

Lab - Reading Server Logs

Objectives

Part 1: Reading Log Files with Cat, More, Less, and Tail

Part 2: Log Files and Syslog

Part 3: Log Files and Journalctl

Background / Scenario

Log files are an important tool for troubleshooting and monitoring. Different applications generate different log files, each one containing its own set of fields and information. While the field structure may change between log files, the tools used to read them are mostly the same. In this lab, you will learn about common tools used to read log files and practice using them.

Required Resources

- CyberOps Workstation virtual machine

Instructions

Part 1: Reading Log Files with Cat, More, Less, and Tail

Log files are files used to record specific events triggered by applications, services or the operating system itself. Usually stored as plain-text, log files are an indispensable resource for troubleshooting.

Step 1: Opening Log Files.

Log files commonly contain plain-text information which can be viewed by practically any program able to handle text (text editors, for example). However, because of convenience, usability, and speed, a few tools are more commonly used than others. This section focuses on four command-line-based programs: **cat**, **more**, **less**, and **tail**.

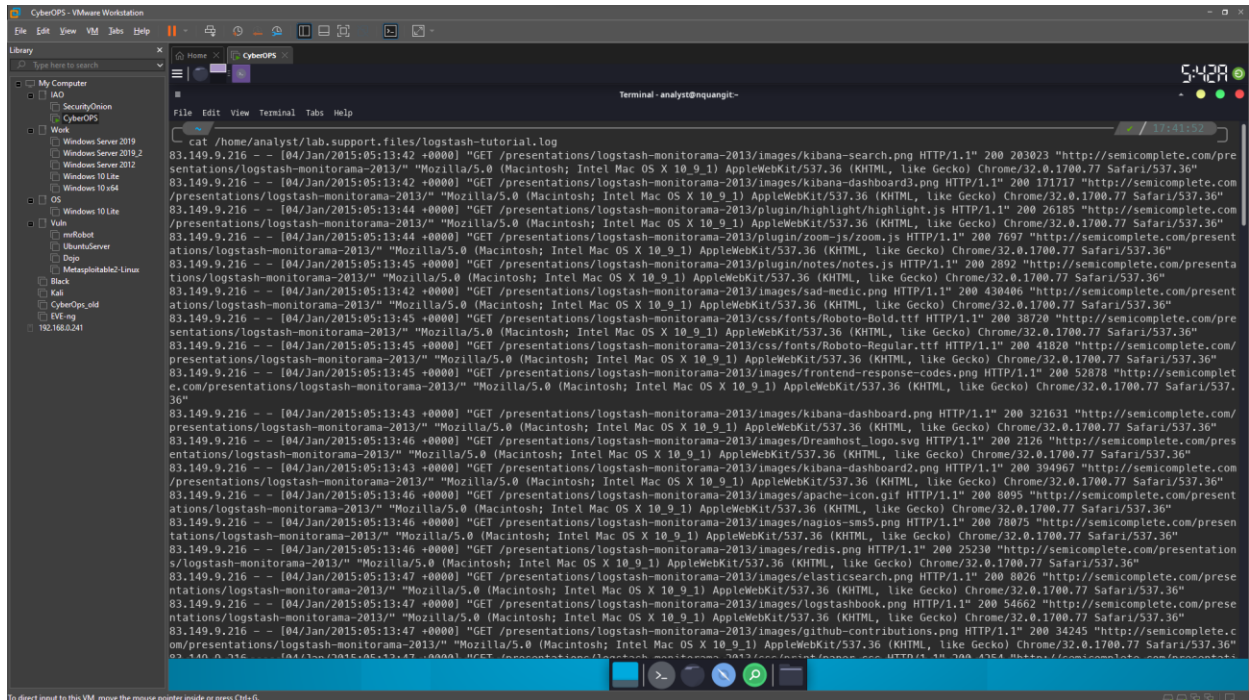
cat, derived from the word 'concatenate', is a UNIX, command-line-based tool used to read and display the contents of a file on the screen. Because of its simplicity and it can open a text file and display it in a text-only terminal, **cat** is widely used to this day.

- Start the **CyberOps Workstation VM** and open a terminal window.
- 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:

```
analyst@secOps ~$ cat /home/analyst/lab.support.files/logstash-tutorial.log
```

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

Lab - Reading Server Logs



The screenshot shows a VMware Workstation interface with a single virtual machine named 'CyberOps'. The terminal window is open, displaying the output of the command `cat /home/analyst/lab.support.files/logstash-tutorial.log`. The output is a long list of log entries, each starting with a timestamp and a source IP address, followed by a description of the event. The terminal window has a title bar that reads 'Terminal - analyst@secOps -'. The VMware Workstation window has a title bar that reads 'CyberOps - VMware Workstation'.

Question:

What is a drawback of using **cat** with large text files?

The beginning of the file may get lost as **cat** doesn't support page breaking.

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**:

```
analyst@secOps ~$ more /home/analyst/lab.support.files/logstash-tutorial.log
```

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.

Question:

What is the drawback of using **more**?

The beginning of the file may get lost as **cat** doesn't support page breaking.

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:

```
analyst@secOps ~$ less /home/analyst/lab.support.files/logstash-tutorial.log
```

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.

Use the **"q"** key on your keyboard to exit the **less** tool.

- 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.

```
analyst@secOps ~$ tail /home/analyst/lab.support.files/logstash-tutorial.log
218.30.103.62 - - [04/Jan/2015:05:28:43 +0000] "GET /blog/geekery/xvfb-firefox.html
HTTP/1.1" 200 10975 "-" "Sogou web
spider/4.0 (+http://www.sogou.com/docs/help/webmasters.htm#07)"
218.30.103.62 - - [04/Jan/2015:05:29:06 +0000] "GET /blog/geekery/puppet-facts-into-
mcollective.html HTTP/1.1" 200 9872 "-" "Sogou web
spider/4.0 (+http://www.sogou.com/docs/help/webmasters.htm#07)"
198.46.149.143 - - [04/Jan/2015:05:29:13 +0000] "GET /blog/geekery/disabling-battery-
in-ubuntu-
vms.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+semicomplete%2Fmai
n+%28semicomplete.com+-+Jordan+Sissel%29 HTTP/1.1" 200 9316 "-" "Tiny Tiny RSS/1.11
(http://tt-rss.org/)"
198.46.149.143 - - [04/Jan/2015:05:29:13 +0000] "GET /blog/geekery/solving-good-or-
bad-
problems.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+semicomplete%
2Fmain+%28semicomplete.com+-+Jordan+Sissel%29 HTTP/1.1" 200 10756 "-" "Tiny Tiny
RSS/1.11 (http://tt-rss.org/)"
```

```
218.30.103.62 - - [04/Jan/2015:05:29:26 +0000] "GET /blog/geekery/jquery-interface-
puffer.html%20target= HTTP/1.1" 200 202 "-" "Sogou web
spider/4.0 (+http://www.sogou.com/docs/help/webmasters.htm#07)"
218.30.103.62 - - [04/Jan/2015:05:29:48 +0000] "GET /blog/geekery/ec2-reserved-vs-
ondemand.html HTTP/1.1" 200 11834 "-" "Sogou web
spider/4.0 (+http://www.sogou.com/docs/help/webmasters.htm#07)"
66.249.73.135 - - [04/Jan/2015:05:30:06 +0000] "GET /blog/web/firefox-scrolling-
fix.html HTTP/1.1" 200 8956 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X)
AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25
(compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /projects/xdotool/ HTTP/1.1" 200
12292 "http://www.haskell.org/haskellwiki/Xmonad/Frequently_asked_questions"
"Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140205 Firefox/24.0
Iceweasel/24.3.0"
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /reset.css HTTP/1.1" 200 1015
"http://www.semicomplete.com/projects/xdotool/" "Mozilla/5.0 (X11; Linux x86_64;
rv:24.0) Gecko/20140205 Firefox/24.0 Iceweasel/24.3.0"
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /style2.css HTTP/1.1" 200 4877
"http://www.semicomplete.com/projects/xdotool/" "Mozilla/5.0 (X11; Linux x86_64;
rv:24.0) Gecko/20140205 Firefox/24.0 Iceweasel/24.3.0"
```

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:

```
analyst@secOps ~$ sudo tail -f /home/analyst/lab.support.files/logstash-
tutorial.log
```

Question:

What is different in the output of **tail** and **tail -f**? Explain.

Tail -f will not end the command after load the "tail" of file, the terminal are not accept any command. Tail still running, if the append data into the fail, tail will load that new data and display it to the screen.

```

CyberOps - VMware Workstation
File Edit View VM Tabs Help

Library
Type here to search
My Computer
  IAO
  Security Onion
  CyberOps
  Work
    Windows Server 2019
    Windows Server 2019.2
    Windows Server 2012
    Windows 10 Lite
    Windows 10 x64
  OS
    Windows 10 Lite
  Vuln
    m0rbot
    UbuntuServer
    Dops
    Metasploit2-Linux
  Black
    Kali
    CyberOps_old
    EVE-ng
    192.168.0.241

Terminal - analyst@nquangit-
File Edit View Terminal Tabs Help
Usage: tail [OPTION]... [FILE]...
Print the last 10 lines of each FILE to standard output.
With more than one FILE, precede each with a header giving the file name.
With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.
-c, --bytes=[+] output the last NUM bytes; or use -c +NUM to
                    output starting with byte NUM of each file
-f, --follow[=(name|descriptor)]
                    output appended data as the file grows;
                    an absent option argument means 'descriptor'
-F                  same as --follow=name --retry
-n, --lines=[+] output the last NUM lines, instead of the last 10;
                    or use -n +NUM to skip NUM-1 lines at the start
--max-unchanged-stats=N
                    with --follow=name, reopen a FILE which has not
                    changed size after N (default 5) iterations
                    to see if it has been unlinked or renamed
                    (this is the usual case of rotated log files);
                    with inotify, this option is rarely useful
--pid=PID           with -f, terminate after process ID, PID dies
--quiet, --silent   never output headers giving file names
--retry             keep trying to open a file if it is inaccessible
                    with -f, sleep for approximately N seconds
                    (default 1.0) between iterations;
                    with inotify and --pid=P, check process P at
                    least once every N seconds
-v, --verbose       always output headers giving file names
-z, --zero-terminated
                    line delimiter is NUL, not newline
--help             display this help and exit
--version          output version information and exit

NUM may have a multiplier suffix:
b 512, K 1000, M 1024, G 1000*1000, T 1024*1024, and so on for T, P, E, Z, Y, R, Q.
Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

With --follow (-f), tail defaults to following the file descriptor, which
means that even if a tailed file is reopened, tail will continue to track

```

- 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.

```

CyberOps - VMware Workstation
File Edit View VM Tabs Help

Library
Type here to search
My Computer
  IAO
  Security Onion
  CyberOps
  Work
    Windows Server 2019
    Windows Server 2019.2
    Windows Server 2012
    Windows 10 Lite
    Windows 10 x64
  OS
    Windows 10 Lite
  Vuln
    m0rbot
    UbuntuServer
    Dops
    Metasploit2-Linux
  Black
    Kali
    CyberOps_old
    EVE-ng
    192.168.0.241

Terminal - tail -f /home/analyst/lab.support.files/logstash-tutorial.log
File Edit View Terminal Tabs Help
218.30.183.62 - - [04/Jun/2015:05:28:43 +0000] "GET /blog/geekery/xvfb-firefox.html HTTP/1.1" 200 10975 "-" "Sogou web spider/4.0(http://www.sogou.com/docs/help/web/on/docs/help/webmasters.htm#07)"
198.46.149.143 - - [04/Jun/2015:05:29:13 +0000] "GET /blog/geekery/disabl micomplete2Fmain+28semicomplete.com+Jordan+Sissel+29 HTTP/1.1" 200 931
198.46.149.143 - - [04/Jun/2015:05:29:13 +0000] "GET /blog/geekery/solving onplete2Fmain+28semicomplete.com+Jordan+Sissel+29 HTTP/1.1" 200 10756
218.30.183.62 - - [04/Jun/2015:05:29:26 +0000] "GET /blog/geekery/query-i u.com/docs/help/webmasters.htm#07)"
218.30.183.62 - - [04/Jun/2015:05:29:48 +0000] "GET /blog/geekery/ec2-rese ocs/help/webmasters.htm#07)"
66.249.73.135 - - [04/Jun/2015:05:30:06 +0000] "GET /blog/web/firefox-scro ) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safar
86.1.76.62 - - [04/Jun/2015:05:30:37 +0000] "GET /projects/adttool/ HTTP/1
18/5.0 (X11; Linux x86_64; rv:24.0) Gecko/2010205 Firefox/24.0 Iceweasel/
86.1.76.62 - - [04/Jun/2015:05:30:37 +0000] "GET /reset.css HTTP/1.1" 200
0) Gecko/20140205 Firefox/24.0 Iceweasel/24.3.0"
86.1.76.62 - - [04/Jun/2015:05:30:37 +0000] "GET /style2.css HTTP/1.1" 200
4.0) Gecko/20140205 Firefox/24.0 Iceweasel/24.3.0"

Terminal - analyst@nquangit-
File Edit View Terminal Tabs Help
echo "this is a new entry to the monitored log file" >> lab.support.files/log
stash-tutorial.log

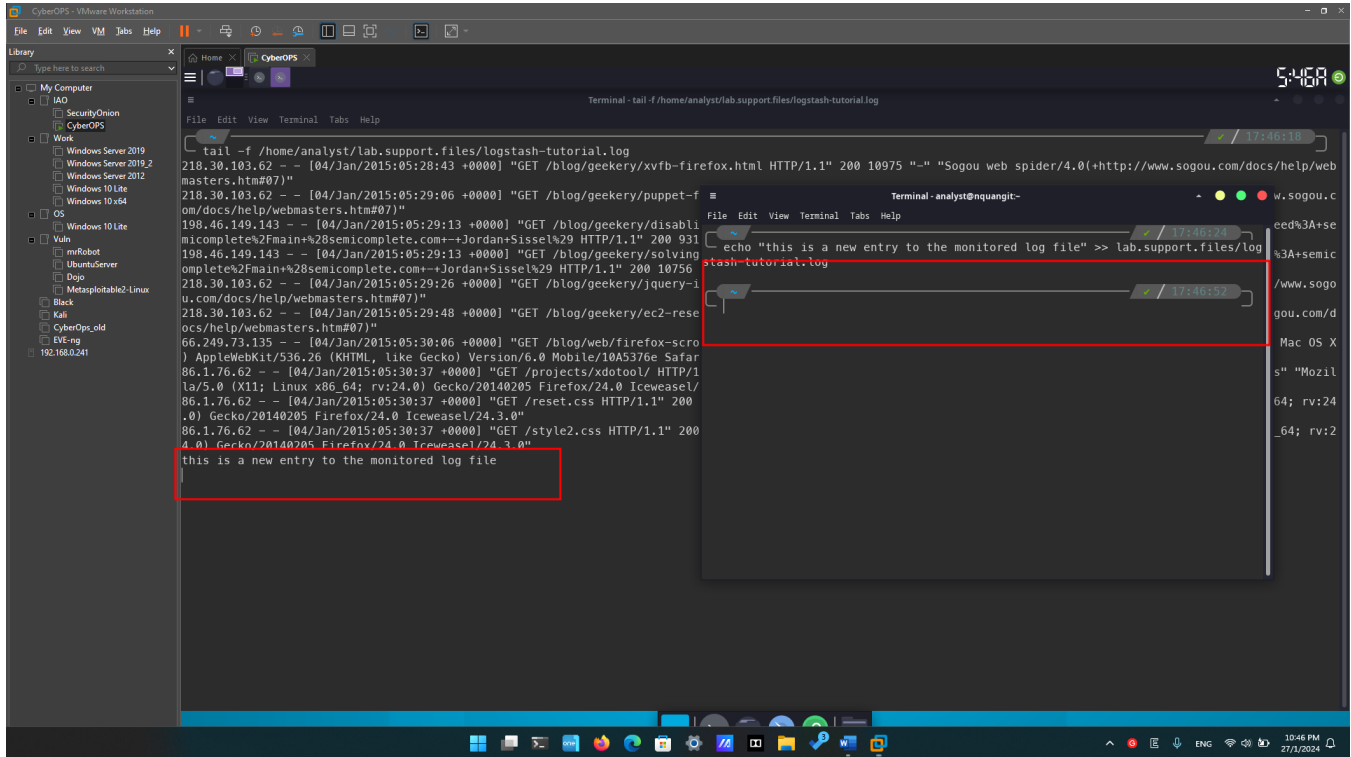
```

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

Lab - Reading Server Logs

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

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.



- Press CTRL + C to stop the execution of **tail -f** and return to the shell prompt.
- Close one of the two terminal windows.

Part 2: Log Files and Syslog

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**.

- 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:
```

```
Feb  7 13:23:15 secOps kernel: [ 5.458959] psmouse serio1: hgpk: ID: 10 00 64
Feb  7 13:23:15 secOps kernel: [ 5.467285] input: ImExPS/2 BYD TouchPad as
/devices/platform/i8042/serio1/input/input6
Feb  7 13:23:15 secOps kernel: [ 5.502469] RAPL PMU: API unit is 2^-32 Joules, 4
fixed counters, 10737418240 ms ovfl timer
Feb  7 13:23:15 secOps kernel: [ 5.502476] RAPL PMU: hw unit of domain pp0-core 2^-
0 Joules
```



```
Feb  7 13:23:15 secOps kernel: [ 5.502478] RAPL PMU: hw unit of domain package 2^-0 Joules
Feb  7 13:23:15 secOps kernel: [ 5.502479] RAPL PMU: hw unit of domain dram 2^-0 Joules
Feb  7 13:23:15 secOps kernel: [ 5.502480] RAPL PMU: hw unit of domain pp1-gpu 2^-0 Joules
Feb  7 13:23:15 secOps kernel: [ 5.672547] ppdev: user-space parallel port driver
Feb  7 13:23:15 secOps kernel: [ 5.709000] pcnet32 0000:00:03.0 enp0s3: renamed from eth0
Feb  7 13:23:16 secOps kernel: [ 6.166738] pcnet32 0000:00:03.0 enp0s3: link up, 100Mbps, full-duplex
Feb  7 13:23:16 secOps kernel: [ 6.706058] random: crng init done
Feb  7 13:23:18 secOps kernel: [ 8.318984] floppy0: no floppy controllers found
Feb  7 13:23:18 secOps kernel: [ 8.319028] work still pending
Feb  7 14:26:35 secOps kernel: [ 3806.118242] hrtimer: interrupt took 4085149 ns
Feb  7 15:02:13 secOps kernel: [ 5943.582952] pcnet32 0000:00:03.0 enp0s3: link down
Feb  7 15:02:19 secOps kernel: [ 5949.556153] pcnet32 0000:00:03.0 enp0s3: link up, 100Mbps, full-duplex
```

Question:

Why did the **cat** command have to be executed as **root**?

The `/var/log/syslog` belongs to root and can only be read by root.

- 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
```

Question:

Can you think of a reason why it is so important to keep the time and date of computers correctly synchronized?

Log systems use log files to record and store events and the date/time they took place. If the system clock is incorrect or not synchronized, it will make the troubleshooting process more difficult.

Part 3: Log Files and Journalctl

Another popular log management system is known as **journal**. Managed by the **journald** daemon, the system is designed to centralize the management of logs regardless of where the messages are originating. 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**.

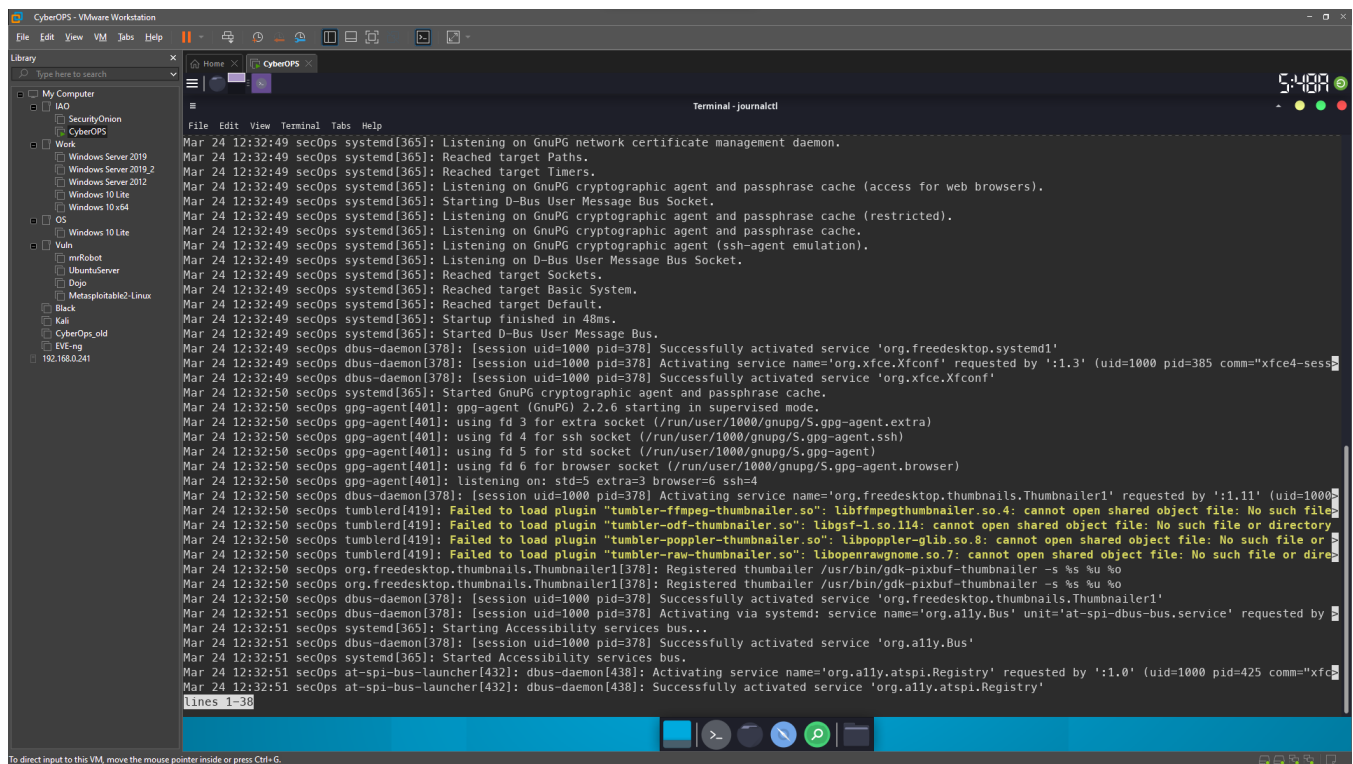
Step 1: Running journalctl with no options.

- 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.

```
analyst@secOps ~$ journalctl
-- Logs begin at Fri 2014-09-26 14:13:12 EDT, end at Tue 2017-02-07 13:23:29 ES
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Paths.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Paths.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Timers.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Timers.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Sockets.
```

Lab - Reading Server Logs

```
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Sockets.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Basic System.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Basic System.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Default.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Default.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Startup finished in 18ms.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Default.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Default.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Basic System.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Basic System.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Paths.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Paths.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Timers.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Timers.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Sockets.
<output omitted>
```



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

b. Use `CTRL+C` to exit the display.

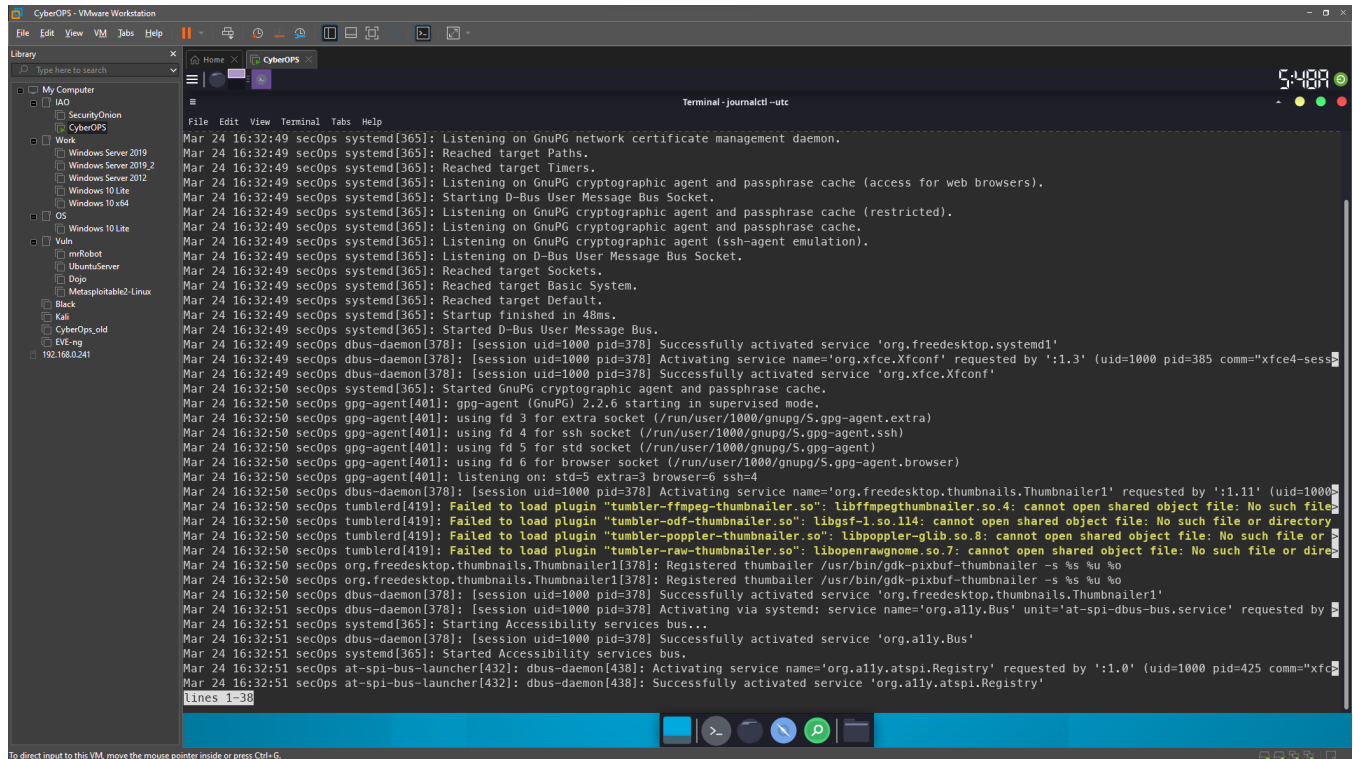
Step 2: Journalctl and a few options.

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

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

```
analyst@secOps ~$ sudo journalctl --utc
```


Lab - Reading Server Logs



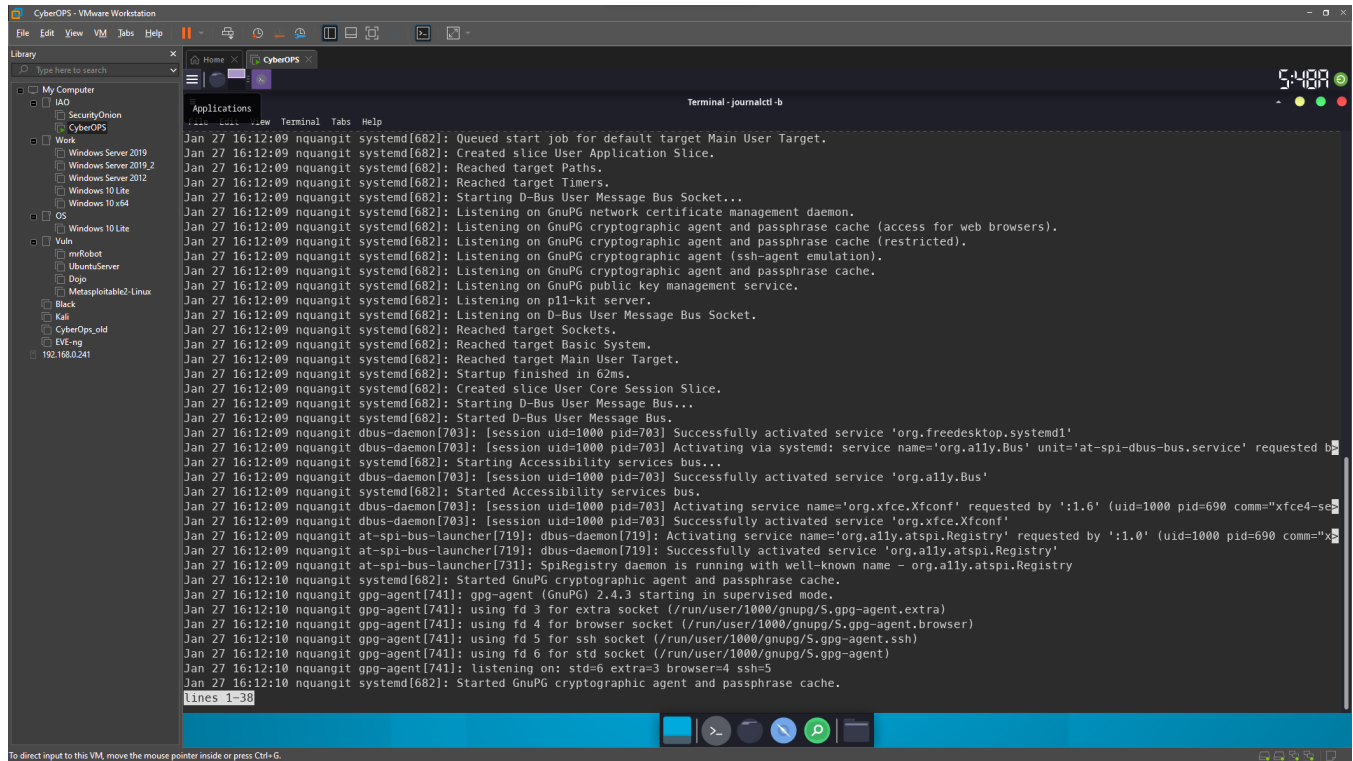
```
Mar 24 16:32:49 secOps systemd[365]: Listening on GnuPG network 'certificate management daemon'.
Mar 24 16:32:49 secOps systemd[365]: Reached target Paths.
Mar 24 16:32:49 secOps systemd[365]: Reached target Timers.
Mar 24 16:32:49 secOps systemd[365]: Listening on GnuPG cryptographic agent and passphrase cache (access for web browsers).
Mar 24 16:32:49 secOps systemd[365]: Starting D-Bus User Message Bus Socket.
Mar 24 16:32:49 secOps systemd[365]: Listening on GnuPG cryptographic agent and passphrase cache (restricted).
Mar 24 16:32:49 secOps systemd[365]: Listening on GnuPG cryptographic agent and passphrase cache.
Mar 24 16:32:49 secOps systemd[365]: Listening on GnuPG cryptographic agent (ssh-agent emulation).
Mar 24 16:32:49 secOps systemd[365]: Listening on D-Bus User Message Bus Socket.
Mar 24 16:32:49 secOps systemd[365]: Reached target Sockets.
Mar 24 16:32:49 secOps systemd[365]: Reached target Basic System.
Mar 24 16:32:49 secOps systemd[365]: Reached target Default.
Mar 24 16:32:49 secOps systemd[365]: Startup finished in 48ms.
Mar 24 16:32:49 secOps systemd[365]: Started D-Bus User Message Bus.
Mar 24 16:32:49 secOps dbus-daemon[378]: [session uid=1000 pid=378] Successfully activated service 'org.freedesktop.systemd1'
Mar 24 16:32:49 secOps dbus-daemon[378]: [session uid=1000 pid=378] Activating service name='org.xfce.Xfconf' requested by ':1.3' (uid=1000 pid=385 comm="xfce4-sess
Mar 24 16:32:50 secOps systemd[365]: Started GnuPG cryptographic agent and passphrase cache.
Mar 24 16:32:50 secOps gpg-agent[401]: gpg-agent (GnuPG) 2.2.6 starting in supervised mode.
Mar 24 16:32:50 secOps gpg-agent[401]: using fd 3 for extra socket (/run/user/1000/gnupg/S.gpg-agent.extra)
Mar 24 16:32:50 secOps gpg-agent[401]: using fd 4 for ssh socket (/run/user/1000/gnupg/S.gpg-agent.ssh)
Mar 24 16:32:50 secOps gpg-agent[401]: using fd 5 for std socket (/run/user/1000/gnupg/S.gpg-agent)
Mar 24 16:32:50 secOps gpg-agent[401]: using fd 6 for browser socket (/run/user/1000/gnupg/S.gpg-agent.browser)
Mar 24 16:32:50 secOps gpg-agent[401]: listening on: std=5 extra=3 browser=b ssh=4
Mar 24 16:32:50 secOps dbus-daemon[378]: [session uid=1000 pid=378] Activating service name='org.freedesktop.thumbnails.Thumbnailer1' requested by ':1.11' (uid=1000
Mar 24 16:32:50 secOps tumblerd[419]: Failed to load plugin "tumbler-ffmpeg-thumbnailer.so": libffmpegthumbnailer.so.4: cannot open shared object file: No such file
Mar 24 16:32:50 secOps tumblerd[419]: Failed to load plugin "tumbler-odf-thumbnailer.so": libgsf-1.so.114: cannot open shared object file: No such file or directory
Mar 24 16:32:50 secOps tumblerd[419]: Failed to load plugin "tumbler-poppler-thumbnailer.so": libpoppler-glib.so.8: cannot open shared object file: No such file or
Mar 24 16:32:50 secOps tumblerd[419]: Failed to load plugin "tumbler-raw-thumbnailer.so": libopenrawgnome.so.7: cannot open shared object file: No such file or dire
Mar 24 16:32:50 secOps org.freedesktop.thumbnails.Thumbnailer1[378]: Registered thumbailer /usr/bin/gdk-pixbuf-thumbnailer -s %s %u %o
Mar 24 16:32:50 secOps dbus-daemon[378]: [session uid=1000 pid=378] Successfully activated service 'org.freedesktop.thumbnails.Thumbnailer1'
Mar 24 16:32:51 secOps dbus-daemon[378]: [session uid=1000 pid=378] Activating via systemd: service name='org.ally.Bus' unit='at-spi-dbus-bus.service' requested by
Mar 24 16:32:51 secOps systemd[365]: Starting Accessibility services bus...
Mar 24 16:32:51 secOps dbus-daemon[378]: [session uid=1000 pid=378] Successfully activated service 'org.ally.Bus'
Mar 24 16:32:51 secOps systemd[365]: Started Accessibility services bus.
Mar 24 16:32:51 secOps at-spi-bus-launcher[432]: dbus-daemon[438]: Activating service name='org.ally.atspi.Registry' requested by ':1.0' (uid=1000 pid=425 comm="xfce
Mar 24 16:32:51 secOps at-spi-bus-launcher[432]: dbus-daemon[438]: Successfully activated service 'org.ally.atspi.Registry'
```

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

```
analyst@secOps ~$ sudo journalctl -b
```

```
Feb 07 08:23:13 secOps systemd-journald[172]: Time spent on flushing to /var is
Feb 07 08:23:13 secOps kernel: Linux version 4.8.12-2-ARCH (builduser@andytrtr)
Feb 07 08:23:13 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 fl
Feb 07 08:23:13 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE re
Feb 07 08:23:13 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX re
Feb 07 08:23:13 secOps kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]
Feb 07 08:23:13 secOps kernel: x86/fpu: Enabled xstate features 0x7, context si
Feb 07 08:23:13 secOps kernel: x86/fpu: Using 'eager' FPU context switches.
Feb 07 08:23:13 secOps kernel: e820: BIOS-provided physical RAM map:
<output omitted>
```

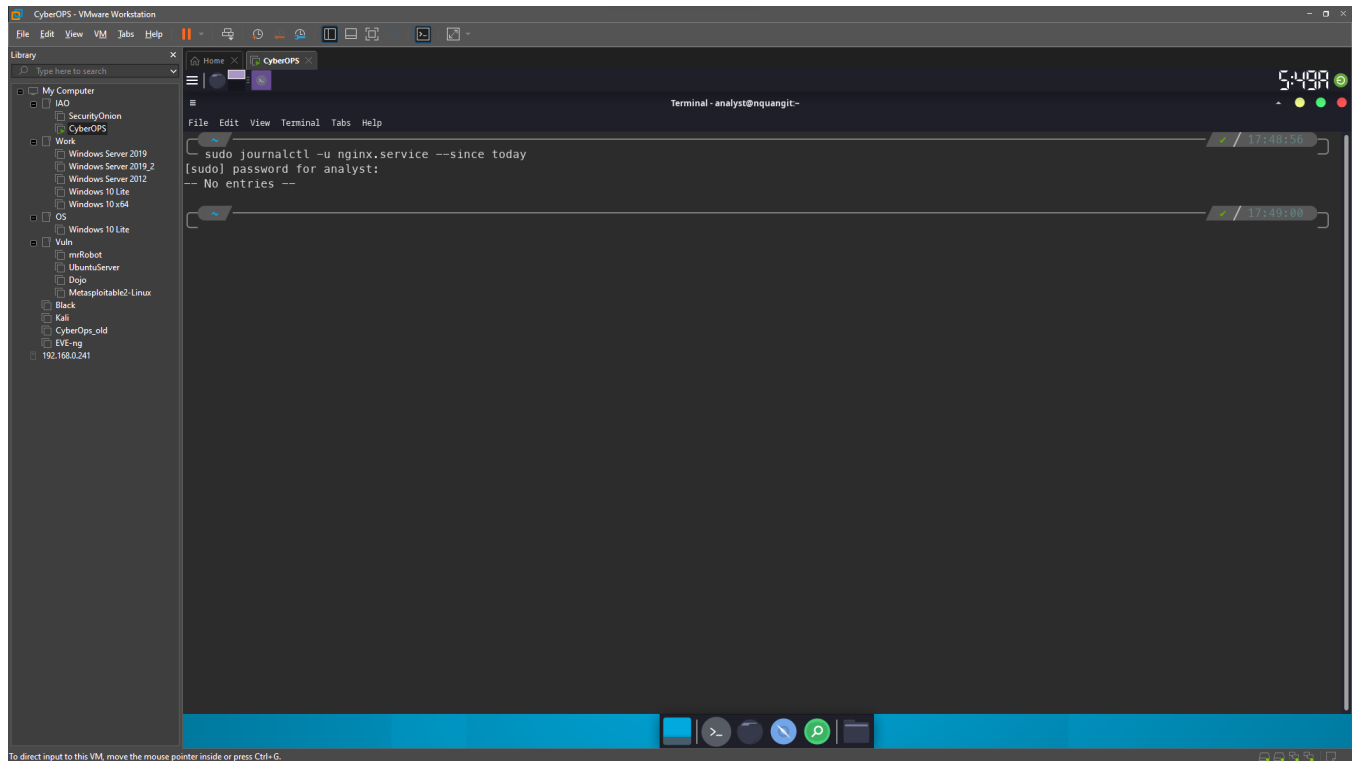
Lab - Reading Server Logs



```
Jan 27 16:12:00 nquangit systemd[682]: Queued start job for default target Main User Target.
Jan 27 16:12:00 nquangit systemd[682]: Created slice User Application Slice.
Jan 27 16:12:00 nquangit systemd[682]: Reached target Paths.
Jan 27 16:12:00 nquangit systemd[682]: Reached target Timers.
Jan 27 16:12:00 nquangit systemd[682]: Starting D-Bus User Message Bus Socket...
Jan 27 16:12:00 nquangit systemd[682]: Listening on GnuPG network certificate management daemon.
Jan 27 16:12:00 nquangit systemd[682]: Listening on GnuPG cryptographic agent and passphrase cache (access for web browsers).
Jan 27 16:12:00 nquangit systemd[682]: Listening on GnuPG cryptographic agent and passphrase cache (restricted).
Jan 27 16:12:00 nquangit systemd[682]: Listening on GnuPG cryptographic agent (ssh-agent emulation).
Jan 27 16:12:00 nquangit systemd[682]: Listening on GnuPG cryptographic agent and passphrase cache.
Jan 27 16:12:00 nquangit systemd[682]: Listening on GnuPG public key management service.
Jan 27 16:12:00 nquangit systemd[682]: Listening on p11-kit server.
Jan 27 16:12:00 nquangit systemd[682]: Listening on D-Bus User Message Bus Socket.
Jan 27 16:12:00 nquangit systemd[682]: Reached target Sockets.
Jan 27 16:12:00 nquangit systemd[682]: Reached target Basic System.
Jan 27 16:12:00 nquangit systemd[682]: Reached target Main User Target.
Jan 27 16:12:00 nquangit systemd[682]: Startup finished in 62ms.
Jan 27 16:12:00 nquangit systemd[682]: Created slice User Core Session Slice.
Jan 27 16:12:00 nquangit systemd[682]: Starting D-Bus User Message Bus...
Jan 27 16:12:00 nquangit systemd[682]: Started D-Bus User Message Bus.
Jan 27 16:12:00 nquangit dbus-daemon[703]: [session uid=1000 pid=703] Successfully activated service 'org.freedesktop.systemd1'
Jan 27 16:12:00 nquangit dbus-daemon[703]: [session uid=1000 pid=703] Activating via systemd: service name='org.ally.Bus' unit='at-spi-dbus-bus.service' requested b
Jan 27 16:12:00 nquangit systemd[682]: Starting Accessibility services bus...
Jan 27 16:12:00 nquangit dbus-daemon[703]: [session uid=1000 pid=703] Successfully activated service 'org.ally.Bus'
Jan 27 16:12:00 nquangit systemd[682]: Started Accessibility services bus.
Jan 27 16:12:00 nquangit dbus-daemon[703]: [session uid=1000 pid=703] Activating service name='org.xfce.Xfconf' requested by ':1.6' (uid=1000 pid=690 comm='"xfce4-se
Jan 27 16:12:00 nquangit dbus-daemon[703]: [session uid=1000 pid=703] Successfully activated service 'org.xfce.Xfconf'
Jan 27 16:12:00 nquangit at-spi-bus-launcher[719]: dbus-daemon[719]: Activating service name='org.ally.atspi.Registry' requested by ':1.0' (uid=1000 pid=690 comm='"x
Jan 27 16:12:00 nquangit at-spi-bus-launcher[731]: SpiRegistry daemon is running with well-known name - org.ally.atspi.Registry
Jan 27 16:12:10 nquangit systemd[682]: Started GnuPG cryptographic agent and passphrase cache.
Jan 27 16:12:10 nquangit gpg-agent[741]: gpg-agent (GnuPG) 2.4.3 starting in supervised mode.
Jan 27 16:12:10 nquangit gpg-agent[741]: using fd 3 for extra socket (/run/user/1000/gnupg/S.gpg-agent.extra)
Jan 27 16:12:10 nquangit gpg-agent[741]: using fd 4 for browser socket (/run/user/1000/gnupg/S.gpg-agent.browser)
Jan 27 16:12:10 nquangit gpg-agent[741]: using fd 5 for ssh socket (/run/user/1000/gnupg/S.gpg-agent.ssh)
Jan 27 16:12:10 nquangit gpg-agent[741]: using fd 6 for std socket (/run/user/1000/gnupg/S.gpg-agent)
Jan 27 16:12:10 nquangit gpg-agent[741]: listening on: std=6 extra=3 browser=4 ssh=5
Jan 27 16:12:10 nquangit systemd[682]: Started GnuPG cryptographic agent and passphrase cache.
```

- 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
```



```
sudo journalctl -u nginx.service --since today
[sudo] password for analyst:
-- No entries --
```

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

Lab - Reading Server Logs

analyst@secOps ~\$ sudo journalctl -k

```
analyst@secOps ~$ sudo journalctl -k
Jan 27 16:09:33 nquangit kernel: Linux version 6.6.9-arch1-1 (linux@archlinux) (gcc (GCC) 13.2.1 20230801, GNU ld (GNU Binutils) 2.41.0) #1 SMP PREEMPT_DYNAMIC Tue.
Jan 27 16:09:33 nquangit kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-linux root=UUID=07c6b457-3f39-4ddf-bfd8-c169e8a877b2 rw quiet
Jan 27 16:09:33 nquangit kernel: BIOS-provided physical RAM map:
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000000f3ff] usable
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x000000000000f400-0x000000000000ffff] reserved
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x000000000000dc00-0x000000000000ffff] reserved
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x0000000000010000-0x0000000000007fedffff] usable
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x0000000000007fee0000-0x0000000000007fefffff] ACPI data
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x0000000000007fef0000-0x0000000000007fefffff] ACPI NVS
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x0000000000007fff0000-0x0000000000007fffff] usable
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x000000000000f0000000-0x000000000000f7ffff] reserved
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x000000000000fec00000-0x000000000000fec0ffff] reserved
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x000000000000fec00000-0x000000000000fec0ffff] reserved
Jan 27 16:09:33 nquangit kernel: BIOS-e820: [mem 0x000000000000ffce0000-0x000000000000ffff] reserved
Jan 27 16:09:33 nquangit kernel: NX (Execute Disable) protection: active
Jan 27 16:09:33 nquangit kernel: APIC: Static calls initialized
Jan 27 16:09:33 nquangit kernel: SMBIOS 2.7 present.
Jan 27 16:09:33 nquangit kernel: DMI: VMware, Inc. VMware Virtual Platform/440BX Desktop Reference Platform, BIOS 6.00 11/12/2020
Jan 27 16:09:33 nquangit kernel: vmware: hypercall mode: 0x02
Jan 27 16:09:33 nquangit kernel: Hypervisor detected: VMware
Jan 27 16:09:33 nquangit kernel: vmware: TSC freq read from hypervisor : 2688.004 MHz
Jan 27 16:09:33 nquangit kernel: vmware: Host bus clock speed read from hypervisor : 66000000 Hz
Jan 27 16:09:33 nquangit kernel: vmware: using clock offset of 6507805067 ns
Jan 27 16:09:33 nquangit kernel: tsc: Detected 2688.004 MHz processor
Jan 27 16:09:33 nquangit kernel: e820: update [mem 0x00000000-0x000000ffff] usable ==> reserved
Jan 27 16:09:33 nquangit kernel: e820: remove [mem 0x00000000-0x000000ffff] usable
Jan 27 16:09:33 nquangit kernel: last_pfn = 0x80000 max_arch_pfn = 0x400000000
Jan 27 16:09:33 nquangit kernel: total RAM covered: 2048M
Jan 27 16:09:33 nquangit kernel: Found optimal setting for mtrr clean up
Jan 27 16:09:33 nquangit kernel: gran_size: 64K chunk_size: 64K num_reg: 1 lose cover RAM: 0G
Jan 27 16:09:33 nquangit kernel: MTRR map: 6 entries (5 fixed + 1 variable; max 21), built from 8 variable MTRRs
Jan 27 16:09:33 nquangit kernel: x86/PAT: Configuration [0-7]: WB WC UC- UC WB WP UC- WT
Jan 27 16:09:33 nquangit kernel: found SMP MP-table at [mem 0x000f6a70-0x000f6a7f]
Jan 27 16:09:33 nquangit kernel: Using GB pages for direct mapping
Jan 27 16:09:33 nquangit kernel: Incomplete global flushes, disabling PCID
Jan 27 16:09:33 nquangit kernel: RAMDISK: [mem 0x36c87000-0x3763afff]
Jan 27 16:09:33 nquangit kernel: ACPI: Early table checksum verification disabled
Jan 27 16:09:33 nquangit kernel: ACPI: RSDP 0x000000000000f6a0 000024 (v02 PLTID )
lines 1-38
```

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

```
analyst@secOps ~$ sudo journalctl -f
Jan 27 17:49:33 nquangit sudo[1932]: pam_unix(sudo:session): session closed for user root
Jan 27 17:49:39 nquangit dbus-daemon[355]: [system] Activating via systemd: service name='org.freedesktop.home1' unit='dbus-org.freedesktop.home1.service' requested by '1.52' (uid=0 pid=1943 comm="sudo journalctl ???")
Jan 27 17:49:39 nquangit dbus-daemon[355]: [system] Activation via systemd failed for unit 'dbus-org.freedesktop.home1.service': Unit dbus-org.freedesktop.home1.serv
ice not found.
Jan 27 17:49:39 nquangit sudo[1943]: analyst : TTY=pts/1 ; PWD=/home/analyst ; USER=root ; COMMAND=/usr/bin/journalctl -f
Jan 27 17:49:39 nquangit sudo[1943]: pam_unix(sudo:session): session opened for user root(uid=0) by analyst(uid=1000)
Jan 27 17:49:40 nquangit sudo[1943]: pam_unix(sudo:session): session closed for user root
Jan 27 17:49:43 nquangit dbus-daemon[355]: [system] Activating via systemd: service name='org.freedesktop.home1' unit='dbus-org.freedesktop.home1.service' requested by '1.53' (uid=0 pid=1948 comm="sudo journalctl -f")
Jan 27 17:49:43 nquangit dbus-daemon[355]: [system] Activation via systemd failed for unit 'dbus-org.freedesktop.home1.service': Unit dbus-org.freedesktop.home1.serv
ice not found.
Jan 27 17:49:43 nquangit sudo[1948]: analyst : TTY=pts/1 ; PWD=/home/analyst ; USER=root ; COMMAND=/usr/bin/journalctl -f
Jan 27 17:49:43 nquangit sudo[1948]: pam_unix(sudo:session): session opened for user root(uid=0) by analyst(uid=1000)
```

Reflection Question

Compare Syslog and Journald. What are the advantages and disadvantages of each?

Syslog serves as a conventional logging solution that employs plaintext files; however, it faces limitations in terms of structure. The data lacks centralization, requiring users to sift through unrelated information to locate relevant details. Syslog lacks a mechanism to segregate messages based on associated applications. Additionally, plaintext files may need rotation to prevent them from becoming excessively large. Journald addresses these issues by replacing plaintext log files with a specialized file format for log messages, facilitating the retrieval of pertinent information.

End of document