Lab 2.4 : Linux System Management

Lab Name:

Systems Administration, Linux System Management (L2-CAT-02-04)

Overview:

This lab discusses the specifics of linux system management, including details of how and when to update, proper password controls, antivirus and antimalware, firewalls, and system logging.

Prerequisites:

- Introduction to command-line
- Introduction to virtual machines (optional)

Materials:

- Computer with access to a terminal or command prompt
- Ubuntu Linux, either in a virtual machine or as the host operating system
- Administrator privileges for certain tasks
- Internet connection (for accessing additional resources)

Learning Objectives:

- How and when to update
- Identify password controls and establish best practices
- Define antivirus and antimalware and compare / contrast the two
- Discuss different firewall softwares and protocols
- System log monitoring and types of logs

Tasks: (Work in Progress)

1. Use the advanced password controls in Ubuntu Linux to create a password that meets the following criteria:
    a. Must be changed quarterly
    b. Has a minimum of 8 characters
    c. Has a minimum of 2 uppercase letters
    d. Has at least one number and one symbol
2. Identify if your system is using UFW or Firewalld
    a. What ports are open?
    b. Can you open port 60?
3. Experiment with the commands provided to view the existing system logs
    a. Open auth.log and understand what you see
    b. Output faillog to another directory

Deliverables:

- A PDF file with the given questions in the review category and your corresponding answers.
- Push the completed lab to your Git repository or submit through the designated method provided by your instructor.

Additional Resources:

- [CISA Password Policies]
- [LibreTexts - UFW & firewalld]

Topic 1: How and When to Update

In Linux, there are three different avenues when it comes to updating:

The first is updating packages, where we update the specific software that is operating on the machine to address bugs or add new features. We can do this through running the update command using our built-in package manager. For example, with the APT package manager we can type ‘sudo apt-get update’ to acquire a list of all the updated packages, followed by ‘sudo apt-get upgrade’ to install the newest version of those packages. For details on how to install updates for specific packages only, you can consult the documentation for your specific package manager.

The second update avenue is updating the operating system, which refers to installing a new version of the linux distribution the machine is operating on. This is done to address operating-system level security issues, to optimize existing functions, or to add new features.

The third is updating the linux kernel, a core component of the operating system that is responsible for managing resources and facilitating communication between hardware and software. Updating the kernel is necessary for major security patches or to support new hardware that is not supported with the existing kernel.

While we may want to update as often as possible, it’s important to note that we need to make sure that the system is stable. This means verifying that dependencies that exist between different packages are maintained and that the update does not introduce new security vulnerabilities.

Topic 2: Password Controls

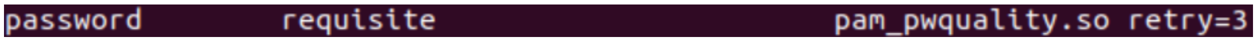
When managing accounts, it’s important to know best practice with regard to password management. Users on the system may set passwords that are not complex or are easy to guess, or never change their passwords. To prevent this, system administrators employ the use of password controls. The three that we will be discussing in this lab are password complexity, password history, and password expiry.

Password complexity is a measure of password strength, or how difficult it is to brute-force a password. The complexity is defined by the password length as well as the characters used (upper and lower case letters, numbers, and special characters). Password complexity requirements vary depending on the organization, but in general the following practices should be followed:

- Passwords should contain at least 8 characters
- Passwords should differ from the previously used one considerably
- Passwords should contain at least one number and special character

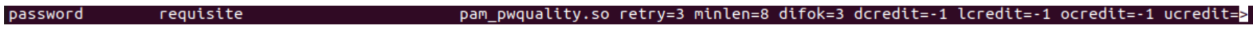
To establish these requirements on the system, we need to configure the settings in /etc/pam.d/common-password. Previously, password settings were found in /etc/login.defs, but this has depreciated over the years.

To edit the password file, type ‘nano /etc/pam.d/common-password’. Here you should find the line:



```
password        requisite                       pam_pwquality.so retry=3
```

To edit our password complexity requirements, we must make some additions to this line. We can adjust it to look something like the following:



```
password        requisite                       pam_pwquality.so retry=3 minlen=8 difok=3 dcredit=-1 lcredit=-1 ocredit=-1 ucredit=>
```

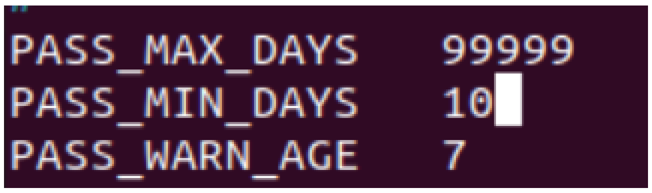
- Retry sets the amount of allowed retries before the account is locked
- Minlen adjusts the minimum password length
- dcredit establishes the required amount of digits
- lcredit establishes the required amount of lowercase letters
- ocredit establishes the required amount of special characters
- ucredit establishes the required amount of uppercase letters

- difok sets requirements on how different the password must be from the previous one. In this case, at least 3 new unique characters must be changed or added.

It is worth noting that the PAM module that is responsible for password requirements has had other libraries in the past, such as pam_unix.so and pam_cracklib.so. You may find these versions on RHEL or CentOS based systems, or older versions of Ubuntu. Most everything is forwards compatible. The newest module and the one we have here is pam_pwquality.so

On top of these complexity requirements, it’s important to establish a setting for password history. The password history dictates when a user is allowed to reuse a password that they have used in the past. This will help to mitigate the security issue of users cycling through the same list of passwords regularly.

In the same line, we can set the setting: ‘remember=10’ to remember the last 10 passwords used. We can then navigate over to /etc/login.defs and uncomment and change the password aging function accordingly:



```
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   10
PASS_WARN_AGE   7
```

Here you can also set the warn age, or when to warn the user they should update their password, as well as the max days a password can be set.

Take some time to explore the various password requirements. For further information about proper password procedures, check the CISA password policies under ‘Additional Resources’.

Topic 3: Antivirus and Antimalware

Another important discussion to have with regard to system management in general is the use of antivirus and antimalware. These software tools are often bundled together or used synonymously, when in fact they are two distinct terms.

Antivirus is software that runs on a computer and uses signature-based detection to identify common system viruses that may be installed. When a program is identified as a virus by the community at large, the collection of bytes that make up that virus is added to a database as a *virus signature*. These databases are then referenced by antivirus softwares to determine if any of the actively running programs contain any known virus signatures, so that those programs can be terminated for the users' protection. This makes antivirus software very good at identifying established threats.

Antimalware on the other hand is a heuristic-based approach. When a program is running on the system, that program has code that executes and produces a specific result. Sometimes, in order to avoid antivirus software, hackers will include useless code in an attempt to disguise the true purpose of the malicious program. Antimalware exists to combat this threat by analyzing code actively running on the machine to identify if that program could be a threat to the user. This makes antimalware much better for identifying the latest threats that have not been established enough, or are custom-tailored to avoid being in an antivirus signature database.

Statistically, less malware is written for the Linux platform than for Mac, and less for Mac than for Windows. This is because Linux makes up a much smaller market share than these other operating systems. Additionally, Linux is open-source, which often allows for faster response times from community members to address vulnerabilities than companies like Apple or Microsoft that abide by release schedules. However, this doesn't mean that Linux is perfectly safe either.

The best practice is to run both an antivirus and antimalware application on the system. This provides the best security coverage, with the antivirus software detecting signatures, and the antimalware picking up any malicious applications that evade signature detection. At the same time, the best practice for security is always to be careful when installing applications, especially as an administrator, and always consider the source. Using the public distribution repositories instead of the internet is the best thing you can do to mitigate any potential for introducing system vulnerabilities.

Topic 4: Firewalls

Just like Windows systems have Windows Defender, Linux systems also have an in-built service for managing network traffic control. The two most common services are *firewalld* and *UFW*. These services provide front-end user interfaces to manage the two backend services directly used to filter traffic: *iptables* and *nftables*.

Firewalld is complex, allowing for specific zone definitions and dynamic firewall rules. It comes pre configured with a more strict set of rules, and is most often used in server environments due to its flexibility with configuration. Firewalld uses nftables by default.

UFW is a simpler front-end firewall software that comes with a more relaxed set of default rules, though it can be configured to be more strict. It typically is run on desktop environments, where the demand for strict security and high performance is lesser. By default it uses iptables.

NFTables is a packet filtering technology with the benefit of a more consistent syntax than iptables, making it easier to learn and less prone to errors. It is also extremely efficient, designed to handle higher throughput and reduce overhead. Nftables is also more flexible in the rules that can be defined than iptables with complex rules, better handling of network protocols, and IPv4 and IPv6 support.

IPTables is an older packet filtering technology, and thus is less efficient and complex than nftables. However, IPtables has existed for far longer and is more widely adopted due to its simplicity and integration with UFW. This community backing supplies more support and resources for this technology than for nftables.

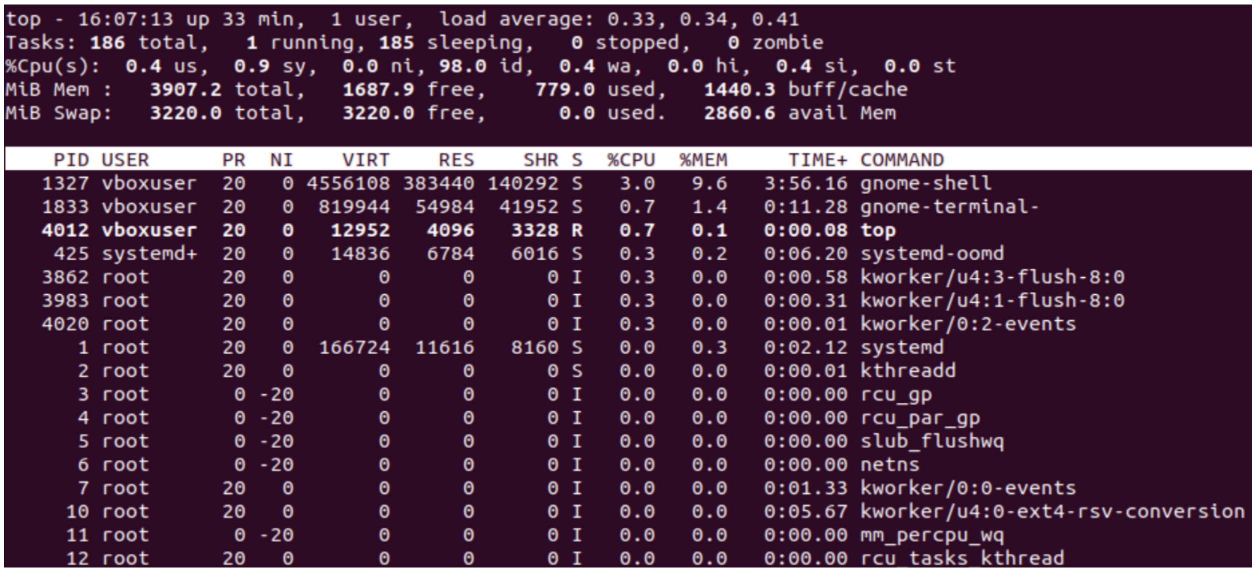
Ultimately, the choice of whether to use firewalld, UFW, nftables, or iptables is preference depending on the system in use and the comfortability on which technologies the administrator is more comfortable working with. It is worth noting that both firewalld and UFW can be configured to use nftables or iptables respectively, but this requires extra overhead (ex. An extra package to translate between the two). System administrators can decide on a case-by-case basis if this is a compromise they are willing to make.

For more information about UFW and firewalld, check out the guide written by LibreTexts under ‘Additional Resources’.

Topic 5: System Monitoring

Another important aspect of Linux administration is system monitoring. This includes being able to track system resources, running processes, and identifying critical log files that can assist in tracking down exploited system vulnerabilities in the event of an attack.

The first system monitoring tool every Linux administrator should be aware of is the ‘top’ command.

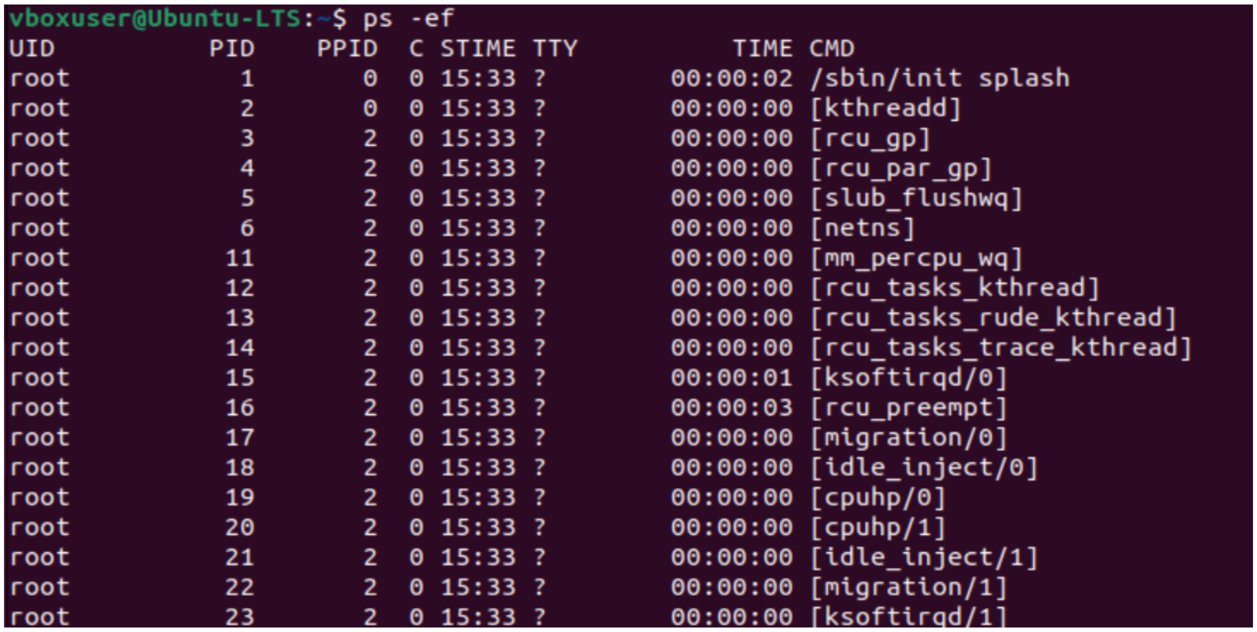


```
top - 16:07:13 up 33 min,  1 user,  load average: 0.33, 0.34, 0.41
Tasks: 186 total,   1 running, 185 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.4 us,  0.9 sy,  0.0 ni, 98.0 id,  0.4 wa,  0.0 hi,  0.4 si,  0.0 st
MiB Mem :   3907.2 total,   1687.9 free,    779.0 used,   1440.3 buff/cache
MiB Swap:   3220.0 total,   3220.0 free,      0.0 used.   2860.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1327 vboxuser  20   0 4556108 383440 140292 S   3.0   9.6   3:56.16 gnome-shell
   1833 vboxuser  20   0  819944  54984  41952 S   0.7   1.4   0:11.28 gnome-terminal-
   4012 vboxuser  20   0   12952   4096   3328 R   0.7   0.1   0:00.08 top
    425 systemd+  20   0   14836   6784   6016 S   0.3   0.2   0:06.20 systemd-oomd
   3862 root      20   0       0      0      0 I   0.3   0.0   0:00.58 kworker/u4:3-flush-8:0
   3983 root      20   0       0      0      0 I   0.3   0.0   0:00.31 kworker/u4:1-flush-8:0
   4020 root      20   0       0      0      0 I   0.3   0.0   0:00.01 kworker/0:2-events
      1 root      20   0  166724  11616   8160 S   0.0   0.3   0:02.12 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      7 root      20   0       0      0      0 I   0.0   0.0   0:01.33 kworker/0:0-events
     10 root      20   0       0      0      0 I   0.0   0.0   0:05.67 kworker/u4:0-ext4-rsv-conversion
     11 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
```

The top command shows a list of the currently allocated system resources, as well as what processes are currently running and what resources they are taking up. This can be useful for identifying if all the physical resources are being detected, and what processes are consuming the most resources.

Another useful command on Linux is the ‘ps’ command, which lists currently active processes on the system.

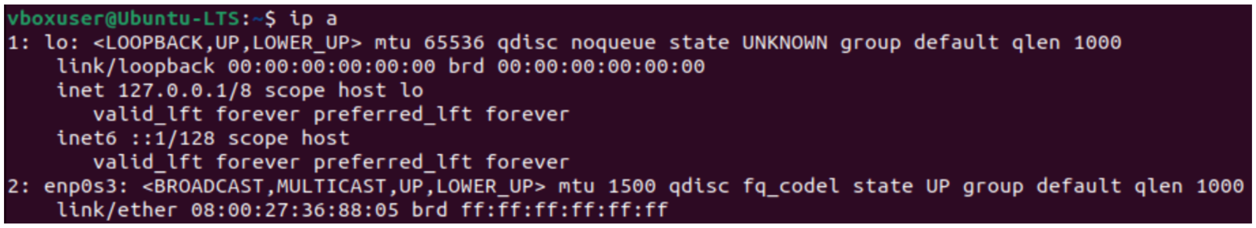

```
vboxuser@Ubuntu-LTS:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 15:33 ?        00:00:02 /sbin/init splash
root           2       0  0 15:33 ?        00:00:00 [kthreadd]
root           3       2  0 15:33 ?        00:00:00 [rcu_gp]
root           4       2  0 15:33 ?        00:00:00 [rcu_par_gp]
root           5       2  0 15:33 ?        00:00:00 [slub_flushwq]
root           6       2  0 15:33 ?        00:00:00 [netns]
root          11       2  0 15:33 ?        00:00:00 [mm_percpu_wq]
root          12       2  0 15:33 ?        00:00:00 [rcu_tasks_kthread]
root          13       2  0 15:33 ?        00:00:00 [rcu_tasks_rude_kthread]
root          14       2  0 15:33 ?        00:00:00 [rcu_tasks_trace_kthread]
root          15       2  0 15:33 ?        00:00:01 [ksoftirqd/0]
root          16       2  0 15:33 ?        00:00:03 [rcu_preempt]
root          17       2  0 15:33 ?        00:00:00 [migration/0]
root          18       2  0 15:33 ?        00:00:00 [idle_inject/0]
root          19       2  0 15:33 ?        00:00:00 [cpuhp/0]
root          20       2  0 15:33 ?        00:00:00 [cpuhp/1]
root          21       2  0 15:33 ?        00:00:00 [idle_inject/1]
root          22       2  0 15:33 ?        00:00:00 [migration/1]
root          23       2  0 15:33 ?        00:00:00 [ksoftirqd/1]
```

This command has a variety of arguments that can be implemented to list processes from a specific user on the system, processes with a specific PID number, top memory consuming processes, and others. You can use the ‘man’ command to explore some of these arguments in more depth. Above, we’ve used ‘ps -ef’ to list kernel processes, which are different from user processes found with ‘ps -l’. If you need to dispose of a process that is consuming a lot of resources, you can use commands such as ‘kill X’, where X is the PID of the process. Just be sure not to terminate important processes!

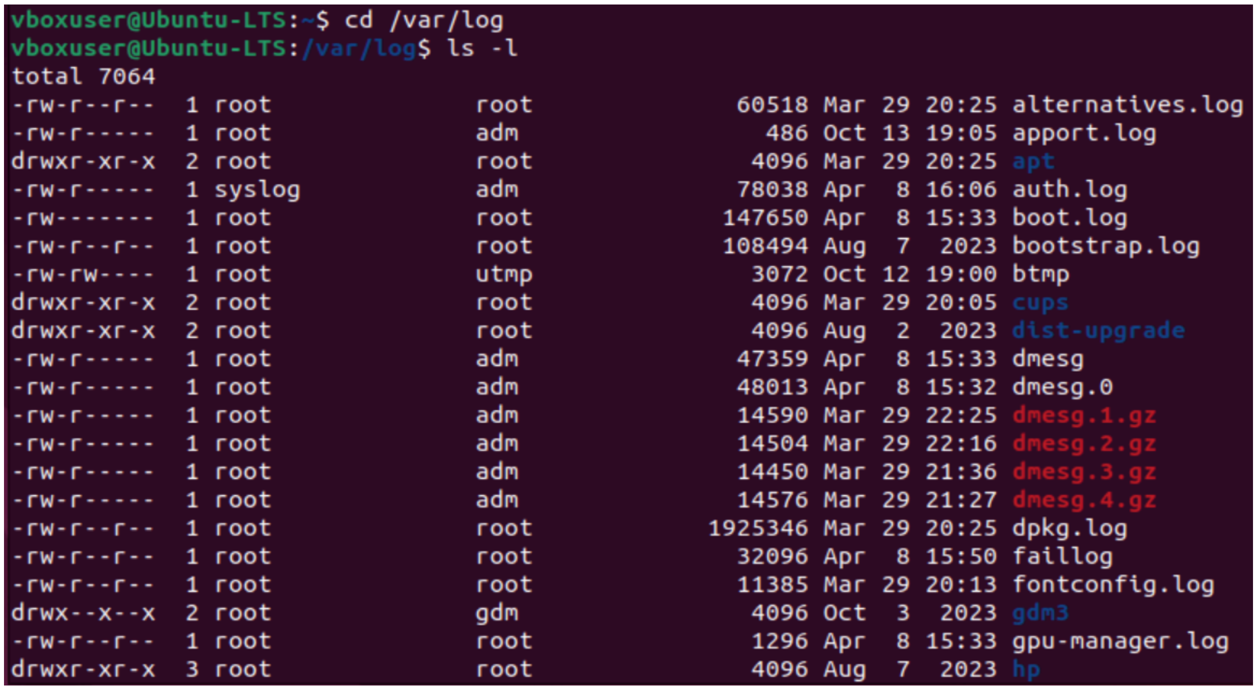
The *ip* command can be used to output networking information. Below we’ve used ‘ip a’ to list ip address information, but we can also list the following:

- ‘ip l’ for link information
- ‘ip r’ for routing information
- ‘ip ru’ for routing policy rules

And many others. Type ‘man ip’ into the terminal for further information.


```
vboxuser@Ubuntu-LTS:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:36:88:05 brd ff:ff:ff:ff:ff:ff
```

Finally, to inspect linux logs you can always check the directory '/var/log'. Across all distributions, this is the location that logs are saved for reference. Types of logs include system, event, application, kernel logs, and more.



```
vboxuser@Ubuntu-LTS:~$ cd /var/log
vboxuser@Ubuntu-LTS:/var/log$ ls -l
total 7064
-rw-r--r--  1 root              root              60518 Mar 29 20:25 alternatives.log
-rw-r-----  1 root              adm                 486 Oct 13 19:05 apport.log
drwxr-xr-x  2 root              root               4096 Mar 29 20:25 apt
-rw-r-----  1 syslog            adm               78038 Apr  8 16:06 auth.log
-rw-------  1 root              root             147650 Apr  8 15:33 boot.log
-rw-r--r--  1 root              root             108494 Aug  7  2023 bootstrap.log
-rw-rw----  1 root              utmp               3072 Oct 12 19:00 btmp
drwxr-xr-x  2 root              root               4096 Mar 29 20:05 cups
drwxr-xr-x  2 root              root               4096 Aug  2  2023 dist-upgrade
-rw-r-----  1 root              adm               47359 Apr  8 15:33 dmesg
-rw-r-----  1 root              adm               48013 Apr  8 15:32 dmesg.0
-rw-r-----  1 root              adm               14590 Mar 29 22:25 dmesg.1.gz
-rw-r-----  1 root              adm               14504 Mar 29 22:16 dmesg.2.gz
-rw-r-----  1 root              adm               14450 Mar 29 21:36 dmesg.3.gz
-rw-r-----  1 root              adm               14576 Mar 29 21:27 dmesg.4.gz
-rw-r--r--  1 root              root            1925346 Mar 29 20:25 dpkg.log
-rw-r--r--  1 root              root              32096 Apr  8 15:50 faillog
-rw-r--r--  1 root              root              11385 Mar 29 20:13 fontconfig.log
drwx--x--x  2 root              gdm                4096 Oct  3  2023 gdm3
-rw-r--r--  1 root              root               1296 Apr  8 15:33 gpu-manager.log
drwxr-xr-x  3 root              root               4096 Aug  7  2023 hp
```

System logs contain some generic information about the system, and are the first go-to log to look for information about any activity, errors, or network status. They provide a surface-level view of the system activities that is elaborated upon in more specific log entries.

Event logs are logs that provide further details about specific events that occurred on the system.

Application logs are those logs that come from specific applications. These can be useful for determining why certain application instabilities exist, or if there are dependency issues. Check these logs for further elaboration.

Kernel logs are a record of all of the kernel activities, including connecting to hardware devices, system calls, and on-boot initialization among others. Though not as regularly used as other logs, these are essential for system administrators for cases where the system fails to boot, or hardware is unresponsive.

Topic 6: Review

1. What is the difference between a kernel update and operating system update?
2. What is password history? Why is it important?
3. What is the difference between antivirus and antimalware? Should you use just one, or both? Why?
4. What backend service does Firewalld utilize, and in what environment is it typically seen?
5. If the system is running extremely slowly, what command would you use to check what processes are running that are taking up the most resources?
6. To terminate a process, what command would you use?