Unix Security

COMP 2400, Fall 2008

Chris GauthierDickey

UNIVERSITY OF DENVER

Understanding Unix Security

- Unix has a reputation for being secure but also for having bugs in programs that allow one to exploit the system
- What is the source of these bugs? How can they be prevented?

Step 1

- The first step to securing your system:
  - Don't turn it on!

- Perhaps that's a bit extreme
  - Lock it in a room that you only have a key to and don't connect it to the network

Considerations

- Do you really think the Pentagon is going to connect their systems to the Internet?
  - Only insecure ones
- Lockheed-Martin has a set of machines that cannot be connected to the outside world when you work on them
  - It has some that can be connected, but they are for projects that don't require national security

Considerations...

- In addition, at LMCO, you cannot bring in devices that let you transfer data--no cell phones, no PDAs, no USB sticks
- Really, Step 1 for security makes sense when it's that important
  - We rely on the fact that physical security is somewhat easier to manage than network security

Step 2

- Keep your system up-to-date
  - Run as little 'new' software as possible
    - Bugs can be bad, but known bugs are better than unknown bugs
- Consider, if you're going the Linux route, installing a more stable distribution--it'll be older, but most/all of the major security bugs have been discovered and addressed

Check out US-CERT

- US-CERT publishes security alerts:
  - http://www.us-cert.gov

- Make sure you keep your own system updated
- In UNIX, we can create a cron job to do the task (and yes, there are other schedulers, but cron is on all the Unix systems)
  - Apple now uses launchd for this purpose

cron

- 'cron' is a program that runs at specified time on the Unix systems
- cron actually checks the configuration files once a minute to see if it needs to run anything
- We can edit a user crontab by the command: crontab -e

```
# m h dom mon dow        command
42 6 * * *               dostuff.sh
@daily                   /usr/bin/apt-get update
@daily                   /usr/bin/apt-get install
0,30 * * * *             dohalfhourstuff.sh
0 * * * 1-5              doweekdaystuff.sh
```

cron continued

- Cron will automatically email output of the command after it runs
  - You can pipe output to a log file or to /dev/null
  - You can set the environment variable MAILTO so that it mails to a specific address
- It may seem simple, but your vast knowledge of bash programming will let you do pretty much whatever you want!

Step 3: File System Security

- Recall that when we list a file in the long format we can see its permissions
  - drwxrwxrwx
- We may also see a couple of other things like:
  - d r w s r w s r w x
    - The 's' means SUID and SGID for user and group

SUID and SGID

- SUID changes the user ID of the executing program to be that listed as the owner of the program
- SGID changes the group ID just like SUID does for executable files
- SGID on a directory causes all new files to be created with that group ID

More SUID and SGID

- SUID bits show up as the 's'
- SUID programs are the source of many, many security problems
  - Only a few programs really need the SUID bit
  - The Linux kernel will ignore SUID on a shell script
    - You can't make shell scripts secure, so the kernel will ignore it
- Combine SUID bits with buffer overflow and you get a hacked system

The root

- root *is* the super-user
- root can access, delete, modify, and execute any file on the system
- Allowing anyone you don't completely trust to have root access is a Bad Thing™
- Understand now why SUID and SGID is bad?

(Almost) Never log in as root

- Instead, use something like sudo or fakeroot, which temporarily give you root permissions, but log all root commands
- Create a user account that you use regularly with minimal permissions
  - You can add yourself to the sudo list for frequent commands you need root access for

A small divergence

- Ever wonder why Windows has had so many problems over the years with security?
  - 1st, it's the most widely used OS, so we'll of course see more problems with it
  - 2nd, until Vista, almost all programs were installed as a root equivalent
    - In essence, they were all SUID and SGID with root as owner
  - Even in Vista, older programs ask you to install them SGID by setting them to run as Administrator!

Back to SUID and SGID

- So, how do you set it?
  - chmod u+s or chmod g+s
  - In octal notation, recall that user, group, others were contained in 3 octal numbers
- There's actually 4 octals:
  - 4000 = SUID
  - 2000 = SGID
  - 1000 = sticky bit, makes files in a directory deleteable only by the owner of the file: used by the /tmp directory

Changing default permissions

- By default, files you create are affected by a mask which removes permissions
  - We call it the 'umask' by its command name (run it!)
  - Each bit in the umask removes that permission
    - 0022, the default, removes write and execute bits so that created files have at most 755 permissions (or 644, if it's a non-executable file being created)
- Most restrictive?
  - umask 077, and yes, root should have this

UNIVERSITY OF DENVER

Finding those SUID/SGID files

- Some Unixes scan for you and save the results (look in /var/log/setuid)
- We can run:
  - find / -type f \(-perm -04000 -o -perm -02000 \)
  - If we keep track of them and notify ourselves when they change, we may be able to detect someone creating a back door

Other suspicious files

- Any world-writable file can be a security hole (especially system files)
  - find / -perm -2 ! -type l -ls
    - ! -type l ignores /dev and symbolic links
- Unowned files can indicate a problem or a hacked file
  - find / \(-nouser -o -nogroup \) -print
- .rhosts file should *never* be allowed
  - find /home -name .rhosts -print

UNIVERSITY OF DENVER

Setting limits

- We can set limits on other users if we're root, but we can also set them on ourselves
- Use 'ulimit' to find out your limits
  - Try it with ulimit -a
- ulimits can be hard or soft
  - hard limits can only be changed by root
  - soft limits can be changed up to the max of the hard limits by the users

General Scanning

- You can use integrity scanners such as Tripwire to store the integrity values of your files on a read-only medium (dvd, cd)

- You can also use security scanners such as Nessus which look for known exploits
  - These port scan and try to log into your system
    - Don't use them on systems you don't own!

User Security

- First, always use ssh and scp to log into or copy files from system to system
  - This encrypts your user name and password
  - telnet, rlogin, ftp all send your password hash in plain-text, making it easy to crack
- Disable services you don't need in inetd, such as ftp, telnet, etc--less is always better, you can turn something on if you discover you need it!

Step 4

- Keep backups!
  - You only really need user files and configuration files
  - Restore if you discover you've been hacked, but reinstall fresh executables
  - Backing up and restoring executables could put the hacker's files right back on your machine

User Security

- Don't use NIS, use NIS+ which is more secure
- Use a firewall: ipchains for > Linux 2.2
- Recent hacking contests show that users who just try to hack a machine over the network are usually unable to if the firewall is enabled
  - However, the next fastest attack was exploiting web browser bugs by getting someone to visit a web-site