VxLEARN Networks

Networking & Cybersecurity Track
Simulated Employment Program

Lab Report:
Reading Server Logs

Prepared by:
Kudzaishe Majeza
Junior Network Engineer – VxLEARN Networks

Mentor:
Titus Majeza
Senior Network Engineer

Date: 12 January 2026

Contents

Objective 3

Tools and Environment 3

Part 1: Reading Log Files 3

Step 1: Opening Log Files. 4

Step 2: Actively Following Logs. 8

Part 2: Syslog 10

Part 3: Journald 12

Step 1: Running journalctl with no options. 13

Step 2: Journalctl and a few options. 14

Reflection 16

Objective

The objective of this lab is to understand how to read, monitor, and analyze server log files using common Linux command-line tools. The lab also explores centralized logging using Syslog and Journald.

Tools and Environment

- Security Workstation Virtual Machine
- Linux Terminal
- Log analysis commands: cat, more, less, tail, journalctl

Part 1: Reading Log Files

In this section, various commands were used to read log files.

Cat Command:
The cat command displays the entire content of a file at once. A drawback of using cat with large files is that the output scrolls quickly, making it difficult to review specific information.

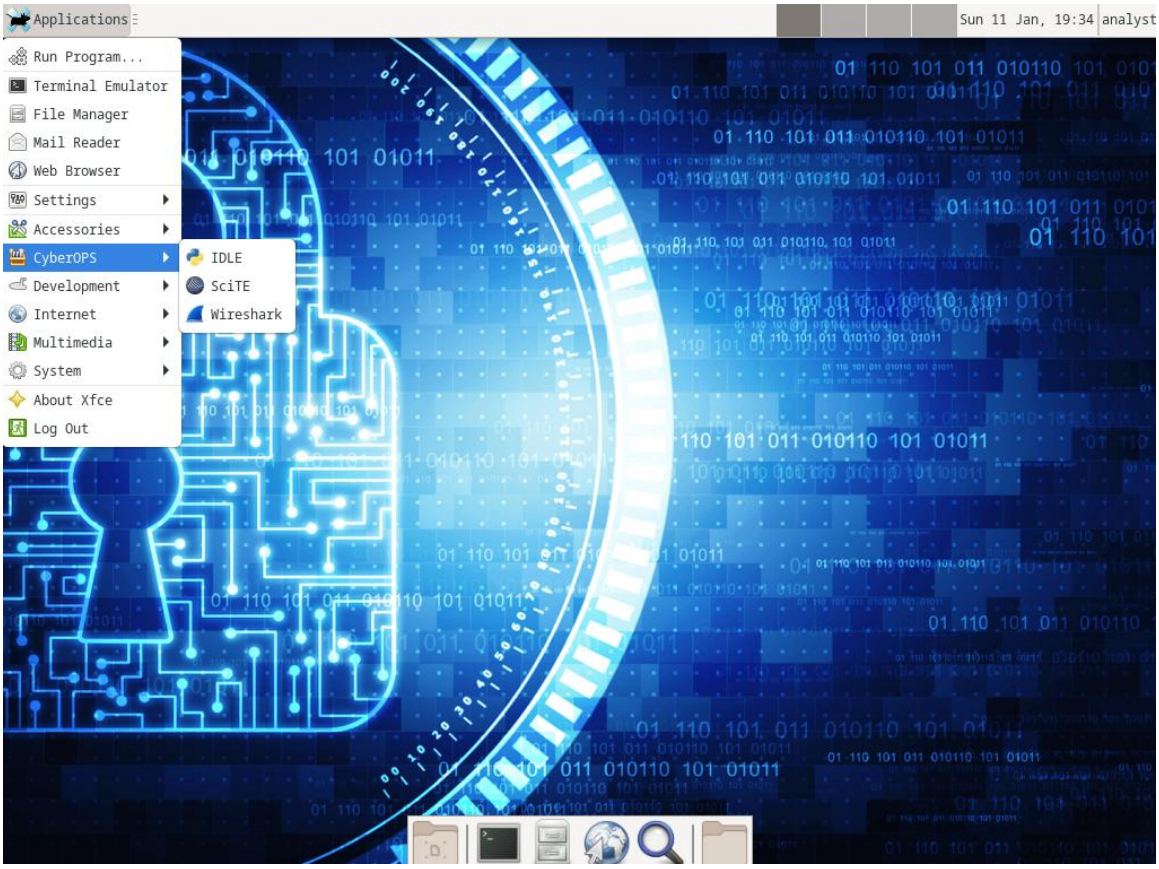
More Command:
The more command displays files page by page. A drawback of more is that it only allows forward navigation and does not support scrolling backward.

Less Command:
The less command improves usability by allowing both forward and backward navigation. It is ideal for analyzing large log files.

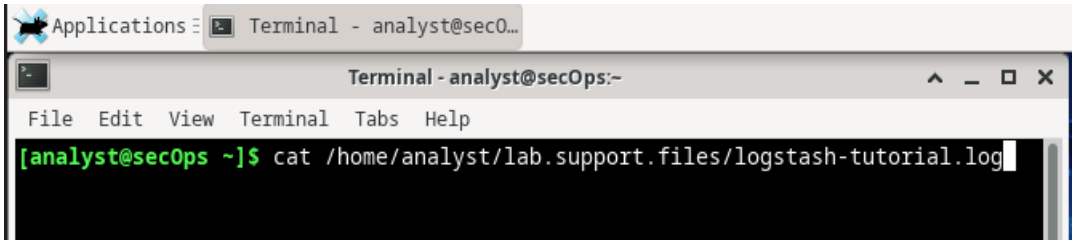
Tail and Tail -f:
The tail command shows the last lines of a file, while tail -f continuously updates the display as new entries are added. Tail -f is useful for real-time log monitoring.

Step 1: Opening Log Files.

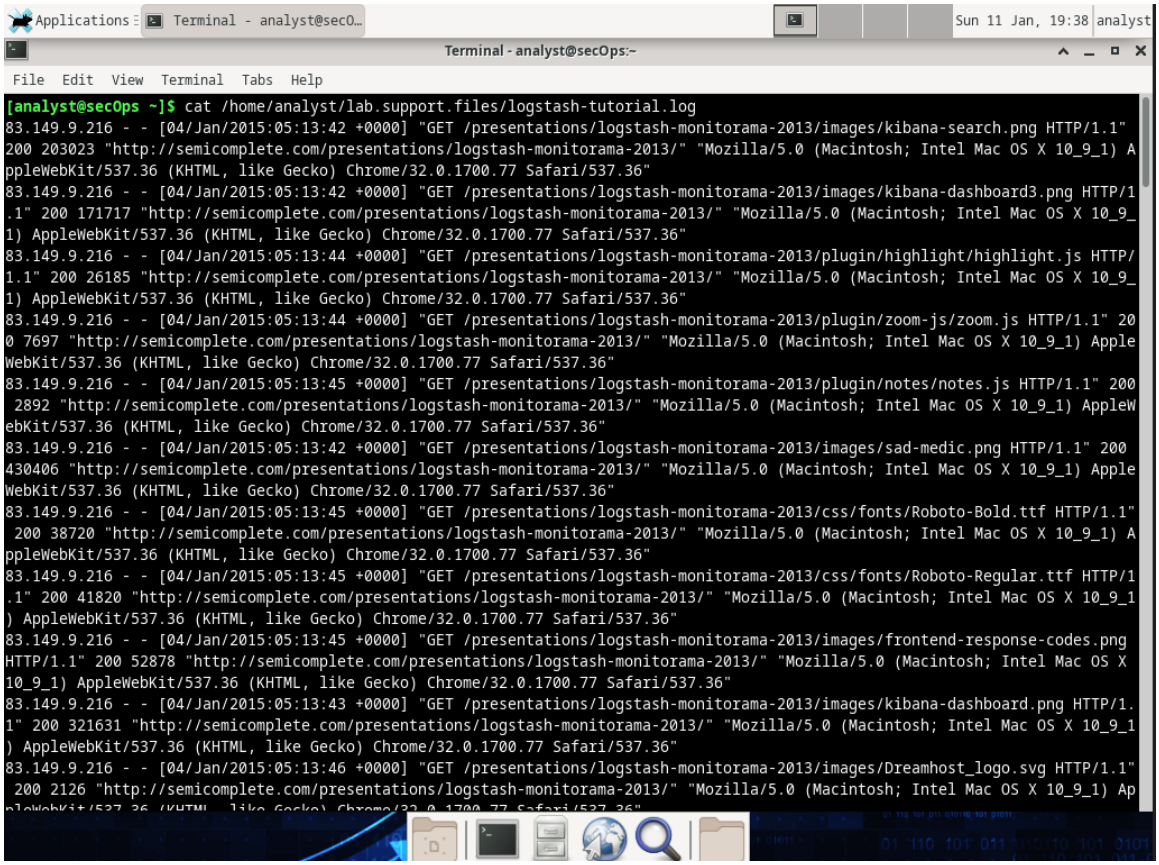
a. Start the **CyberOps Worstation VM** and open a terminal window.



b. From the terminal window, issue the command below to display the contents of the **logstash-tutorial.log** file, located in the **/home/analyst/lab.support.files/** folder:

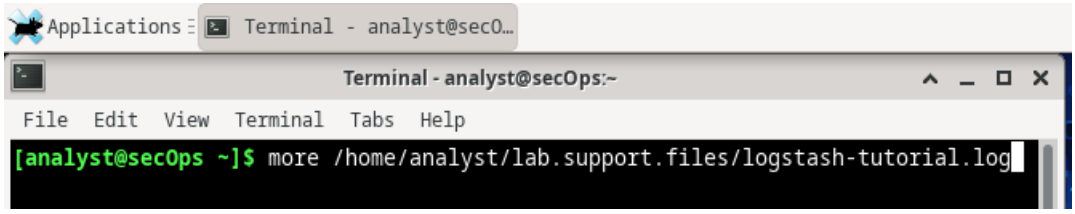


The contents of the file should scroll through the terminal window until the all contents have been displayed.

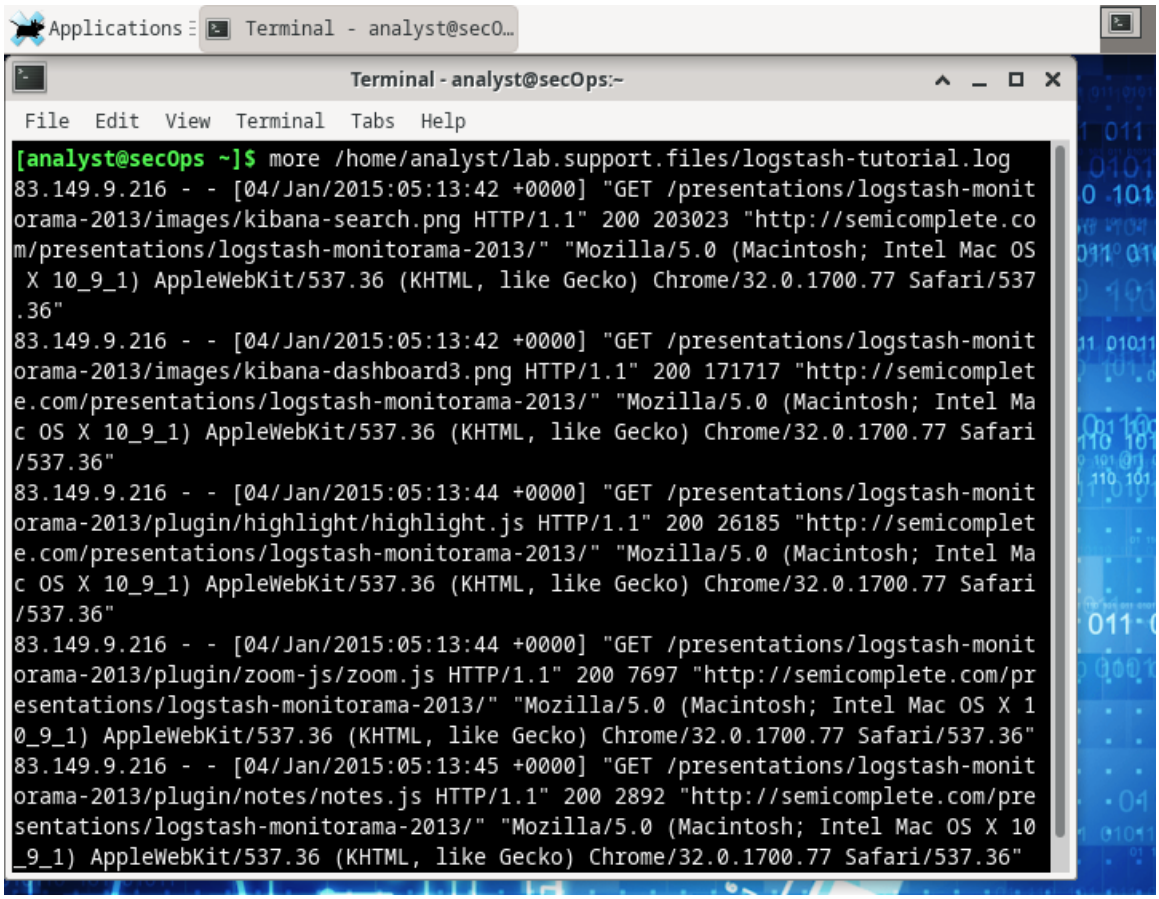


Another popular tool for visualizing log files is **more**. Similar to **cat, more** is also a UNIX command-line-based tool that can open a text-based file and display the file contents on the screen. The main difference between **cat** and **more** is that **more** supports page breaks, allowing the user to view the contents of a file, one page at a time. This can be done using the space bar to display the next page.

c. From the same terminal window, use the command below to display the contents of the **logstash-tutorial.log** file again. This time using **more:**

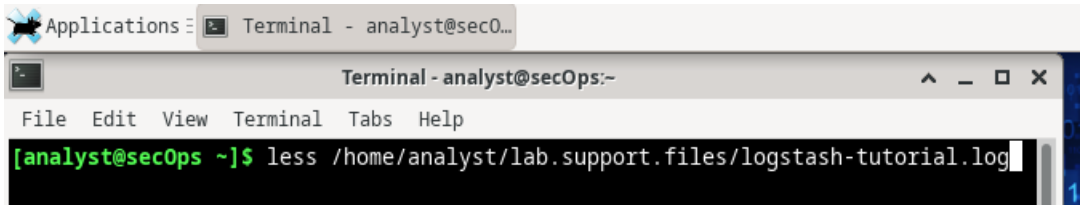


The contents of the file should scroll through the terminal window and stop when one page is displayed. Press the space bar to advance to the next page. Press enter to display the next line of text.

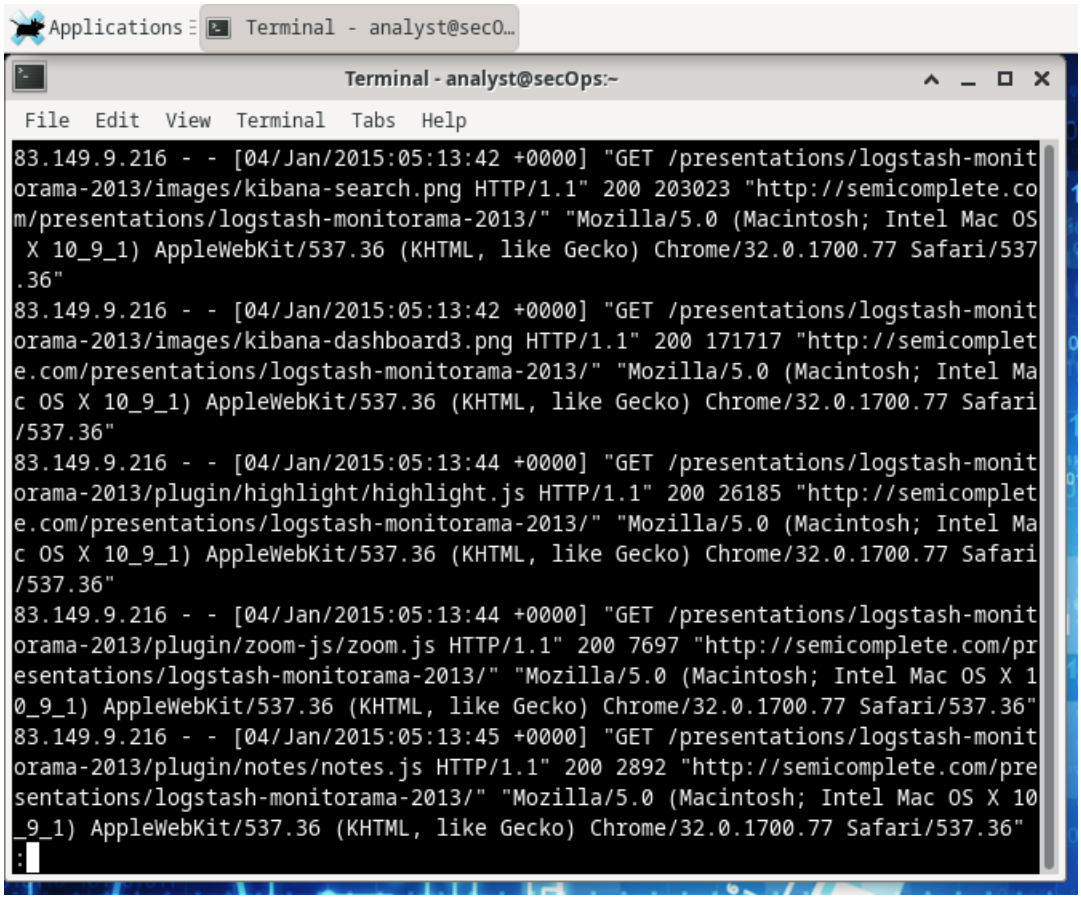


Building on the functionality of **cat** and **more**, the **less** tool allows the contents of a file to be displayed page by page, while also allowing the user the choice of viewing previously displayed pages.

d. From the same terminal window, use **less** to display the contents the **logstash-tutorial.log** file again:

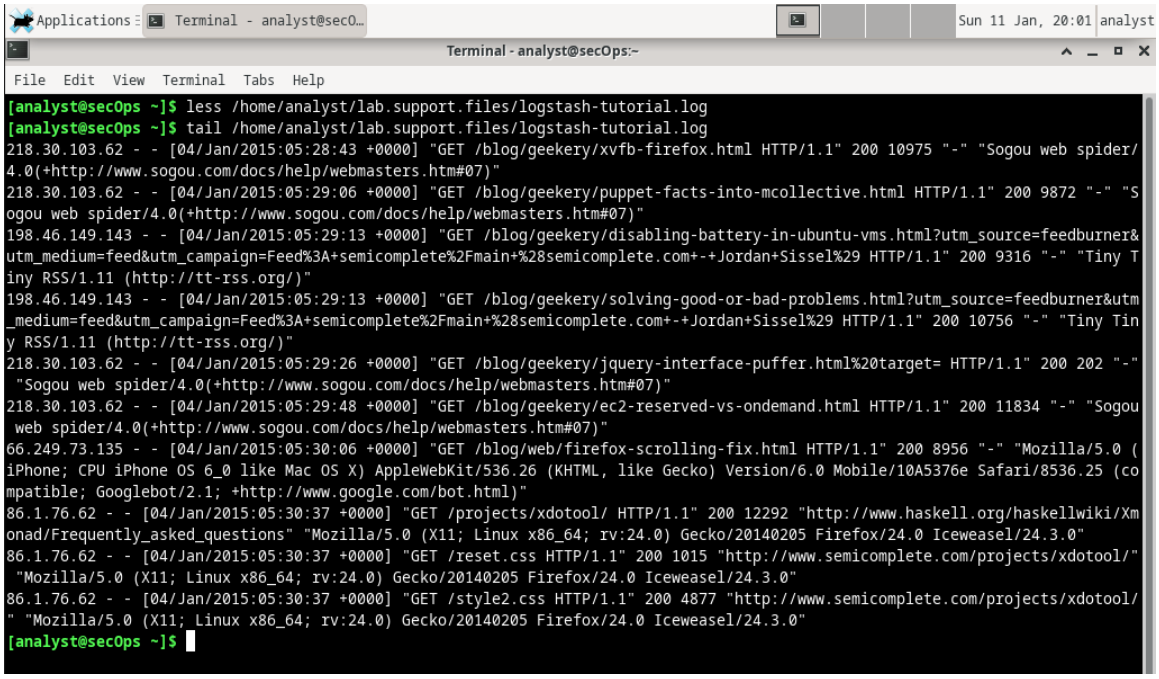


The contents of the file should scroll through the terminal window and stop when one page is displayed. Press the space bar to advance to the next page. Press enter to display the next line of text. Use the up and down arrow keys to move back and forth through the text file.



e. The **tail** command displays the end of a text file. By default, **tail** displays the last ten lines of the file.

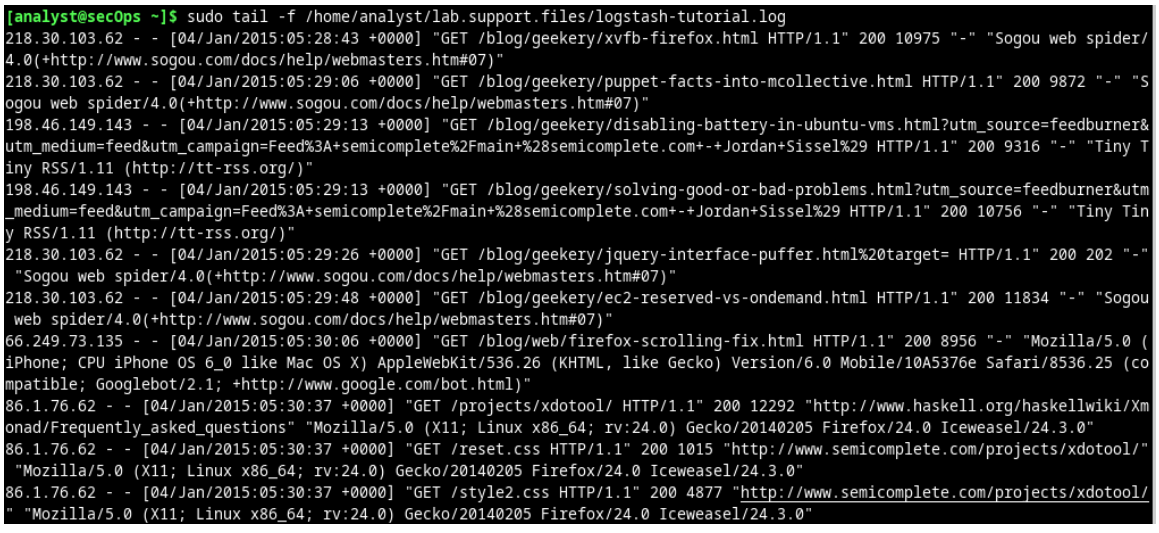
Use tail to display the last ten lines of the **/home/analyst/lab.support.files/logstash-tutorial.log** file.



Step 2: Actively Following Logs.

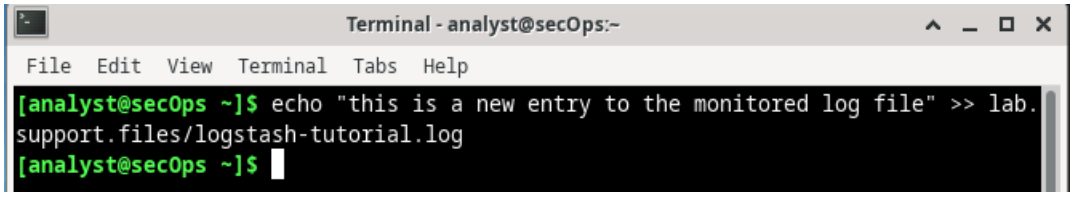
In some situations, it is desirable to monitor log files as log entries are written to the log files. For those cases, the **tail -f** command is very helpful.

a. Use tail -f to actively monitor the contents of the **/var/log/syslog** file:

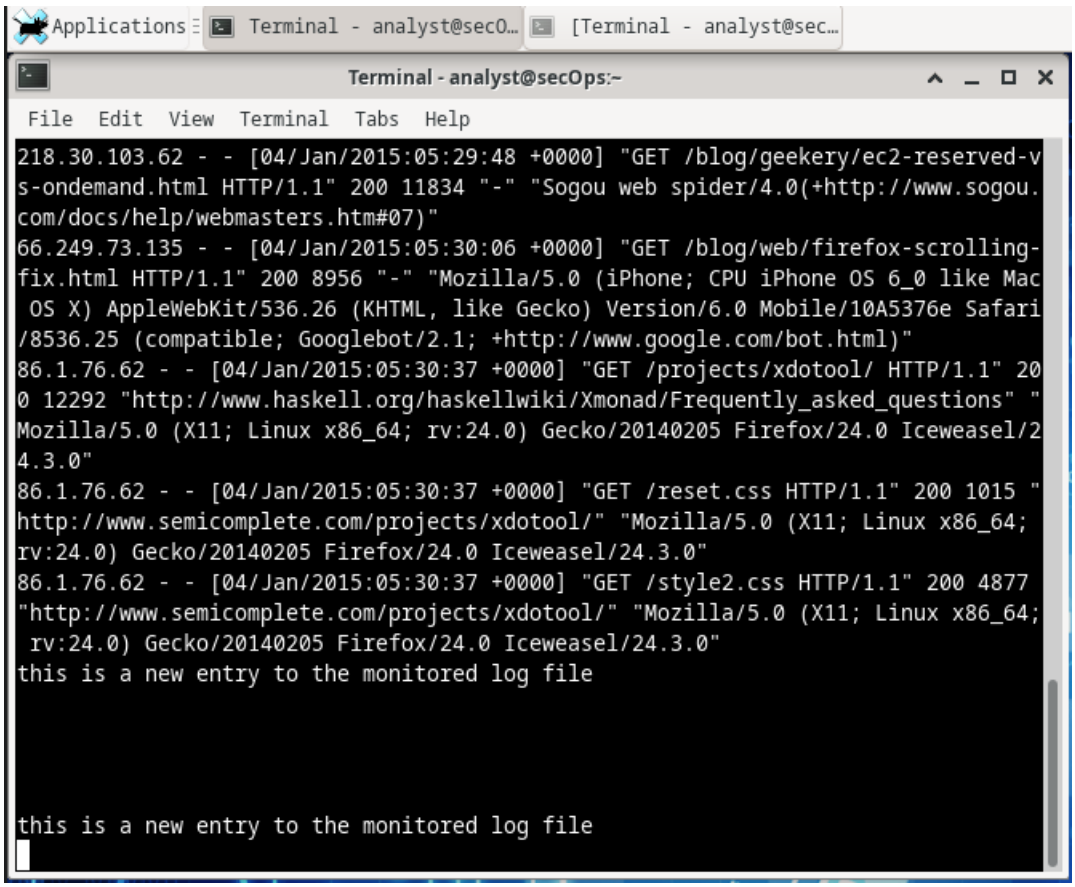


b. To watch tail –f in action, open a second terminal window. Arrange your display so you can see both terminal windows. Re-size the windows so you can see them both at the same, as shown in the image below:

The terminal window on the top is running **tail -f** to monitor the **/home/analyst/lab.support.files/logstash-tutorial.log** file. Use the terminal window on the bottom to add information to the monitored file.



To make it easier to visualize, select the top terminal window (the one running **tail -f**) and press enter a few times. This will add a few lines between the current contents of the file and the new information to be added.



c. Select the bottom terminal window and enter the following command:

```
[analyst@secOps ~]$ echo "this is a new entry to the monitored log file" >> lab.
support.files/logstash-tutorial.log
[analyst@secOps ~]$
```

The command above appends the #this is a new entry to the monitored log file# message to the **/home/analyst/lab.support.files/logstash-tutorial.log** file. Because tail –f is monitoring the file at the moment a line is added to the file. The top window should display the new line in real-time.

d. Press CTRL + C to stop the execution of **tail -f** and return to the shell prompt.

e. Close one of the two terminal windows.

Part 2: Syslog

Syslog is a centralized logging system that collects operating system and application logs.

Root Access:
The cat command had to be executed as root because syslog files contain sensitive system information and are protected to prevent unauthorized access.

Log Rotation:
Older syslog files are rotated and renamed (syslog.1, syslog.2, etc.) to prevent files from becoming too large.

Time Synchronization:
Accurate system time is essential for log analysis, troubleshooting, and security investigations. Incorrect timestamps can lead to misinterpretation of events.

Because of their importance, it is common practice to concentrate log files in one monitoring computer. **Syslog** is a system designed to allow devices to send their log files to a centralized server, known as **syslog** server. Clients

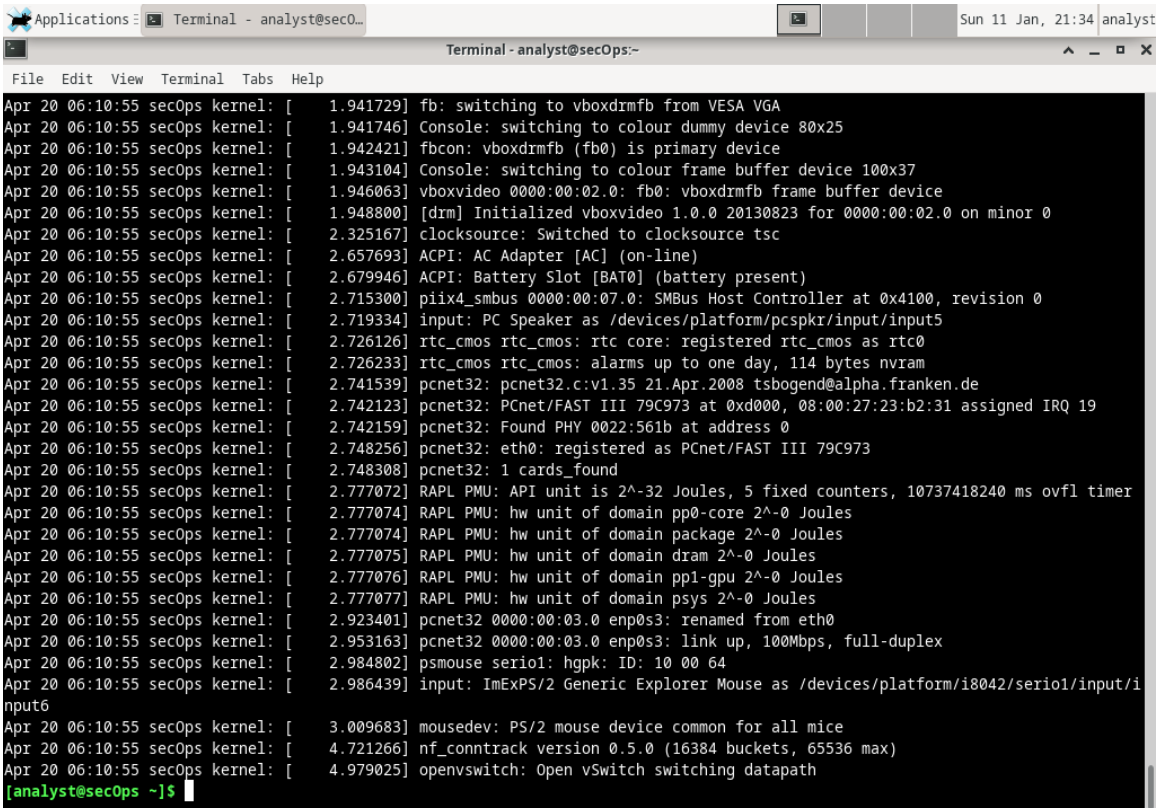
communicate to a syslog server using the **syslog** protocol. **Syslog** is commonly deployed and supports practically all computer platforms.

The CyberOps Workstation VM generates operating system level log files and hands them over to **syslog**.

a. Use the **cat** command as **root** to list the contents of the **/var/log/syslog.1** file. This file holds the log entries that are generated by the CyberOps Workstation VM operating system and sent to the **syslog** service.

analyst@secOps ~$ **sudo cat /var/log/syslog.1**

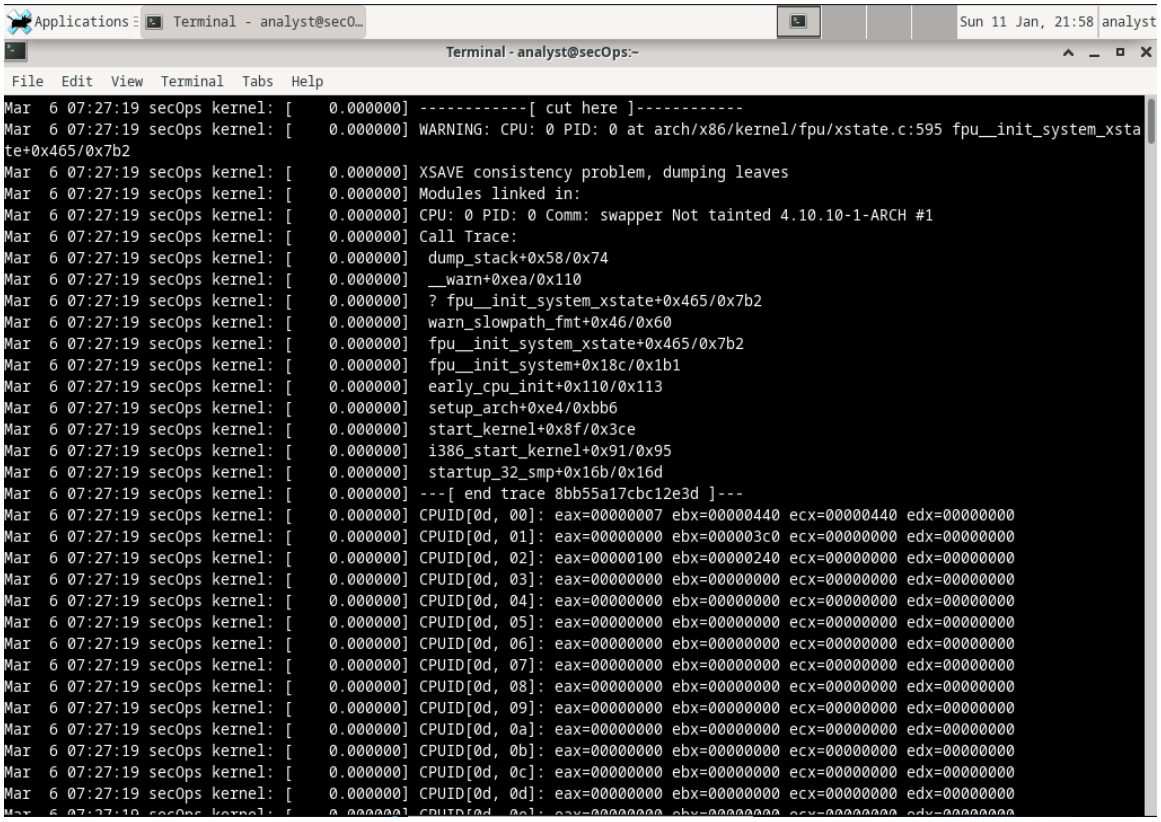
[sudo] password for analyst:



b. Notice that the **/var/log/syslog** file only stores the most recent log entries. To keep the syslog file small, the operating system periodically rotates the log files, renaming older log files as **syslog.1, syslog.2**, and so on.

Use the **cat** command to list older **syslog** files:

analyst@secOps ~$ **sudo cat /var/log/syslog.2**
analyst@secOps ~$ **sudo cat /var/log/syslog.3**
analyst@secOps ~$ **sudo cat /var/log/syslog.4**



Part 3: Journald

Journald manages logs using binary files and centralizes log data from multiple sources.

Journalctl:
The journalctl command allows filtering by boot session, service, kernel messages, and time range.

Advantages:
- Structured logging
- Advanced filtering
- Tamper resistance

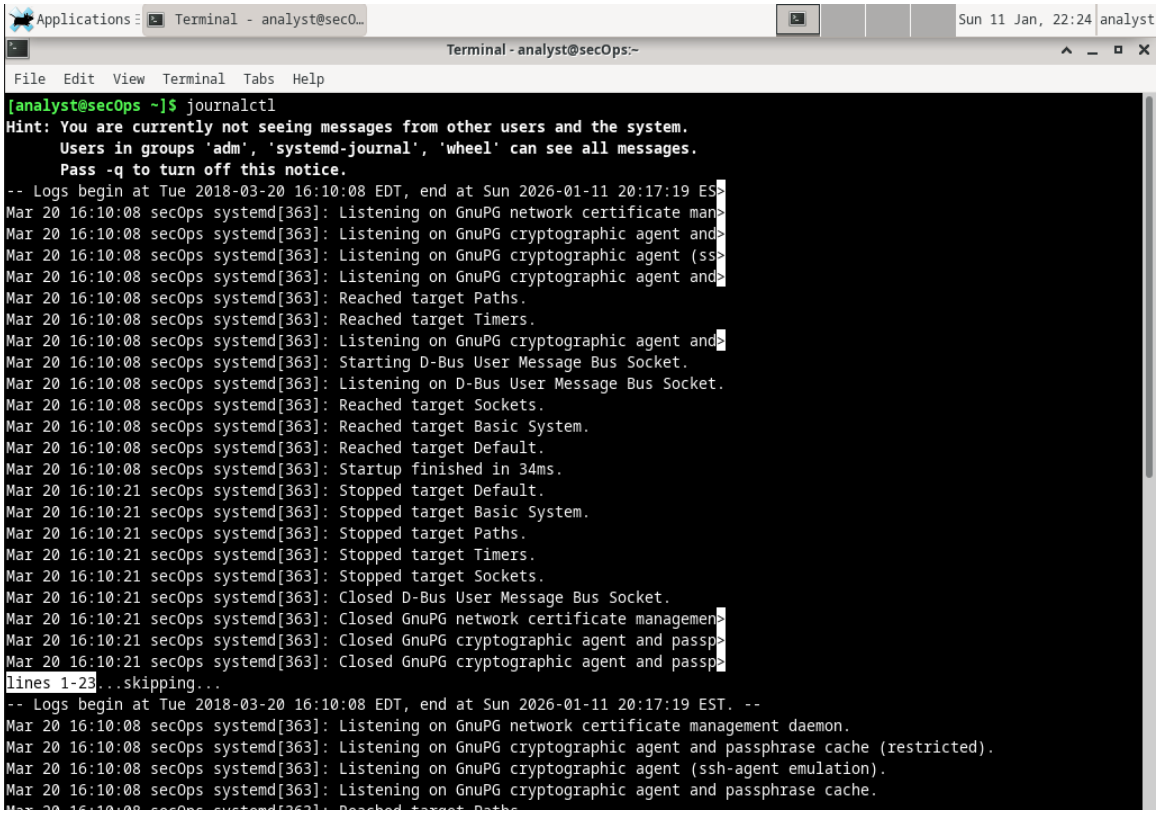
Disadvantages:

- Binary format is not human-readable without journalctl
- Slightly higher resource usage

In the context of this lab, the most evident feature of the **journal** system daemon is the use of append-only binary files serving as its **log files**.

Step 1: Running journalctl with no options.

a. To look at the **journald** logs, use the **journalctl** command.
The **journalctl** tool interprets and displays the log entries previously stored in the **journal** binary log files.



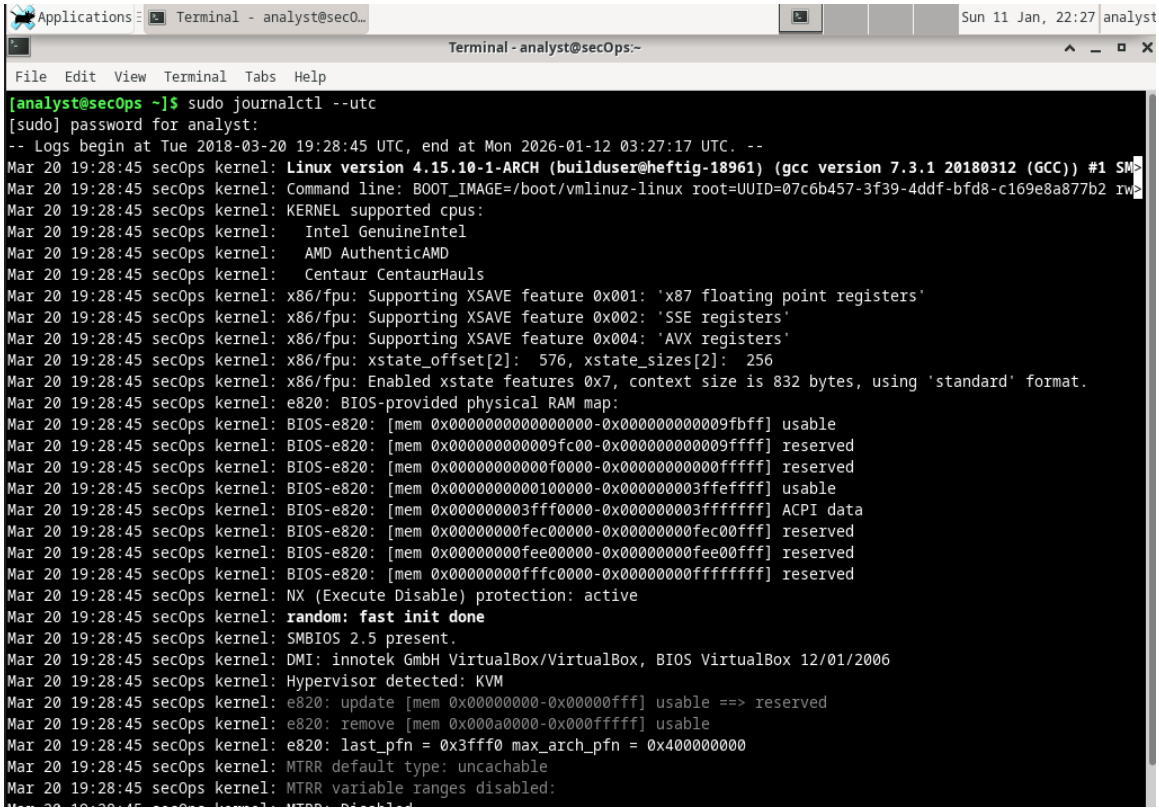
Note: Running journalctl as root will display more detailed information.

b. Use CTRL+C to exit the display.

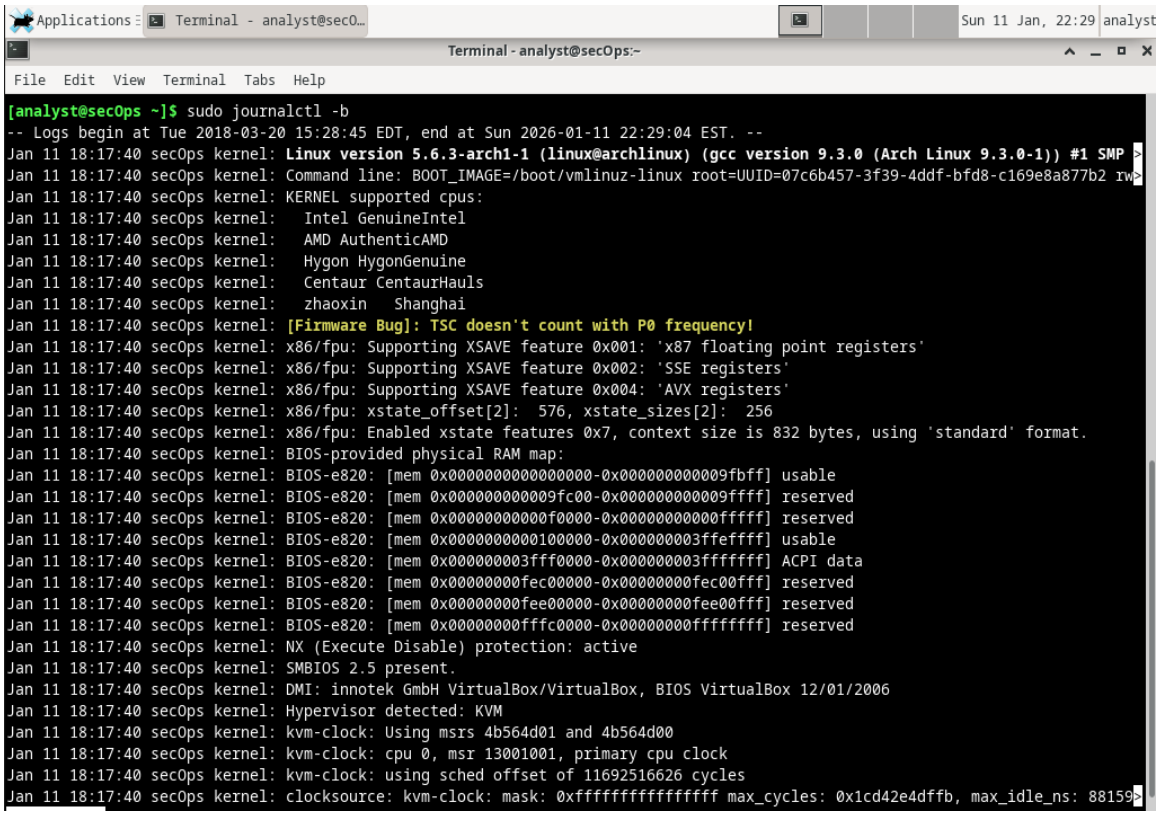
Step 2: Journalctl and a few options.

Part of the power of using journalctl lies in its options. For the following commands, use CRTL+C to exit the display.

a. Use **journalctl --utc** to display all timestamps in UTC time:



b. Use **journalctl -b** to display log entries recorded during the last boot:



c. Use **journalctl** to specify the service and timeframe for log entries. The command below shows all **nginx** service logs recorded today:

```
[analyst@secOps ~]$ sudo journalctl -u nginx.service --since today
-- Logs begin at Tue 2018-03-20 15:28:45 EDT, end at Sun 2026-01-11 22:33:53 EST. --
-- No entries --
```

d. Use the **-k** switch to display only messages generated by the kernel:

```
[analyst@secOps ~]$ sudo journalctl -k
[sudo] password for analyst:
Sorry, try again.
[sudo] password for analyst:
-- Logs begin at Tue 2018-03-20 15:28:45 EDT, end at Sun 2026-01-11 22:37:28 EST. --
Jan 11 18:17:40 secOps kernel: Linux version 5.6.3-arch1-1 (linux@archlinux) (gcc version 9.3.0 (Arch Linux 9.3.0-1)) #1 SMP >
Jan 11 18:17:40 secOps kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-linux root=UUID=07c6b457-3f39-4ddf-bfd8-c169e8a877b2 rw>
Jan 11 18:17:40 secOps kernel: KERNEL supported cpus:
Jan 11 18:17:40 secOps kernel:   Intel GenuineIntel
Jan 11 18:17:40 secOps kernel:   AMD AuthenticAMD
Jan 11 18:17:40 secOps kernel:   Hygon HygonGenuine
Jan 11 18:17:40 secOps kernel:   Centaur CentaurHauls
Jan 11 18:17:40 secOps kernel:   zhaoxin   Shanghai
Jan 11 18:17:40 secOps kernel: [Firmware Bug]: TSC doesn't count with P0 frequency!
Jan 11 18:17:40 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
Jan 11 18:17:40 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
Jan 11 18:17:40 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
Jan 11 18:17:40 secOps kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
Jan 11 18:17:40 secOps kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.
Jan 11 18:17:40 secOps kernel: BIOS-provided physical RAM map:
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x0000000000100000-0x00000003ffefffff] usable
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x00000003fff0000-0x00000003ffffffff] ACPI data
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved
Jan 11 18:17:40 secOps kernel: BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
Jan 11 18:17:40 secOps kernel: NX (Execute Disable) protection: active
Jan 11 18:17:40 secOps kernel: SMBIOS 2.5 present.
Jan 11 18:17:40 secOps kernel: DMI: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
Jan 11 18:17:40 secOps kernel: Hypervisor detected: KVM
Jan 11 18:17:40 secOps kernel: kvm-clock: Using msrs 4b564d01 and 4b564d00
```

e. Similar to **tail -f** described above, use the **-f** switch to actively follow the logs as they are being written:

```
[analyst@secOps ~]$ sudo journalctl -f
-- Logs begin at Tue 2018-03-20 15:28:45 EDT. --
Jan 11 22:38:53 secOps audit[1410]: USER_END pid=1410 uid=0 auid=1000 ses=2 msg='op=PAM:session_close grantors=pam_limits,pam_
unix,pam_permit acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'
Jan 11 22:38:53 secOps audit[1410]: CRED_DISP pid=1410 uid=0 auid=1000 ses=2 msg='op=PAM:setcred grantors=pam_unix,pam_permit
acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'
Jan 11 22:39:15 secOps audit[1423]: USER_ACCT pid=1423 uid=1000 auid=1000 ses=2 msg='op=PAM:accounting grantors=pam_unix,pam_p
ermit,pam_time acct="analyst" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'
Jan 11 22:39:15 secOps sudo[1423]:  analyst : TTY=pts/2 ; PWD=/home/analyst ; USER=root ; COMMAND=/usr/bin/journalctl -f
Jan 11 22:39:15 secOps kernel: audit: type=1101 audit(1768189155.039:120): pid=1423 uid=1000 auid=1000 ses=2 msg='op=PAM:accou
nting grantors=pam_unix,pam_permit,pam_time acct="analyst" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=succe
ss'
Jan 11 22:39:15 secOps audit[1423]: CRED_REFR pid=1423 uid=0 auid=1000 ses=2 msg='op=PAM:setcred grantors=pam_unix,pam_permit,
pam_env acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'
Jan 11 22:39:15 secOps sudo[1423]: pam_unix(sudo:session): session opened for user root by (uid=0)
Jan 11 22:39:15 secOps audit[1423]: USER_START pid=1423 uid=0 auid=1000 ses=2 msg='op=PAM:session_open grantors=pam_limits,pam
_unix,pam_permit acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'
Jan 11 22:39:15 secOps kernel: audit: type=1110 audit(1768189155.046:121): pid=1423 uid=0 auid=1000 ses=2 msg='op=PAM:setcred
grantors=pam_unix,pam_permit,pam_env acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'
Jan 11 22:39:15 secOps kernel: audit: type=1105 audit(1768189155.046:122): pid=1423 uid=0 auid=1000 ses=2 msg='op=PAM:session_
open grantors=pam_limits,pam_unix,pam_permit acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success
'
```

Reflection

Syslog is simple, widely supported, and uses plain-text logs, making it easy to integrate with other tools. However, it lacks advanced filtering. Journald provides powerful querying and structured logging but relies on binary files, which require specific tools to read. Both systems are effective depending on use case.