17.2.7 Lab - Reading Server Logs

This lab has been updated for use on NETLAB+.
www.netdevgroup.com

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 application generates 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 file and practice using them.

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.

a. Log on to the **Workstation** VM as the **analyst**, using the password **cyberops**.

b. Open a 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 the all contents have been displayed.

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

_____

_____

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.

What is the drawback of using *more*?

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
```

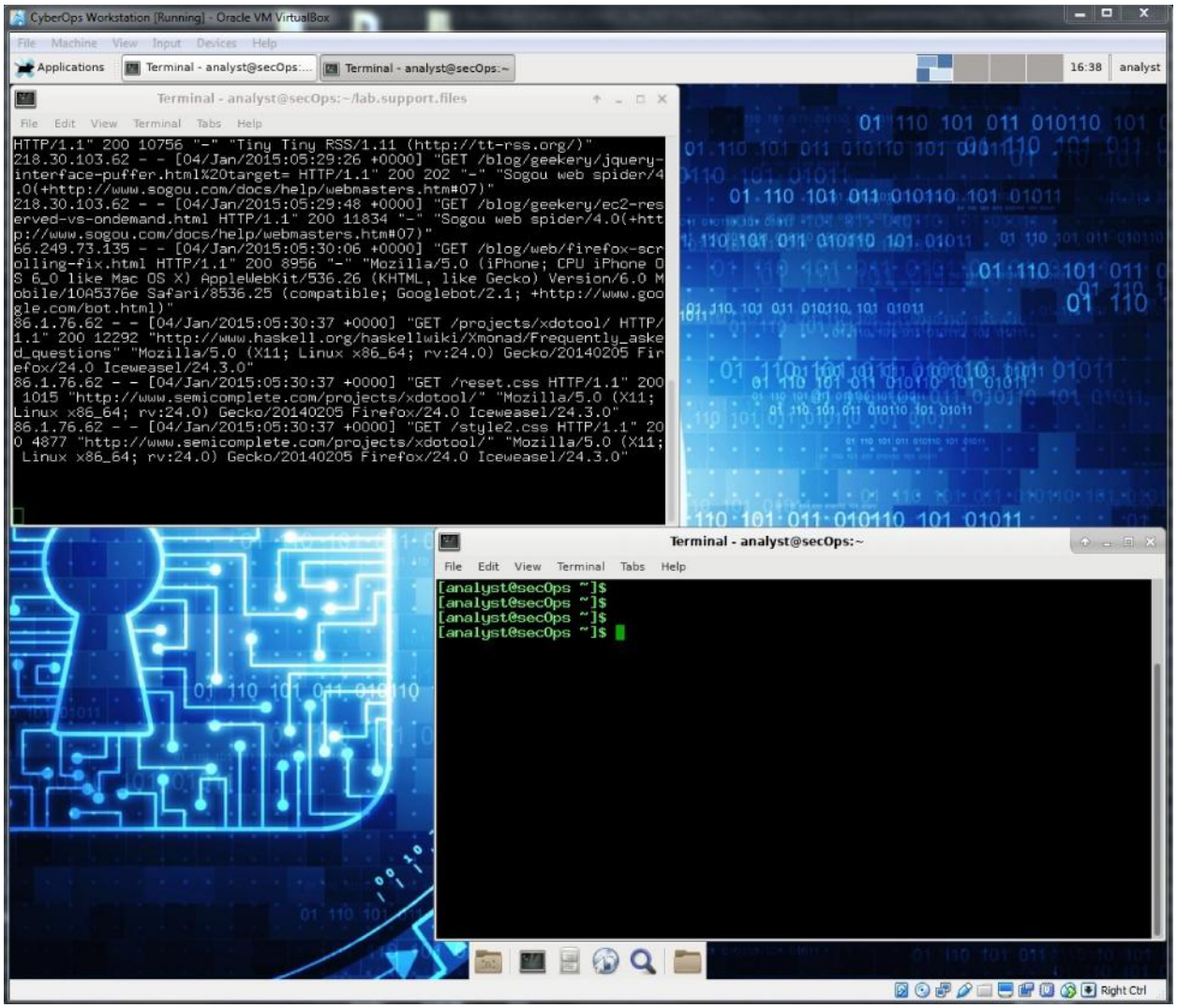
What is different in the output of **tail** and **tail -f**? Explain.

_____

_____

_____

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
```

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: 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**.

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:
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
```

Why did the *cat* command have to be executed as *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
```

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

_____

_____

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.
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 CRTL+C to exit the display.

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

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

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@andyrtr)
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>
```

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
```

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

```
analyst@secOps ~$ sudo journalctl –k
```

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
```

Reflection Question

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

_____

_____

_____

_____