INFO-6003

O/S & Application Security

Week 12

Agenda

- Exam in Next Week
- More Access and File System Control
- More detail on PAM
- Patching
- Logging
- init, Upstart init, systemd init
- Managing Services
- chroot

Next Week

- Final Exam
- FOL Respondus Browser
- 90 minutes long
- 3 long answer questions, also MC and TF

More Access Control

Access Control

- Control and Manage the following:
  - User accounts and system administrator functions
  - Access to files, utilities and services
- SUID/SGID and sudo command allow users to assume root permissions for specific commands
  - Only when needed
- Visudo can be used to manage who can use sudo

Visudo

- visudo is the command used to edit the sudoers file
- Contains a list of users who can perform operations with the sudo command
- You can limit the commands a user can use with sudo (**sudo tcpdump** for example)
- We will be doing this in the lab today

Changing Passwords

- The passwd command can be used to set password options on accounts
  - Length of password
  - Lock or Unlock account
  - Number of days a password can be used
  - Days before password can be changed
  - Expiration date
    - File remain intact, but account is locked
- Similar options to what we saw in Windows

Passwd Options

-a, --all	report password status on all accounts
-d, --delete	delete the password for the named account
-e, --expire	force expire the password for the named account
-i, --inactive INACTIVE	set password inactive after expiration to INACTIVE
-l, --lock	lock the password of the named account
-n, --mindays MIN_DAYS	set minimum number of days before password change to MIN_DAYS
-S, --status	report password status on the named account
-u, --unlock	unlock the password of the named account
-w, --warndays WARN_DAYS	set expiration warning days to WARN_DAYS
-x, --maxdays MAX_DAYS	set maximum number of days before password change to MAX_DAYS

File Security & Integrity

- File Integrity Checkers
  - Monitor system files
  - Creates snapshot of files: a hashed signature (message digest) for each file
  - After an attack, compare post-hack signature with snapshot
    - This allows systems administrator to determine which files were changed
- Tripwire is file integrity checker that can be used for some Linux systems
  - Red Hat and SUSE

Tripwire & AIDE

- Tripwire
  - Tripwire detects changes to file system objects
  - Tripwire creates a cryptographic hash of a file and stores the hash on each file scanned in a database
  - Hashes created on next scan are compared to stored hash
  - Changes could indicate a hacker altering files
- AIDE
  - Advanced Intrusion Detection Environment
  - Developed as a free replacement to Tripwire

Samhain

- Samhain
  - Host intrusion detection & integrity checker
  - Host & network scan options
  - uses cryptographic checksums of files to detect modifications,
  - can find rogue SUID executable files anywhere on disk
  - Supports logging to a central server

http://www.la-samhna.de/samhain/

More on Pluggable Authentication Modules

PAM

- PAM provides a library containing functions for proper authentication procedures
- Allows for a separate module to provide authentication so it does not have to be in the program API

PAM

- PAM configuration files have 3 entries
  - The first entry indicates one of four categories which identify different types of modules for controlling access to a particular service
  - The second field in each entry is called the control flag and determines the action taken when the module succeeds or fails
  - The third field contains the values

```
password   [success=2 default=ignore]   pam_unix.so obscure sha512
```

PAM

- Authentication
  - Provides the actual authentication (perhaps asking for and checking a password) and sets credentials, such as group membership or Kerberos tickets
- Account
  - Checks the account has not expired, the user is allowed to log in at this time of day, and so on
- Password
  - Module is used to set passwords
- Session
  - Used after a user has been authenticated
  - Performs additional tasks which are needed to allow access such as mounting the user's home directory

PAM Control Options

- Required
  - The module must succeed
  - Regardless of whether the module fails or succeeds, processing will continue with the next line (other modules of the same module type will be executed)
  - At the end of all of the processing, a failure will be recorded
- Requisite
  - The module must succeed for the module type to succeed
  - If it fails, processing stops immediately
  - If it succeeds, processing continues with the next line

PAM Control Options

- Sufficient
  - If the module succeeds, then the module type succeeds and processing stops immediately
  - If it fails, processing continues with the next line
- Optional
  - The module is executed, but the failure or success of the module is ignored
- Include
  - In place of a module name, another configuration file is given
  - All of the lines of the same type from that configuration file are treated as if they were present in this configuration file

Logging

Logging

- Logging is used to record system events
  - This is sometimes called auditing
- Linux syslogd, rsyslogd and kernel log daemon, klogd, write events to log files
- The /etc/syslog.conf or /etc/rsyslog.conf file specifies were log files are located
  - Exact location depends on the version of Linux
  - In addition to log location the .conf files also specify additional parameters such as log file format

Logging

- /var/log/secure
  - Successful & failed logins
- /var/log/messages
  - General error messages from kernel or other services
- /var/log/boot.log
  - Information logged during system boot
- Specific services that have a lot of activity can have their own log file
  - /var/log/httpd
  - Web service messages

Logging

- /var/log or /var/adm hold the following files that are used to investigate hacking or suspicious activity
  - utmp
    - Current status of system: boot time, logins, logouts, system events
  - wtmp
    - Historical record of utmp
  - btmp
    - Records failed login attempts
  - lastlog
    - Last time and location of user's last login to system
    - Console, ssh or telnet

Patching

Update Manager

- Most Linux distributions have an update tool to download patches for OS or software packages
  - Some have more than one package manager
- Red Hat, Fedora, CentOS
  - **up2date** (Red Hat Update Agent) & **YUM** (Yellowdog Updater, Modified), **RPM** (Red Hat Package Mgr.)
- SUSE Linux
  - **YaST** (Yet another Setup Tool)
- Debian
  - **APT** (Advanced Packaging Tool)

Apt Command Examples

- apt-get update
  - Updates packages listings from the repo, should be run at least once a week
- apt-get upgrade
  - Upgrades all currently installed packages with those updates available from the repo. should be run once a week
- You may need to turn off your AV to get these commands to work
  - Especially when we start downloading hacking tools

Apt Command Examples

- apt-cache search <pattern>
  - Searches packages and descriptions for <pattern>
- apt-get install <package>
  - Downloads <package> and all of its dependencies, and installs or upgrades them
- apt-cache search extundelete
  - Will search the repository for extundelete
- apt-cache install extundelete
  - Will install extundelete on your system

Hardening or Hardened

Hardening or Hardened

- When is comes to securing your Linux distribution you have a couple choices
- Hardening
  - This is the process of locking down an existing distribution to make it more secure
- Hardened
  - Choosing a distribution that has been specifically developed with security in mind

Hardening or Hardened

- EnGarde Secure Linux (hardened distro)
  - http://www.engardelinux.org
- Hardened Linux (hardened distro)
  - http://hardenedlinux.sourceforge.net
- Bastille (hardening program)
  - http://bastille-linux.sourceforge.net
- SELinux (security enhancements)
  - http://selinuxproject.org
- AppArmor (security enhancements)
  - http://wiki.apparmor.net

Bastille

- The Bastille hardening program
  - Designed to lock a system down based on best practices
  - Interactively configures the system
  - Can be used to analyze the systems state
  - Supports a wide variety of platforms
    - Red Hat / Fedora Core / SUSE / Mandrake
    - Debian
    - Gentoo
    - HP-UX
    - Mac OS X

SELinux

- SELinux is not a hardened distribution
  - It is a set of Kernel modifications and tools that can be added to Linux distributions
- Primarily developed by the NSA
- Makes heavy use of MACLs and Linux Security Modules (LSMs) in the Linux Kernel
  - The MACLs are used in place of DACLs
- Great for hardening the system, but does make management more difficult

SELinux

- Makes use of subjects, objects, labels and policies to lock the system down
- Subjects
  - Users, applications, process, etc.
- Objects
  - Files, sockets
- Labels
  - Metadata applied to objects
- Policies
  - Access permissions for subjects and objects

Init Daemons

init Daemon

- The first process to start, aside from the kernel, is the init daemon
  - The Linux Kernel has a process ID of 0
  - The init daemon has a process ID of 1
- The init daemon starts all the other processes and services as the system boots
- The traditional init isn’t used in modern distros
  - System V init
- Modern versions of init
  - Upstart init
  - systemd init

init Daemon

- The newer init Daemons were developed to deal with dynamic environments
  - USB keys
  - Other hot-plug devices
- The newer init Daemons have some backwards compatibility built in, but the move is towards getting rid of this

Run Levels

- Allow users to start the OS in different operating modes
  - Depending on the run level different functionality will be available
    - Single User
    - Multi-User
    - Network Services start, or Stopped
- Windows also has this functionality, but implemented differently
  - Safe Mode, Safe Mode with Networking, Command Prompt, etc.

Run Levels

- Run Level 0
  - Used to Halt or shutdown the system
- Run Level 1
  - Single user mode minimal configuration
- Run Level 2
  - Multi user mode without networking
- Run Level 3
  - Multi user mode with networking
- Run Level 5
  - X Window mode (Full GUI)
- Run Level 6
  - Used to reboot the system D

Variations in Run Levels

- Run levels vary by distribution
  - Especially run levels 2 through 4
- You can use the init command to change run levels (good for troubleshooting)
  - init 0: Will shutdown the system
  - init 1: Will start the system in single user mode
  - init 6: Will reboot the system
- You can see your current run level a couple ways
  - runlevel
  - who -r

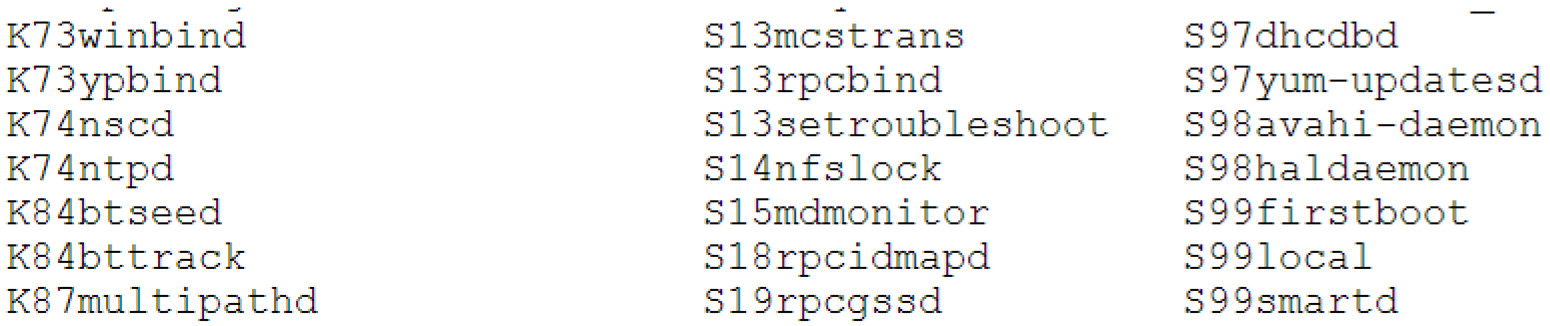
init

init

- Uses the /etc/inittab file to determine the default run level
  - This is used if you don’t actually specify a run level
  - id:5:initdefault:
    - Would start an X windows sessions (GUI)
- Based on the run level, certain scripts are run
  - /etc/rc.d/rc#.d
  - /etc/rc.d/rc5.d in the example above
  - The location of the files varies by distro

init rc#.d

- The rc#.d directories contain scripts that start or kill processes
  - S for starting process
  - K for killing process
- The scripts are also numbered which allows them to be started or stopped in a particular order



```
K73winbind          S13mcstrans           S97dhcdbd
K73ypbind           S13rpcbind            S97yum-updatesd
K74nscd             S13setroubleshoot     S98avahi-daemon
K74ntpd             S14nfslock            S98haldaemon
K84btseed           S15mdmonitor          S99firstboot
K84bttrack          S18rpcidmapd          S99local
K87multipathd       S19rpcgssd            S99smartd
```

Dependencies

- The fact that the administrator can control the order the services start allows dependencies to be taken into account
  - If service A is a dependency for service B it will need to have a lower number, so that service A is up and running before service B

Upstart init

Upstart init

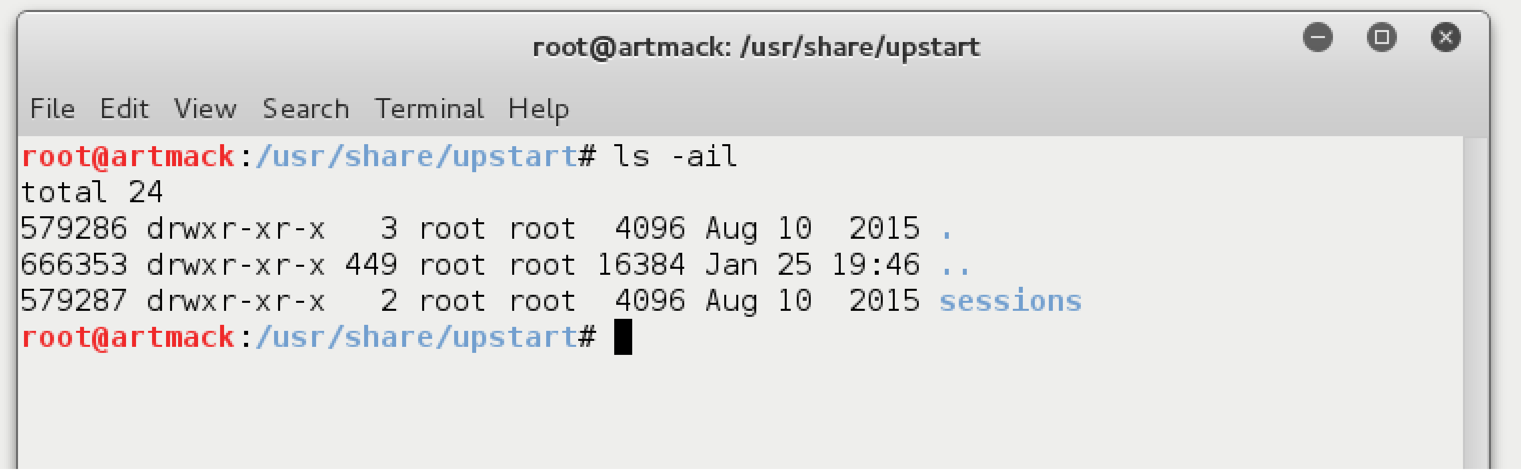
- Used in many newer distributions
  - Fedora 9 to 14
  - Ubuntu 6.10 and later
  - Google's Chrome OS
- The primary difference between Upstart init and init is the handling of starting and stopping services
  - Geared towards handling and ever changing environment
    - Hot pluggable devices

Upstart init

- Handles services through defined jobs
  - Jobs can be either a task of a service
- Tasks
  - Preforms a limited duty, and when finished returns to the waiting state
    - stop/waiting
- Service
  - Long running program that never self-terminates
  - Stays in the running state
    - start/running

Upstart init

- /usr/share/upstart
- Good way of finding out that you are on an Upstart-based system

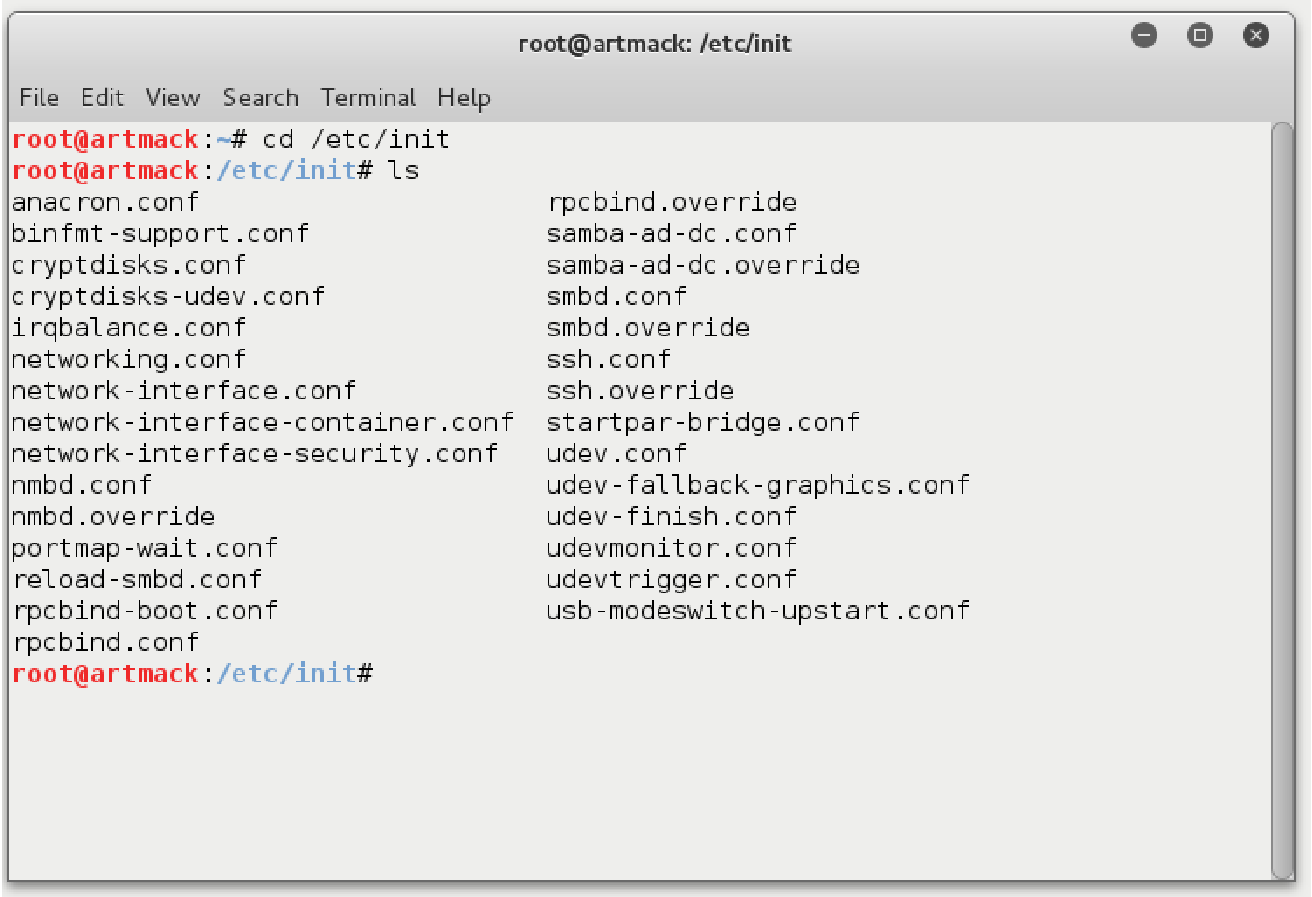


root@artmack: /usr/share/upstart
File Edit View Search Terminal Help

```
root@artmack:/usr/share/upstart# ls -ail
total 24
579286 drwxr-xr-x   3 root root  4096 Aug 10  2015 .
666353 drwxr-xr-x 449 root root 16384 Jan 25 19:46 ..
579287 drwxr-xr-x   2 root root  4096 Aug 10  2015 sessions
root@artmack:/usr/share/upstart# 
```

Upstart init

- The various jobs are defined in /etc/init



```
root@artmack:~# cd /etc/init
root@artmack:/etc/init# ls
anacron.conf                          rpcbind.override
binfmt-support.conf                   samba-ad-dc.conf
cryptdisks.conf                       samba-ad-dc.override
cryptdisks-udev.conf                  smbd.conf
irqbalance.conf                       smbd.override
networking.conf                       ssh.conf
network-interface.conf                ssh.override
network-interface-container.conf      startpar-bridge.conf
network-interface-security.conf       udev.conf
nmbd.conf                             udev-fallback-graphics.conf
nmbd.override                         udev-finish.conf
portmap-wait.conf                     udevmonitor.conf
reload-smbd.conf                      udevtrigger.conf
rpcbind-boot.conf                     usb-modeswitch-upstart.conf
rpcbind.conf
root@artmack:/etc/init#
```

Upstart init

- The job definition files determine what services are started, stopped, restarted, etc.
- When certain conditions are met certain actions are taken
  - Allows for a much more dynamic environment
  - Ability to respond to change

systemd init

sytemd init

- Alternative to init and Upstart init
- Runs on a variety of distributions
  - Arch Linux (since 2012)
  - Fedora 15 and later
  - Mandriva (since 2011)
  - openSUSE 12.1 and later
- Some key differences as compared to Upstart init
  - Starts fewer services
  - Starts services in a parallel manner
  - Supervises all processes

systemd init

- Instead of run levels systemd uses target units
- A unit is a group consisting of
  - name
  - type
  - configuration file
- A unit is focused on a particular service or action

systemd init

- There are eight unit types
  - automount
  - device
  - mount
  - path
  - service
  - snapshot
  - socket
  - target
- The service and target units primarily deal with services

systemd init

- A service unit is used to manage a daemon
  - rsyslog.service
  - sshd.service
- A target unit is a group of other units
  - sysinit.target
    - All the actions required for system initialization
  - syslog.target

Auditing Services

Auditing Services

- For init systems
  - chkconfig –list
    - Lets you know what services are on or off for particular run levels
  - service –status-all
    - Allows you to see if a service is running or not
- For Upstart init systems
  - initctl list
    - Let you see the service state
      - start/running
      - stop/waiting

Auditing Services

- For systemd init systems
  - – systemctl list-unit-files –type=*unittype*
- With all of the auditing commands you can use grep to filter your results
  - – | grep running
    - for running services
  - – | grep wait
    - for stopped services

Controlling Services

Controlling Services

- All the init daemons have built in mechanisms for manually controlling/polling the services
  - start
  - stop
  - restart (will stop and start the service)
  - reload (only reloads the configuration file)
  - status
- These can be helpful when troubleshooting
- Additionally, you can restart a service without rebooting the server

Controlling Services

- With init
  - service cups status
  - service cups start
  - service cups stop
  - service cups restart
- With Upstart init
  - initctl status cups
  - initctl start cups
  - initctl stop cups
  - initctl restart cups
  - initctl reload cups

Controlling Services

- With systemd init
  - systemctl status cups.service
  - systemctl start cups.service
  - systemctl stop cups.service
  - systemctl restart cups.service
  - systemctl reload cups.service
  - systemctl condrestart cups.service
- The condrestart is a conditional restart that will only restart the service if it is already running
  - inactive services stay inactive

chroot

chroot

- Pronounced “cha-root”
- This is also sometimes referred to as “chroot jail”
- Chroot will limit access to only that part of the file system defined by chroot
  - Normally implemented to restrict access for untrusted or anonymous users, untrusted applications etc.
  - The initial process and all its child processes will perceive the root directory to be that which is   defined when setting up the chroot environment

chroot

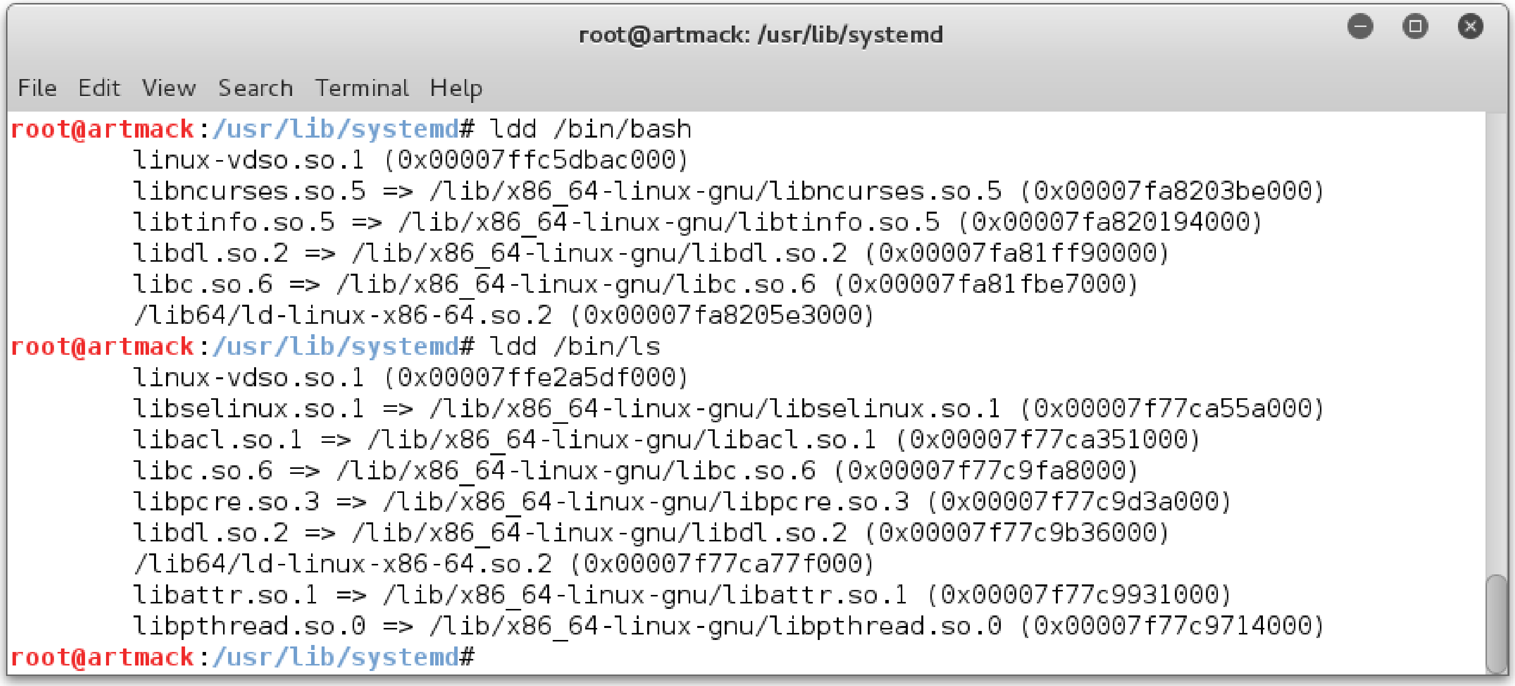
- The directory configured for chroot is treated as the root of the files system for all processes started in the chroot directory
  - Impossible to access any files or binaries outside of the chroot directory
    - As long as no privilege escalation exists
- Because an application running within a chroot jail can’t access any files outside, all its required files need to be within the chroot jail
  - passwd files
  - libraries
  - binaries, etc.

chroot

- Directory Structure
  - You match the expected directory structure within the chroot environment
- Bash and ls example
  - If you wanted to run a bash shell and use the ls command in a chroot environment you would need the following
    - /tmp/newroot/bin/bash
      - to hold bash and ls
    - /tmp/newroot/lib
      - to hold the libraries

chroot

- You can determine which libraries will be required to run your programs
  - Will vary by distribution



root@artmack: /usr/lib/systemd
File Edit View Search Terminal Help

```
root@artmack:/usr/lib/systemd# ldd /bin/bash
	linux-vdso.so.1 (0x00007ffc5dbac000)
	libncurses.so.5 => /lib/x86_64-linux-gnu/libncurses.so.5 (0x00007fa8203be000)
	libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5 (0x00007fa820194000)
	libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007fa81ff90000)
	libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fa81fbe7000)
	/lib64/ld-linux-x86-64.so.2 (0x00007fa8205e3000)
root@artmack:/usr/lib/systemd# ldd /bin/ls
	linux-vdso.so.1 (0x00007ffe2a5df000)
	libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007f77ca55a000)
	libacl.so.1 => /lib/x86_64-linux-gnu/libacl.so.1 (0x00007f77ca351000)
	libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f77c9fa8000)
	libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f77c9d3a000)
	libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f77c9b36000)
	/lib64/ld-linux-x86-64.so.2 (0x00007f77ca77f000)
	libattr.so.1 => /lib/x86_64-linux-gnu/libattr.so.1 (0x00007f77c9931000)
	libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f77c9714000)
root@artmack:/usr/lib/systemd#
```

chroot

- After copying all the bin, ls and library files to the chroot environment you need to enable to environment
- Most distributions contain a program called chroot that invokes the chroot() system call for you
  - The program takes two variable
    - chroot directory
    - command we want to run
- In our example

```
chroot   /tmp/newroot   /bin/bash
```

chroot

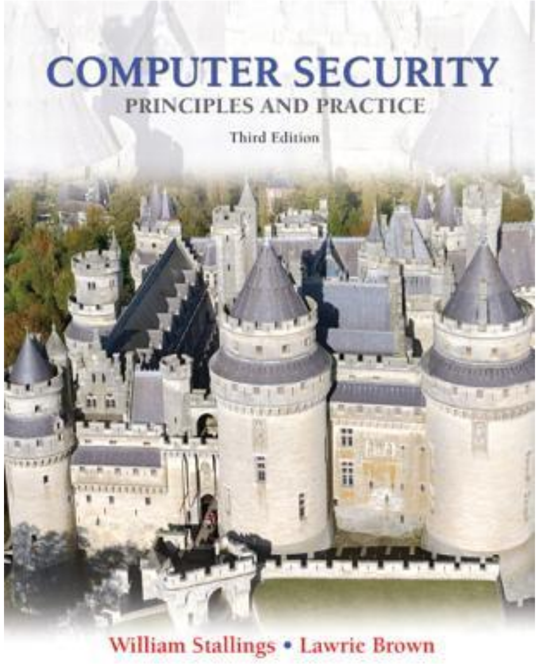
- It can be difficult to find all the libraries and other dependencies required
- There are tools available to find these dependencies
  - ldd finds the library dependencies
  - strace finds the system calls
  - lsof lists the open files and the processes that opened them

chroot Cautions

- Never add any files into chroot that have functions that can be used to escape
  - No compilers
  - No interpreter
  - No services that require root to run
- Daemons placed in chroot should not run with root permissions
  - They would be able to break out

Homework

- Read Online Chapter 25
- 25.1 – 25.9 – Linux Security



Lab 10 – initCtl & Sudo

Lab 10 Details

- Create LAN Segment for VMs
- Configure Network Interfaces
- Use Sudo for administration