COSC 3364 – Principles of Cybersecurity

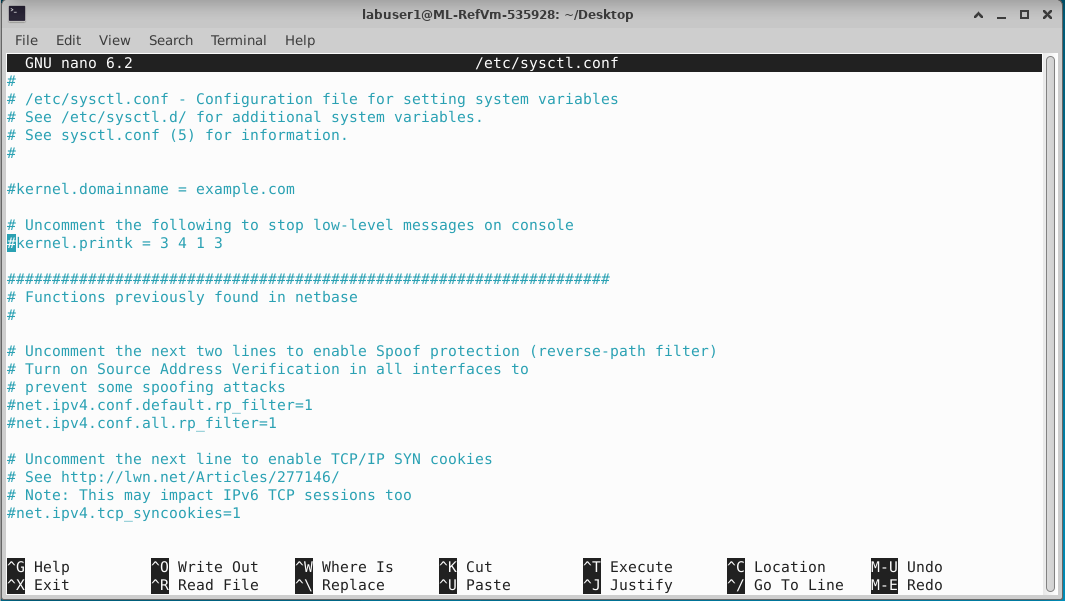
Lab 08

Provide screenshots where \* is indicated.

Network Security Policy

You can use kernel parameters to modify the behavior of the kernel by adjusting key features. They can be used to change how the kernel manages devices, optimize memory usage, and enhance the security of the system.

* Navigate to the system control configuration file: **/etc/sysctl.conf**
* Determine which kernel parameters are enabled\* (only a portion of the file is required for the screenshot)



None of the parameters are currently enabled because they are all commented out.

Possible kernel parameters:

1. **Ignoring ping requests** – the ping command is often used to determine if a remote host is accessible through the network. An adversary can use ping to probe for active systems trying to find systems that they can break into. Responding to ping requests can leave a system vulnerable to denial of service attacks. To ignore ping requests, use the following setting in the **/etc/sysctl.conf:**

net.ipv4.icmp\_echo\_ignore\_all = 1

1. **Ignoring Broadcast Requests** – Broadcast requests can be used for DoS and DDoS attacks. To ignore broadcast requests, use the following setting in the **/etc/sysctl.conf:**

net.ipv4.icmp\_echo\_ignore\_broadcasts = 1

1. **Enabling TCP SYN Protection** – A SYN flood attack is another DoS attack where SYN requests are used to make a system unresponsive. To Ignore SYS requests, use the following setting in the **/etc/sysctl.conf:**

net.ipv4.tcp\_syncookies = 1

1. **Disabling IP Source Routing** – a feature that enables the sender of a packet to specify the network route that should be taken. This feature bypasses routing tables and makes your system vulnerable to man-in-the-middle attacks. To disable this feature from a specific network device, use the following setting in the **/etc/sysctl.conf:**

net.ipv4.conf.eth0.accept\_source\_route = 0

TCP wrappers are used when server programs that have been compiled with the **libwrap** library call that library when a system tries to access the service. An easy way to determine whether a service uses the **libwrap** library is to use the **ldd** command:

ldd <program>| grep libwrap

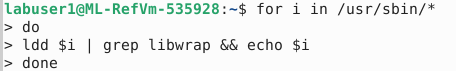
if the command returns **libwrap.so.0**, then the program uses TCP Wrappers.

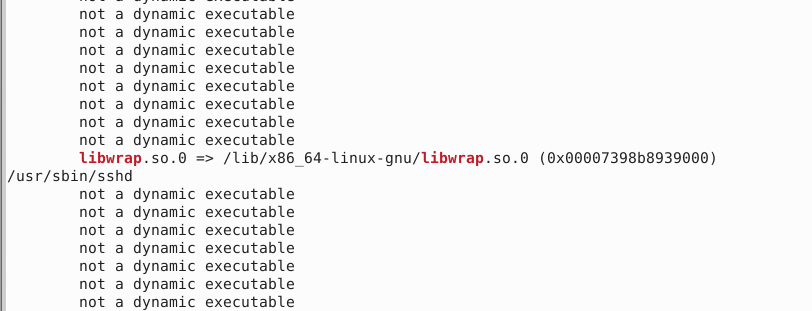
* Execute to determine which program(s) use TCP Wrappers\*:

for i in /usr/sbin/\*  
>do

>ldd $i | grep libwrap && echo $i

>done





The **libwrap** library uses configuration files to determine whether the SSH connection should be allowed, based on what machine is initiating the connection. The files used are the **/etc/hosts.allow** and **/etc/hosts.deny** files.

A diagram of a match rule

Description automatically generated

The syntax of the rules in the **/etc/hosts.allow** and **/etc/hosts.deny** files is

service\_list: client\_list [options]

The service is the name of the binary executable service program (for example, sshd or xinetd). The client list is what system(s) this rule should apply to.

The **client\_list** is also flexible. The following list details the different values you can provide:

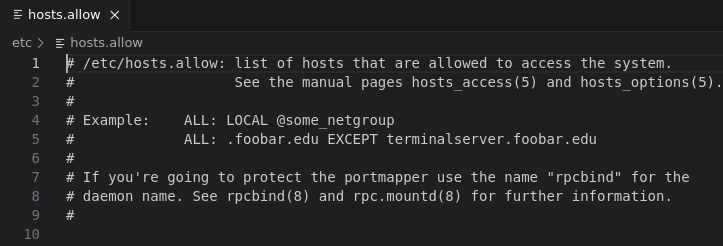
* + **IP address**: Example: 192.168.0.100.
  + **Network**: Example: 192.168.0.0/255.255.255.0 or 192.168.0.
  + **Entire domain**: Example: **.example.com**.
  + **ALL**: Matches every possible client.
  + **LOCAL**: Matches clients without a dot in their hostname. Example: test1.
  + **UNKNOWN**: Matches clients that can’t be resolved via the hostname resolver (DNS, local hosts file, and so on).
  + **KNOWN**: Matches clients that can be resolved via the hostname resolver (DNS, local hosts file, and so on).

Example: sshd: test.onecoursesource.com

Example: xinetd,sshd: test.onecoursesource.com

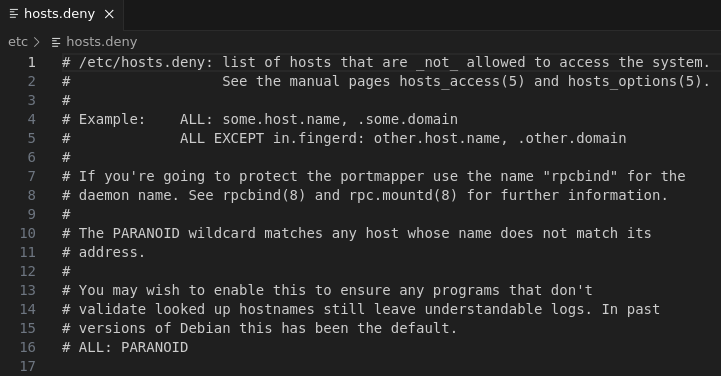
Example: ALL: test.onecoursesource.com

* Navigate to **/etc/hosts.allow** and determine what connections are allowed\*



No connections are currently allowed.

* Navigate to **/etc/hosts.deny** and determine what connections are denied\*



No connections are currently denied.

Process Control

The ps command is used to list processes that are running on the system. With no arguments, the command will list any child process of the current shell as well as the BASH shell itself.

Each line describes one process. By default, the **ps** command displays the following information:

* + **PID**: Process ID number; each process has a unique ID that can be used to control the process.
  + **TTY**: This is the terminal window that started the process. A terminal window is essentially a place where a user is able to issue commands from the command line.
  + **TIME**: CPU time; how much time the process has used to execute code on the CPU. Although this number can grow over time, it is typically a very small number (a handful of seconds or maybe a few minutes, but rarely more), unless something is wrong with the process.
  + **CMD**: The command.

To list all processes running on the system, add the **-e** option. The command **wc -l** can be used with the **ps** command to display the number of lines of data produced by the command to determine the number of processes running on a system. Each process is displayed on a separate line, so to be able to display a specific process you will want to use a **grep** command to filter the output.

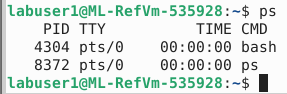
The **top** command displays process information that is updated on a regular basis (by default, every two seconds). The first half of the output of the top command contains overall information, whereas the second half displays a select list of processes (by default, the processes with the most CPU utilization).

The **free** command displays memory statistics.

To execute a process in the background, add an ampersand (**&**) character to the end of the command. Running a process in the background allows you to continue to work in the BASH shell and execute additional commands. Each BASH shell keeps track of the processes that are running from that BASH shell. These processes are referred to as jobs. To list the currently running jobs, execute the **jobs** command from the BASH shell.

The phrase “kill a process” is used to describe when you completely stop a process. Several methods are available: the **kill** command, the **pkill** command, the **killall** command and the **xkill** command. The **kill** command can be used to change the state of a process, including to stop (kill) a process. To stop a process, first determine its process ID or (%) job number and then provide that number as an argument to the **kill** command. You can kill a process by running the xkill command and then just clicking the process that you want to stop.

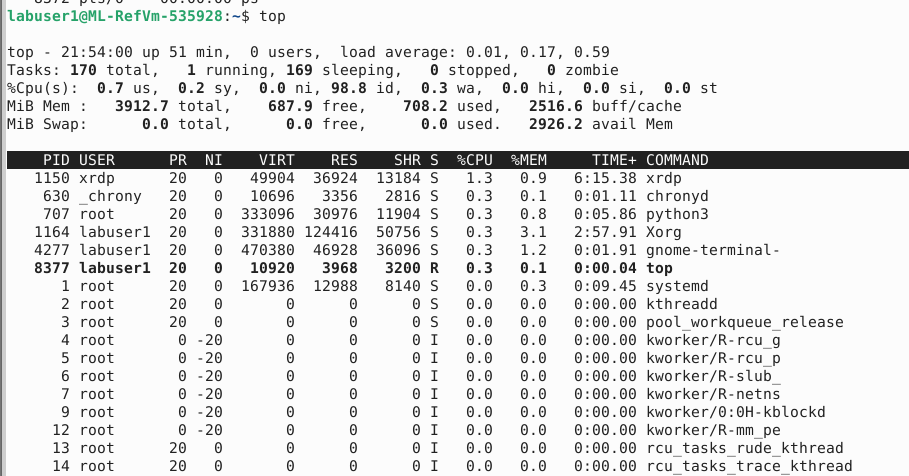
* Display the processes running\*



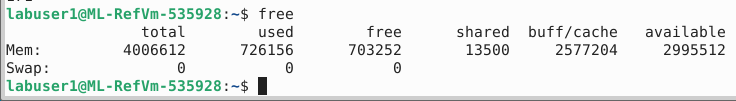
* Display the number of all processes running\*

.

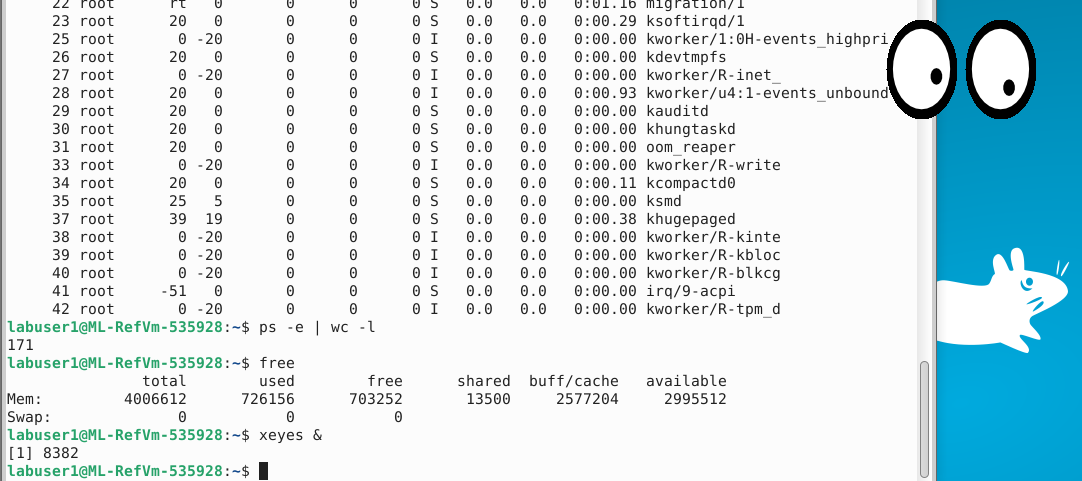
* Display process information\*



* Display memory statistics\*



* Begin running **xeyes** in the background\*



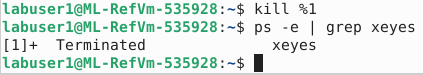
* Display the process information for **xeyes**\*



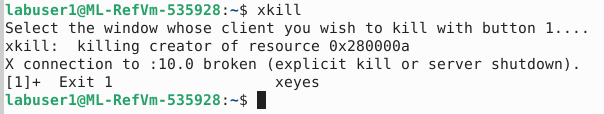
* Display the job information for **xeyes**\*



* Kill **xeyes** using either the process ID or job number\*



* Begin running **xeyes**
* Kill **xeyes** by clicking the process\* (the screenshot of the command used)



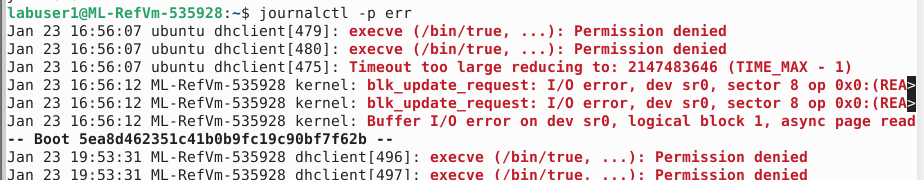
System Logging

System logs are critical for several reasons: These logs provide administrators with useful information to aid in troubleshooting problems. They are also useful in identifying potential hacking attempts. Additionally, logs can be used to provide general information about services, such as which web pages have been provided by a web server. On modern Linux systems, the logging process is handled by the **systemd-journald** service. To query **systemd** log entries, use the **journalctl** command. The log entries can be filtered in a variety of forms such as by priority (**-p**), unit/process (**-u**), and boot logs(**-b**).

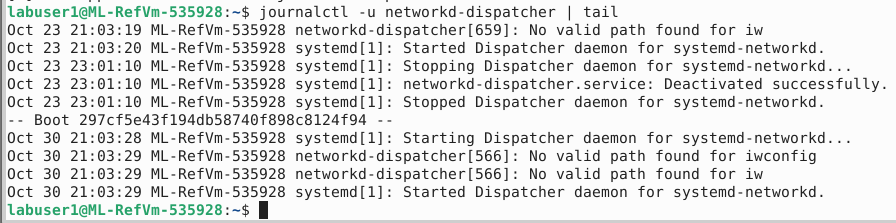
* View the tail of the log entries\*



* View the log entries of the **err** priority\*



* View the tail of the log entries of the **networkd-dispatcher** process\*



* View the tail of the log entries for the previous system boot (**-1**)\*

