **exit**

    Type: **exit**

End of PE

PE 17 - Log Files

This PE will familiarize the student with some of the more important UNIX log files and how to read or manipulate them.

1. One of the logs in UNIX tells you who is actually logged in to the system and on what terminal. This log is a binary log called the utmpx file (utmp in older systems). It can be read by the "who" or "w" command. Other commands can access it, but knowing these about two is fine for now. Type: **who**

   Who was listed as being actively logged on the system? ___________________

2. Another of the UNIX logs tells us who is historically logging on the system. This log is referred to as the "lastlog" but normally is known as the wtmpx file (wtmp in older systems). The wtmpx file is a binary file that requires a special tool to read it. The command to use is the "last" command. Type: **last**

   Did you see a long list of logins? ____

   Were any of the logins from IP addresses other than your own? ___________

   Were any of the logins for accounts you haven't used? ____________________

   If you answered yes to b or c, you may have experienced some unauthorized visits during the class.

3. You can target individual users to verify their login habits or history.

   Type: **last labuser1**

   Do you see the login history for labuser1? ________________________________

4. Since the wtmpx file is binary and does not clean itself up, you have to take special steps to keep its size in check.

   To archive the file, type: **last > /lastlog1**

   Type: **cat lastlog1**

   Do you see the contents of the wtmpx file in ascii format in the lastlog1 file? _______

5. To zero out the file, pass a null value to it.

   Type: **cat /dev/null > /etc/wtmpx**

   Test your efforts. Type: **last**

   Is the log empty? __________________________

   Do you see a new log start date? ____

6. Process accounting is when the system keeps track of all commands that were run and by whom they were run. It is not a DA requirement to do process accounting as it takes a toll on system resources. It does, however, give you a complete picture of what is going on with respect to commands. Process accounting is started by running the “accton” command and then specifying the file in which to store the log data. Start up process accounting.

   Type: **/usr/lib/acct/accton /var/adm/pacct**

   Process accounting is now started. To read it you need to use special commands. The "acctcom" and "lastcomm" commands will do this for you.

   Type: **clear**

   Type: **ls**

   Type: **acctcom**

   Type: **lastcomm**

   When you executed acctcom and lastcomm, did you see the previous commands

   you ran? ________________________.

   Did you see that both commands read the /var/adm/pacct file but deliver a slightly

   different output? _______________________

7. Set up some of the reporting features within process accounting.

   Type: **/usr/lib/acct/runacct**

   This will set up process report info. You may see mention of it failing or setting up of holidays. Ignore these for classroom purposes.

   Type: **acctcom**

   Did you see the process accounting structured format? ___________________

   Type: **/usr/lib/acct/prdaily | more**

   This will print out a daily process report that can be viewed a page at a time. You will also notice that one page gives a record of the last time users logged in which is good for auditing account activities.

8. The general purpose logging facility known as "syslog" keeps track of errors and messages from numerous sources. Error messages are prioritized and linked to their creating facility. Syslog allows you to configure where errors are recorded via the syslog.conf file.

   Type: **cat /etc/syslog.conf**

Notice where different information is directed.

Type: **cat /var/adm/messages**

Type: **cat /var/log/syslog**

Did you see some of the various errors or messages that have been generated during the class? ____________ Normally we only what to view the most current info.

Type: **tail -10 /var/log/syslog**

This views the last 10 entries in the /var/log/syslog file

Type: **dmesg | tail -10**

This will read the last 10 entries in the /var/adm/messages file

9. It is important to track super-user activity. The "sulog" tracks this information.

   Type: **more /var/adm/sulog**

   Do you see the various times you became a different user during previous PE's?

   __________ Check the sulog configuration: Type: **cat /etc/default/su**

10. Set up logging of network services. Type: **dtpad /etc/init.d/inetsvc**

    Go to the bottom of the file where the line containing "inetd -s " is. Edit it to look like "**inetd -s -t** " (without quotation marks). Solaris 8 may also have a & symbol.

    Save the file and close the editor. Check to see if /etc/rc2.d/S72inetsvc is also changed. S72inetsvc is a link of inetsvc. Type: **cat /etc/rc2.d/S72inetsvc**

    Do both files look identical? _____________________ A change to one is a change to both. /etc/rc2.d/S72inetsvc is a link of /etc/init.d/inetsvc.

    Now edit the default inetd file. Type: **dtpad /etc/default/inetd**

    Un-comment (remove the # symbol) the line for enable_connection_logging=NO and change enable_connection_logging=NO to enable_connection_logging=YES. Save the file and exit the dtpad program.

    Next find the ID for the inetd process. Type: **ps –ef | grep inetd**

    Kill and restart the process, type: **kill –HUP** (**PID**)

    Test the logging features, type: **telnet localhost 21**

    Solaris 9: also type: **USER labuser1** (press **ENTER**) **PASS student1** (press **ENTER**)

    Type: **quit**

telnet localhost 79

Press **ENTER** twice (2 times)

Type: **dmesg | tail -10**

Do the logs indicate some ftp or finger activity? ________________________

11. A big problem with logs are that they grow big and need to be kept in manageable sizes. Take a look at the current log sizes. (Displayed in kilobytes).

    Type: **du –s –k /var/log/* | sort -rn**

    Type: **du –s –k /var/adm/* | sort -rn**

    Were you able to see some familiar log names and the size of the files? __________

12. Check the overall health of your file systems.

    Type: **df –k | awk '{ print $6 "\t" $4 }**'

    Do you see the names of the directories and how many kilobytes of free space they

    have? _______________________________

13. From time to time you will need to store files in an encrypted state. Encrypting sensitive data can be very useful.

    Type: **dtpad testfile**

    Enter the text: **I am a test file with very little ambition in life.**

    Save the file and close the editor

    Type: **cat testfile | crypt > testfile.cpt**

    Type: **password** when prompted for a key

    Type: **cat testfile.cpt**

    Is the file encrypted? ____________________

    Un-encrypt it. Type: **cat testfile.cpt | crypt > testfile.new**

    Type: **password** when prompted for the key

    Type: **cat testfile.new**

    Is the file unencrypted? _________________

End of PE

PE 18 - Specialized Logs

The purpose of this PE is to familiarize the student with setting up specialized UNIX log files and observing the output.

The /var/adm/loginlog file will collect failed login attempts. Bad login attempts will not be logged unless:

- the /var/adm/loginlog file exists
- is owned by root
- the group is sys
- has read and write permissions only for root

1. Create a log to collect failed login attempts.

   Type: **touch /var/adm/loginlog**

2. Change the group to sys.

   Type: **chgrp sys /var/adm/loginlog**

3. Change the permissions so that only root has read and write permissions.

   Type: **chmod 600 /var/adm/loginlog**

4. Verify your changes.

   Type: **ls -l /var/adm/loginlog**

5. Right click the background, left click Hosts and select Terminal Console. Minimize the Console Window -- do not close the window.

6. In the Terminal Window

   Type: **rlogin localhost –l labuser1**

   When a password is requested, enter a password of **duh**

7. Repeat step 6 four (4) more times.

8. Open the Console window by double clicking on the Console icon.

   What message is displayed ?

   ____

   ____

   Close the Console window.

9. Check to see if the bad login attempts have been captured by the /var/adm/loginlog file.

Type: **cat /var/adm/loginlog**

You should see that the bad login attempts have been logged. Have they?

____

10. The log file will tell you the account that had the bad login attempts, the port used, and the date and time of the attempts. Which port was used during the attempted logins?

____

____

11. Type: **grep FAILURES /var/adm/messages | more**

What information is displayed?

____

____

End of PE

Acronyms

ACE	Access Control Entries
A-CERT	Army Computer Emergency Response Team
ACK	Acknowledgement Character
ACL	Access Control List
ADO	ActiveX Data Object
AES	Advanced Encryption Standard
AH	Authentication Header
AIAP	Army Information Assurance Program
AISSP	Army Information Systems Security Program
AMC	Commanding General, US Army Material Command
ANSI	American National Standards Institute
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ASA(RDA)	Assistant Secretary of the Army for Research, Development and Acquisition
ASCII	American Standard Code for Information Interchange
ASET	Automated Security Enhancement Tool
ASP	Active Server Pages
ATM	Asynchronous Transfer Mode
BDC	Backup Domain Controller
BSD	Berkley Software Distribution
BSM	Basic Security Module
C^2 Protect	Command and Control Protect
CA	Certification Authority
CDAP	Computer Defense Assistance Program
CERT	Computer Emergency Response Team
CFAA	Computer Fraud and Abuse Act
CGI	Common Gateway Interface
CHAP	Challenge Handshake Authentication Protocol
CIDR	Classless InterDomain Routing
CIFS	Common Internet File System
CMOS	Complimentary Metal-Oxide Semiconductor
COM	Component Object Model
COMPUSEC	Computer Security
COMSEC	Communications Security
CRL	Certificate Revocation List
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CSIRT	Computer Security Incident Response Team
DAA	Designated Approving Authority
DAC	Discretionary Access Control
DACL	Discretionary Access Control List

DCD	Data Carrier Detect
DCE	Data Communication Equipment
DCID	Director of Central Intelligence Directive
DCSINT	Deputy Chief of Staff for Intelligence
DCSLOG	Deputy Chief of Staff for Logistics
DCSOPS	Deputy Chief of Staff for Operations and Plans
DDOS	Distributed Denial Of Service
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DH	Diffie-Helman
DHCP	Dynamic Host Configuration Protocol
DISC4	Director of Information Systems for Command, Control Communications & Computers
DITSCAP	DOD Information Technology Security Certification and Accreditation Process
DMS	Defense Messaging System
DNS	Domain Name Service
DODIIS	Department of Defense Intelligence Information Systems
DoS	Denial of Service
DRA	Data Recovery Agent
DRAM	Dynamic RAM
EAP	Extensible Authentication Protocol
EFS	Encrypting File System
ESP	Encapsulated Security Payload
EUID	Effective User Identification
FAT	File Allocation Table
FEK	File Encryption Key
FSCK	File System CHeck
FTP	File Transfer Protocol
GFE	Government Furnished Equipment
GNU	GNU Not UNIX
GPO	Group Policy Object
GREP	Generalized Regular Expression Pattern matcher
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HSFS	High Sierra File System
HTTP	Hypertext Transfer Protocol
I&A	Identification & Authentication
IA	Information Assurance
IAM	Information Assurance Manager
IATO	Interim Authority to Operate
IAO	Information Assurance Officer

IANM	Information Assurance Network Manager
IASO	Information Assurance Security Officer
IASSM	Information Assurance Systems Security Manager
IASSPM	Information Assurance Systems Security Program Manager
ICANN	Internet Corporation for Assigned Names and Numbers
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Algorithm
IDS	Intrusion Detection System
IGMP	Internet Group Management Protocol
IIS	Internet Information Server
IKE	Internet Key Exchange
IMAP	Internet Message Access Protocol
ISAKMP	Internet Security Association and Key Management Protocol
INSCOM	Commanding General, US Army Intelligence & Security Command
ISSA	Information Systems Security Agreement
ISSM	Information Systems Security Manager
ISSO	Information Systems Security Officer
KDC	Key Distribution Center
L2TP	Layer Two Tunneling Protocol
LBA	Logical Block Addressing
LDAP	Lightweight Directory Access Protocol
LGPO	Local Group Policy Object
LIWA	Land Information Warfare Activity
LSA	Local Security Authority
MAC	Mandatory Access Controls
MAC	Message Authentication Codes
MBR	Master Boot Record
MBSA	Microsoft Baseline Security Analyzer
MIME	Multipurpose Internet Mail Extension
NAT	Network Address Translation
NCSC	National Computer Security Center
NFS	Network File System
NIAP	National Information Assurance Partnership
NIS	Network Information Service
NIST	National Institute of Standards and Technology
NTFS	New Technology File System
NTLM	NT LAN Manager
NTISSC	National Telecommunications & Information Systems Security Committee
OPSEC	Operations Security
OU	Organizational Unit
OWF	One Way Function

PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PDC	Primary Domain Controller
PEAP	Protected Extensible Authentication Protocol
PEO	Program Executive Officers
PGP	Pretty Good Privacy
PID	Process ID
PKI	Public Key Infrastructure
PM	Program Manager
POP	Post Office Protocol
POSIX	Portable Operating System Interface for UNIX
PPP	Point to Point Protocol
PPTP	Point to Point Tunneling Protocol
PTR	Pointer Resource Records
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RAID	Redundant Array of Independent [Inexpensive] Disks
RBAC	Role Based Access Control
RDTE	Research, Development, Test, and Evaluation
RID	Relative Identifier
RPC	Remote Procedure Call
RRAS	Routing and Remote Access Service
SACL	System Access Control List
SAM	Security Accounts Manager
SANS	System Administrator and Network Security
SATAN	System Administrators Tool to Analyze Networks
SATs	System Access Tokens
SCI	Sensitive Compartmented Information
SFUG	Security Features User's Guide
SGID	Set Group ID
SHA	Secure Hash Algorithm
SID	Security Identifier
SIOP-ESI	Single Integrated Operations Plan - Extremely Sensitive Information
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNA	System Network Architecture
SPAP	Shiva Password Authentication Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
ST&E	Security Test & Evaluation
SUID	Set User ID

TCB	Trusted Computing Base (Orange Book)
TCP	Transmission Control Protocol
TCSEC	Trusted Computer System Evaluation Criteria (Orange Book)
TED	Truck Encryption Devices
Tempest	(not an acronym) Unclassified US government code word for compromising emanations; now called Emissions Security or EMSEC
TFM	Trusted Facility Manual
TFTP	Trivial File Transfer Protocol
TGT	Ticket Granting Ticket
TRADOC	Commanding General, US Army Training and Doctrine Command
UDP	User Datagram Protocol
UFS	UNIX File System
UID	User Identification
UNC	Universal Naming Convention
UPN	Universal Principle Name
UPS	Uninterruptable Power System
URL	Uniform Resource Locator
UUCP	UNIX to UNIX Copy
VPN	Virtual Private Network
WAP	Wireless Application (access) Protocol
WEP	Wireless Equivalent Privacy
WINS	Windows Internet Name Service
WML	Wireless Markup Language
WWW	World Wide Web