Lab - Locating Log Files

Objectives

In this lab, you will get familiar with locating and manipulating Linux log files.

Part 1: Log File Overview

Part 2: Locating Log Files in Unknown Systems

Part 3: Monitoring Log Files in Real Time

Required Resources

- Security Workstation virtual machine

Instructions

Part 1: Log File Overview

Log files (also spelled logfiles), are files used by computers to log events. Software programs, background processes, services, or transactions between services, including the operating system itself, may generate such events. Log files are dependent on the application that generates them. It is up to the application developer to conform to log file convention. Software documentation should include information on its log files.

Step 1: Web server log file example

Because log files are essentially a way to track specific events, the type of information stored varies depending of the application or services generating the events.

a. Consider the single log entry below. It was generated by Apache, a popular web server.

```
[Wed Mar 22 11:23:12.207022 2017] [core:error] [pid 3548:tid 4682351596] [client 209.165.200.230] File does not exist:
/var/www/apache/htdocs/favicon.ico
```

The single log entry above represents a web event recorded by Apache. A few pieces of information are important in web transactions, including client IP address, time and details of the transaction. The entry above can be broken down into five main parts:

Timestamp: This part records when the event took place. It is very important that the server clock is correctly synchronized as it allows for accurately cross-referencing and tracing back events.

Type: This is the type of event. In this case, it was an error.

PID: This contains information about the process ID used by Apache at the moment.

Client: This records the IP address of the requesting client.

Description: This contains a description of the event.

Based on the log entry above, describe what happened.

Answer Area

Type your answers here.

Show Answer

[analyst@secOps ~]$ **cat /var/log/logstash-tutorial.log**

```
83.149.9.216 - - [04/Jan/2015:05:13:42 +0000] 'GET /presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1' 200 20302

'http://semicomplete.com/presentations/logstash-monitorama-2013/' 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (K

83.149.9.216 - - [04/Jan/2015:05:13:42 +0000] 'GET /presentations/logstash-monitorama-2013/images/kibana-dashboard3.png HTTP/1.1' 200 1

'http://semicomplete.com/presentations/logstash-monitorama-2013/' 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (K

83.149.9.216 - - [04/Jan/2015:05:13:44 +0000] 'GET /presentations/logstash-monitorama-2013/plugin/highlight/highlight.js HTTP/1.1' 200 2

'http://semicomplete.com/presentations/logstash-monitorama-2013/' 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (K
```

Is the output above still considered a web transaction? Explain why the output of the cat command is in a different format than the single entry shown in item (a).

Answer Area

Type your answers here.

Show Answer

Step 2: Operating system log file example

Any software can keep log files, including the operating system itself. Conventionally, Linux uses the /var/log directory to stores various log files, including operating system logs. Modern operating systems are complex pieces of software and therefore use several different files to log events. This section takes a quick look at the /var/log/messages file.

a. Stored under /var/log, the messages file stores various system events. The connection of new USB drive, a network card becoming available, and too many missed root login attempts, are a few examples of events logged to the /var/log/messages file. Use the **more** command to display the contents of the /var/log/messages file. Unlike the **cat** command, **more** allows for a paced navigation through the file. Press **ENTER** to advance line-by-line or **SPACE** to advance an entire page. Press **q** or **CTRL + C** to abort and exit **more**.

   Note: the sudo command is required because the messages file belongs to the root user.

```
[analyst@secOps ~]$ sudo more /var/log/messages
[sudo] password for analyst:
```

```
Mar 20 08:34:38 secOps kernel: [6.149910] random: crng init done

Mar 20 08:34:40 secOps kernel: [8.280667] floppy0: no floppy controllers found

Mar 20 08:34:40 secOps kernel: [8.280724] work still pending

Mar 20 08:35:16 secOps kernel: [   44.414695] hrtimer: interrupt took 5346452 ns

Mar 20 14:28:29 secOps kernel: [21239.566409] pcnet32 0000:00:03.0 enp0s3: link down

Mar 20 14:28:33 secOps kernel: [21243.404646] pcnet32 0000:00:03.0 enp0s3: link up, 100Mbps, full-duplex

Mar 20 14:28:35 secOps kernel: [21245.536961] pcnet32 0000:00:03.0 enp0s3: link down

Mar 20 14:28:43 secOps kernel: [21253.427459] pcnet32 0000:00:03.0 enp0s3: link up, 100Mbps, full-duplex

Mar 20 14:28:53 secOps kernel: [21263.449480] pcnet32 0000:00:03.0 enp0s3: link down

Mar 20 14:28:57 secOps kernel: [21267.500152] pcnet32 0000:00:03.0 enp0s3: link up, 100Mbps, full-duplex

Mar 20 14:29:01 secOps kernel: [21271.551499] pcnet32 0000:00:03.0 enp0s3: link down

Mar 20 14:29:05 secOps kernel: [21275.389707] pcnet32 0000:00:03.0 enp0s3: link up, 100Mbps, full-duplex

Mar 22 06:01:40 secOps kernel: [0.000000] Linux version 4.8.12-2-ARCH (builduser@andyrtr) (gcc version 6.2.1 20160830 (GCC) ) #1 SMP PRI

Mar 22 06:01:40 secOps kernel: [0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'

Mar 22 06:01:40 secOps kernel: [0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'

Mar 22 06:01:40 secOps kernel: [0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'

Mar 22 06:01:40 secOps kernel: [0.000000] x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256

Mar 22 06:01:40 secOps kernel: [0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.

Mar 22 06:01:40 secOps kernel: [0.000000] x86/fpu: Using 'eager' FPU context switches.
```

Notice that the events listed above are very different from the web server events. Because the operating system itself is generating this log, all recorded events are in relation to the OS itself.

b. If necessary, enter Ctrl + C to exit out of the previous command.
c. Log files are very important for troubleshooting. Assume that a user of that specific system reported that all network operations were slow around 4:20 am on May 19.

Can you find evidence of that in the log entries shown above? If so, in what lines? Explain.

Answer Area

Type your answers here.

Show Answer

Part 2: Locating Log Files in Unknown Systems

The Security Workstation VM includes nginx, a lightweight web server. This section will show how to find and display nginx logs using the Security Workstation VM.

Note: nginx was installed on the CyberOps Workstation VM with its default settings. With default settings, its global configuration file is located under /etc/nginx/nginx.conf, its access log file is at /var/log/nginx/access.log, and errors are redirected to the terminal window. However, it is common for a security analyst to work on computers in which the installation details for tool and services are unknown. This section describes the process of locating such files described for nginx but is by no means complete. Nevertheless, it should be a good exercise about locating and displaying log files on unfamiliar systems.

a. When working with new software, the first step is to look at the documentation. It provides important information about the software, including information about its log files. Use the **man** command to display the nginx manual page:

```
  [analyst@secOps ~]$ man nginx

NGINX(8)                      BSD System Manager's Manual                      NGINX(8)

NAME

     nginx — HTTP and reverse proxy server, mail proxy server

SYNOPSIS

     nginx [-?hqTtVv] [-c file] [-g directives] [-p prefix] [-s signal]

DESCRIPTION
```

```
nginx (pronounced “engine x”) is an HTTP and reverse proxy server, as well as a mail proxy

server.  It is known for its high performance, stability, rich feature set, simple configura-

 tion, and low resource consumption.
```

b. Scroll down the page to locate the nginx logging section. The documentation makes it clear that nginx supports logging, with the location of its log files defined at compilation time.

```
[PARTIAL OUTPUT EXTRACTED FROM NGINX MANUAL PAGE]



DEBUGGING LOG
     To enable a debugging log, reconfigure nginx to build with debugging:

           ./configure --with-debug ...

     and then set the debug level of the error_log:

           error_log /path/to/log debug;

     It is also possible to enable the debugging for a particular IP address:

           events {
                   debug_connection 127.0.0.1;
           }
```

c. The manual page also contains information on the files used by nginx. Scroll down further to display the nginx operating files under the Files section:

```
FILES
     %%PID_PATH%%
             Contains the process ID of nginx.  The contents of this file are
             not sensitive, so it can be world-readable.

     %%CONF_PATH%%
             The main configuration file.

     %%ERROR_LOG_PATH%%
             Error log file.
```

The outputs above help you to conclude that nginx supports logging and that it can save to log files. The output also hints at the existence of a configuration file for nginx.

d. Before looking for nginx files, use the **ps** and the **grep** commands to ensure nginx is running in the VM.

Note: Use **man** to learn more about **ps** and **grep** commands.

```
[analyst@secOps ~]$ ps ax | grep nginx
  415 ?        Ss     0:00 nginx: master process /usr/bin/nginx -g pid /run/nginx.pid; error_log stderr;
  416 ?        S      0:00 nginx: worker process
 1207 pts/0    S+     0:00 grep nginx
```

The output above confirms that nginx is running. In addition, the output also displays the parameters used when nginx was started. nginx process ID is being stored in /run/nginx.pid and error messages are being redirected to the terminal.

Note: If nginx is not running, enter the **sudo /usr/sbin/nginx** at the prompt to start the service using the default configuration.

Note: If you need to restart nginx, you can kill the service by using the **sudo pkill nginx** command. To start nginx with the custom configuration from a previous lab, run the following command: **sudo nginx -c custom_server.conf**, and test the server by opening a web browser and going to URL: 127.0.0.1:8080. If you wish to start **nginx** with a default configuration you can start it with the command: **sudo /usr/sbin/nginx**, and open a web browser and go to URL: 127.0.0.1.

Because the location to the log files was not specified, the global nginx configuration file should be checked for the location of the log files.

e. By design, the CyberOps Workstation VM utilizes default locations and definitions as much as possible. Conventionally, the /var/log directory holds various log files for various applications and services while configuration files are stored under the /etc directory. While the nginx manual page did not provide an exact location for its log files, it not only confirmed that nginx supports logging but also hinted at the location of a configuration file. Because the log file locations can often be customized in configuration files, a logical next step is to use the **ls** command to look under /etc and look for a nginx configuration file:

```
[analyst@secOps ~]$ ls /etc/
```

Better viewed in landscape mode.

adjtime	host.conf	mke2fs.conf	rc_maps.cfg
apache-ant	hostname	mkinitcpio.conf	request-key.conf
apparmor.d	hosts	mkinitcpio.d	request-key.d
arch-release	ifplugd	modprobe.d	resolv.conf
avahi	initcpio	modules-load.d	resolvconf.conf
bash.bash_logout	inputrc	motd	rpc
bash.bashrc	iproute2	mtab	rsyslog.conf
binfmt.d	iptables	nanorc	securetty
ca-certificates	issue	netconfig	security
crypttab	java-7-openjdk	netctl	services
dbus-1	java-8-openjdk	netsniff-ng	shadow
default	kernel	nginx	shadow-
depmod.d	krb5.conf	nscd.conf	shells
dhcpcd.conf	ld.so.cache	nsswitch.conf	skel
dhcpcd.duid	ld.so.conf	ntp.conf	ssh
dkms	ld.so.conf.d	openldap	ssl
drirc	libnl	openvswitch	sudoers
elasticsearch	libpaper.d	os-release	sudoers.d
environment	lightdm	pacman.conf	sudoers.pacnew

ethertypes	locale.conf	pacman.conf.pacnew	sysctl.d

f. Notice the nginx folder under /etc in the output above. Using ls again, we find a number of files, including one named nginx.conf.

```
[analyst@secOps ~]$ ls -l /etc/nginx/
total 48
-rw-r--r-- 1 root root 2730 Mar 21 16:02 custom_server.conf
-rw-r--r-- 1 root root 1077 Nov 18 15:14 fastcgi.conf
-rw-r--r-- 1 root root 1007 Nov 18 15:14 fastcgi_params
-rw-r--r-- 1 root root 2837 Nov 18 15:14 koi-utf
-rw-r--r-- 1 root root 2223 Nov 18 15:14 koi-win
-rw-r--r-- 1 root root 2743 Jan  6 15:41 mal_server.conf
-rw-r--r-- 1 root root 3957 Nov 18 15:14 mime.types
-rw-r--r-- 1 root root 3264 Mar 22 13:34 nginx.conf
-rw-r--r-- 1 root root 3261 Oct 19 16:42 nginx.conf.working
-rw-r--r-- 1 root root  636 Nov 18 15:14 scgi_params
-rw-r--r-- 1 root root  664 Nov 18 15:14 uwsgi_params
-rw-r--r-- 1 root root 3610 Nov 18 15:14 win-utf
```

g. Use the **cat** command to list the contents of /etc/nginx/nginx.conf. You can also use **more or less** to view the file and **nano** or **SciTE** to edit it. These tools make it easier to navigate through long text files (only the output of cat is displayed below).

```
[analyst@secOps ~]$ cat /etc/nginx/nginx.conf
#user html;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;
#pid        logs/nginx.pid;

events {
    worker_connections  1024;
}
```

Note: Lines that start with '#' are comments and are ignored by nginx.

h. A quick look at the configuration file reveals that it is an nginx configuration file. Because there is no direct mention to the location of nginx log files, it is very likely that nginx is using default values for it. Following the convention of storing log files under /var/log, use the **ls** command to list its contents:

```
[analyst@secOps ~]$ ls -l /var/log/

total 5708

-rw-r-----  1 root    log              188962 Apr 19 10:35 auth.log

-rw-rw----  1 root    utmp                384 Apr 19 10:05 btmp

-rw-rw----  1 root    utmp               1536 Mar 22 08:50 btmp.1

-rw-r-----  1 root    log              849038 Apr 19 10:05 daemon.log

-rw-r-----  1 root    log                4416 Apr 19 09:45 errors.log

-rw-r-----  1 root    log             1819814 Apr 19 10:05 everything.log

-rw-------  1 root    root              32032 Apr 19 10:05 faillog

drwxr-sr-x+ 4 root    systemd-journal    4096 Mar 20 15:28 journal

-rw-r-----  1 root    log              927701 Apr 19 09:45 kernel.log

-rw-rw-r--  1 root    utmp             292292 Mar 26 11:03 lastlog

drwx--x--x  2 root    lightdm            4096 Apr 19 09:45 lightdm

-rw-r--r--  1 analyst analyst           24464 Apr 19 10:05 logstash-tutorial.log

-rw-r-----  1 root    log             1673153 Apr 19 10:05 messages

drwxr-xr-x  2 root    root               4096 Apr 19 10:28 nginx

-rw-r--r--  1 http    root                989 Apr 19 10:05 nginx-logstash.log

drwxr-xr-x  2 root    root               4096 Jan  5 14:17 old

-rw-r--r--  1 root    root              97655 Apr 17 12:52 pacman.log

drwxr-xr-x  2 snort   snort              4096 Mar 26 11:03 snort

-rw-r-----  1 root    log                 563 Apr 19 09:45 syslog.log

-rw-------  1 root    root              64064 Mar 26 11:03 tallylog

-rw-r-----  1 root    log                 216 Apr 17 13:04 user.log

-rw-rw-r--  1 root    utmp              70272 Apr 19 09:45 wtmp

-rw-r--r--  1 root    root              24756 Apr 19 09:45 Xorg.0.log

-rw-r--r--  1 root    root              25585 Apr 17 14:43 Xorg.0.log.old
```

i. As shown above, the **/var/log** directory has a subdirectory named **nginx.** Use the **ls** command again to list the contents of **/var/log/nginx.**

Note: Because the **/var/log/nginx** belongs to the **http** user, you must execute ls as root by preceding it with the **sudo** command.

```
[analyst@secOps ~]$ sudo ls -l /var/log/nginx

[sudo] password for analyst:

total 16
```

```
-rw-r----- 1 http log   0 May 18 17:53 access.log
-rw-r----- 1 http log 175 May  6 09:42 access.log.1.gz
-rw-r----- 1 http log 593 May  5 16:58 access.log.2.gz
-rw-r----- 1 http log 193 Jul 19  2018 access.log.3.gz
-rw-r----- 1 http log 425 Apr 19  2018 access.log.4.gz
```

These are very likely to be the log files in use by nginx. Move on to the next section to monitor these files and get confirmation that they are indeed nginx log files.

Note: Your output may be different. The .GZ log files above were generated by a log rotation service. Linux systems often implement a service to rotate logs, ensuring that individual log files do not become too large. The log rotate service takes the latest log file, compresses it and saves it under a different name (access.log.1.gz, access.log.2.gz, etc). A new empty main log file is then created and used to store the latest log entries.

Part 3: Monitoring Log Files in Real Time

As seen in the previous sections, log files can be displayed with many text-presentation tools. While **cat**, **more**, **less**, and **nano** can be used to work with log files, they are not suitable for log file real-time monitoring. Developers designed various tools that allow for log file real-time monitoring. Some tools are text-based while others have a graphical interface. This lab focuses on tail, a simple but efficient tool, available in practically every Unix-based system.

Step 1: Using the tail command

The **tail** command displays the end of a text file. By default, tail will display the last ten (10) lines of a text file.

Note: If you do not see any log entries, navigate to 127.0.0.1 in a web browser and refresh the page a few time.

a. Use the **tail** command to display the end of the **/var/log/nginx/access.log.**

```
[analyst@secOps ~]$ sudo tail /var/log/nginx/access.log
[sudo] password for analyst:
127.0.0.1 - - [21/May/2017:15:32:32 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [21/May/2017:15:32:34 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [21/May/2017:15:32:41 -0400] GET / HTTP/1.1' 200 612 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Fire
127.0.0.1 - - [21/May/2017:15:32:41 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
127.0.0.1 - - [21/May/2017:15:32:44 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/May/2017:11:20:27 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
27.0.0.1 - - [22/May/2017:12:49:26 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firefo
127.0.0.1 - - [22/May/2017:12:49:50 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/May/2017:12:49:53 -0400] 'GET / HTTP/1.1' 200 612 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Fir
127.0.0.1 - - [22/May/2017:13:01:55 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
[analyst@secOps ~]$
```

Note: If you do not see any log entries, navigate to 127.0.0.1 in a web browser and refresh the page a few time.

b. Use the **–n** option to specify how many lines from the end of a file, **tail** should display.

```
[analyst@secOps ~]$ sudo tail -n 5  /var/log/nginx/access.log
127.0.0.1 - - [22/May/2017:11:20:27 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
127.0.0.1 - - [22/May/2017:12:49:26 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/May/2017:12:49:50 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/May/2017:12:49:53 -0400] 'GET / HTTP/1.1' 200 612 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Fir
127.0.0.1 - - [22/May/2017:13:01:55 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
[analyst@secOps ~]$
```

c. You can use the **tail** command with the **-f** option to monitorthe nginx access.log in real-time. Short for follow, **-f** tells **tail**to continuously display the end of a given text file. In a terminal window,issue **tail** with the **–f** option:

```
[analyst@secOps log]$ sudo tail -f/var/log/nginx/access.log
[sudo] password for analyst:
127.0.0.1 - - [21/Mar/2017:15:32:32 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [21/Mar/2017:15:32:34 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [21/Mar/2017:15:32:41 -0400] 'GET / HTTP/1.1' 200 612 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Fir
127.0.0.1 - - [21/Mar/2017:15:32:41 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
127.0.0.1 - - [21/Mar/2017:15:32:44 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/Mar/2017:11:20:27 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2
127.0.0.1 - - [22/Mar/2017:12:49:26 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/Mar/2017:12:49:50 -0400] 'GET / HTTP/1.1' 304 0 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firef
127.0.0.1 - - [22/Mar/2017:12:49:53 -0400] 'GET / HTTP/1.1' 200 612 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Fir
```

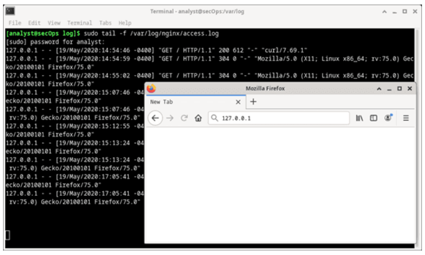
```
127.0.0.1 - - [22/Mar/2017:13:01:55 -0400] 'GET /favicon.ico HTTP/1.1' 404 169 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/2(
```

As before, **tail** displays the last 10 lines of the file. However, notice that **tail** does not exit after displaying the lines; the command prompt is not visible, indicating that **tail** is still running.

Note: Your /var/log/access.log file may be empty due to log rotation. Continue following the lab as an empty /var/log/access.log file will not impact the lab.

d. With **tail** still running on the terminal window, click the web browser icon on the Dock to open a web browser window. Re-size the web browser window in a way that it allows you to see the bottom of the terminal window where **tail** is still running.

Note: In the screenshot below, the Enter key was pressed a few times in the terminal window running **tail.** This is for visualization only as **tail** does not process any input while running with **–f.** The extra empty lines make it easier to detect new entries, as they are displayed at the bottom of the terminal window.



```
[analyst@secOps log]$ sudo tail -f /var/log/nginx/access.log
[sudo] password for analyst:
127.0.0.1 - - [19/May/2020:14:54:46 -0400] "GET / HTTP/1.1" 200 612 "-" "curl/7.69.1"
127.0.0.1 - - [19/May/2020:14:54:59 -0400] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:75.0) Gec
ko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:14:55:02 -0400] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:75.0) Gec
ko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:15:07:46 -04
ecko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:15:07:46 -04
 rv:75.0) Gecko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:15:12:55 -04
ko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:15:13:24 -04
ecko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:15:13:24 -04
 rv:75.0) Gecko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:17:05:41 -04
ecko/20100101 Firefox/75.0"
127.0.0.1 - - [19/May/2020:17:05:41 -04
 rv:75.0) Gecko/20100101 Firefox/75.0"
```

e. In the web browser address bar, enter **127.0.0.1** and press Enter. This is the address of the VM itself, which tells the browser to connect to a web server running on the local computer. A new entry should be recorded in the /var/log/nginx/access.log file. Refresh the webpage to see new entries added to the log.

```
127.0.0.1 - - [23/Mar/2017:09:48:36 -0400] 'GET / HTTP/1.1' 200 612 '-' 'Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Fir(
```

Because **tail** is still running, it should display the new entry at the bottom of the terminal window. Aside from the timestamp, your entry should look like the one above.

Note: Firefox stores pages in cache for future use. If a page is already in cache, force Firefox to ignore the cache and place web requests, reload the page by pressing **.**

f. Because the log file is being updated by nginx, we can state with certainty that /var/log/acess.log is in fact the log file in use by nginx.

g. Enter **Ctrl + C** to end the tail monitoring session.

Step 2: BONUS TOOL: Journalctl

The Security Workstation VM is based on Arch Linux. Categorized as a Linux distribution, Arch Linux is designed to be lightweight, minimalist and simple. As part of this design philosophy, Arch Linux uses systemd as its init system. In Linux, the init process is the first process loaded when the computer boots. Init is directly or indirectly, the parent of all processes running on the system. It is started by the kernel at boot time and continues to run until the computer shuts down. Typically, init has the process ID 1.

An init system is a set of rules and conventions governing the way the user space in a given Linux system is created and made available to the user. Init systems also specify system-wide parameters such as global configuration files, logging structure and service management.

Systemd is a modern init system designed to unify Linux configuration and service behavior across all Linux distributions and has been increasingly adopted by major Linux distributions. Arch Linux relies on systemd for init functionality. The CyberOps Workstation VM also uses systemd.

system-journald (or simply journald) is systemd's event logging service and uses append-only binary files serving as its log files. Notice that journald does not impede the use of other logging systems, such as syslog and rsyslog.

This section provides a brief overview of journalctl, a journald utility used for log viewing and real-time monitoring.

a. In a terminal window in the CyberOps Workstation VM, issue the journalctl command with no options to display all journal log entries (it can be quite long):

```
 [analyst@secOps ~]$ journalctl

Hint: You are currently not seeing messages from other users and the system.

      Users in groups 'adm', 'systemd-journal', 'wheel' can see all messages.

      Pass -q to turn off this notice.

-- Logs begin at Fri 2014-09-26 14:13:12 EDT, end at Fri 2017-03-31 09:54:58 EDT

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.

<some output omitted>
```

The output begins with a line similar to the one below, marking the timestamp where the system started logging. Notice that the timestamps will vary from system to system.

```
-- Logs begin at Fri 2014-09-26 13:22:51 EDT, end at Fri 2017-03-31 10:12:19 EDT. –-
```

journalctl includes a number of functionalities such as page scrolling, color-coded messages and more. Use the keyboard up/down arrow keys to scroll up/down the output, one line at a time. Use the left/right keyboard arrow keys to scroll sideways and display log entries that span beyond the boundaries of the terminal window. The key displays the next line while the space bar displays the next page in the output. Press the **q** key to exit journalctl.

Notice the hint message provided by journalctl:

```
Hint: You are currently not seeing messages from other users and the system.

      Users in groups 'adm', 'systemd-journal', 'wheel' can see all messages.

      Pass -q to turn off this notice.
```

This message reminds you that, because analyst is a regular user and not a member of either the adm, systemd-journal or wheel groups, not all log entries will be displayed by journalctl. It also states that running journalctl with the –q option suppresses the hint message.

How can you run **journalctl** and see all log entries?

Answer Area

Type your answers here.

Show Answer

b. **journalctl** includes options to help in filtering the output. Use the **–b** option to display boot-related log entries:

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

-- Logs begin at Fri 2014-09-26 13:22:51 EDT, end at Fri 2017-03-31 10:18:04 EDT. --

Mar 31 05:54:43 secOps systemd-journald[169]: Time spent on flushing to /var is 849us for 0 entries.

Mar 31 05:54:43 secOps kernel: Linux version 4.8.12-2-ARCH (builduser@andyrtr) (gcc version 6.2.1 20160830 (GCC) ) #1 SMP PREEM

Mar 31 05:54:43 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'

Mar 31 05:54:43 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'

Mar 31 05:54:43 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'

Mar 31 05:54:43 secOps kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256

Mar 31 05:54:43 secOps kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.

Mar 31 05:54:43 secOps kernel: x86/fpu: Using 'eager' FPU context switches.

Mar 31 05:54:43 secOps kernel: e820: BIOS-provided physical RAM map:

Mar 31 05:54:43 secOps kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable

Mar 31 05:54:43 secOps kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved

Mar 31 05:54:43 secOps kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved

Mar 31 05:54:43 secOps kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007ffeffff] usable

<some output omitted>
```

c. To see entries related to the last boot, add the **-1** to the command above. To see entries related to the two last boots, add the **-2** option.

```
[analyst@secOps ~]$ sudo journalctl –b -2

-- Logs begin at Fri 2014-09-26 13:22:51 EDT, end at Fri 2017-03-31 10:21:03 EDT. --

Mar 22 09:35:11 secOps systemd-journald[181]: Time spent on flushing to /var is 4.204ms for 0 entries.

Mar 22 09:35:11 secOps kernel: Linux version 4.8.12-2-ARCH (builduser@andyrtr) (gcc version 6.2.1 20160830 (GCC) ) #1 SMP PREEM

Mar 22 09:35:11 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'

Mar 22 09:35:11 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'

Mar 22 09:35:11 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'

Mar 22 09:35:11 secOps kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256

Mar 22 09:35:11 secOps kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.

Mar 22 09:35:11 secOps kernel: x86/fpu: Using 'eager' FPU context switches.

Mar 22 09:35:11 secOps kernel: e820: BIOS-provided physical RAM map:

Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable

Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved

Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved

Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007ffeffff] usable

Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x000000007fff0000-0x000000007fffffff] ACPI data

Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved
Mar 22 09:35:11 secOps kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved
<some output omitted>
```

d. Use the **--list-boots** option to list previous boots:

```
[analyst@secOps ~]$ sudo journalctl —-list-boots
-144 fbef03a1b59c40429f3e083613ab775a Fri 2014-09-26 13:22:51 EDT—Fri 2014-09-26 14:05:00 EDT
-143 69ebae646d6b41f0b3de9401cb3aa591 Fri 2014-09-26 14:05:07 EDT—Fri 2014-09-26 20:35:29 EDT
-142 73a305f65dea41e787b164411dfc6750 Fri 2014-09-26 20:35:34 EDT—Fri 2014-09-26 20:52:22 EDT
-141 48a113d5d2f44979a849c9c0d9ecdfa2 Fri 2014-09-26 20:52:33 EDT—Fri 2014-09-26 21:08:35 EDT
-140 002af74c3fc44008a882384f546c438d Fri 2014-09-26 21:08:45 EDT—Fri 2014-09-26 21:16:39 EDT
-139 f3ca1d06495c4e26b367e6867f03374c Fri 2014-09-26 21:16:47 EDT—Fri 2014-09-26 21:50:19 EDT
-138 bd232f288e544a79aa3bc444e02185a8 Fri 2014-09-26 21:50:28 EDT—Fri 2014-09-26 22:33:13 EDT
-137 2097c11f249c431aa8ad8da31a5b26d1 Fri 2014-09-26 22:40:39 EDT—Fri 2014-09-26 23:55:46 EDT
-136 b24d5e718a724b18b352e9b2daed3db6 Sat 2014-09-27 10:57:32 EDT—Sat 2014-09-27 14:26:43 EDT
-135 5a189fc68352484a8b40cd719ff7dd41 Sat 2014-09-27 19:44:23 EDT—Sat 2014-09-27 22:50:24 EDT
-134 d0be08c1f26642a1a20bb70bfc7b722c Mon 2014-09-29 09:17:14 EDT—Mon 2014-09-29 12:12:10 EDT
-133 b00b0d4c07464071b0d3cac4eb79dda3 Mon 2014-09-29 12:39:12 EDT—Mon 2014-09-29 13:24:38 EDT
<some output omitted>
```

e. Use the **--since** **“”** to specify the time range of which log entries should be displayed. The two commands below display all log entries generated in the last two hours and in the last day, respectively:

```
[analyst@secOps ~]$ sudo journalctl —-since '2 hours ago'
-- Logs begin at Fri 2014-09-26 13:22:51 EDT, end at Fri 2017-03-31 10:28:29 EDT. --
Mar 31 09:54:45 secOps kernel: 00:00:00.008577 main     5.1.10 r112026 started. Verbose level = 0
Mar 31 09:54:45 secOps systemd[1]: Time has been changed
Mar 31 09:54:45 secOps systemd[1]: Started Rotate log files.
Mar 31 09:54:45 secOps ovsdb-server[263]: 2017-03-31T13:54:45Z|00001|ovsdb_server|INFO|ovsdb-server (Open vSwitch) 2.6.1
Mar 31 09:54:45 secOps ovsdb-server[263]: ovs|00001|ovsdb_server|INFO|ovsdb-server (Open vSwitch) 2.6.1
Mar 31 09:54:45 secOps kernel: openvswitch: Open vSwitch switching datapath
Mar 31 09:54:45 secOps systemd[1]: Started Open vSwitch Daemon.
Mar 31 09:54:45 secOps dhcpcd[279]: enp0s3: soliciting an IPv6 router
Mar 31 09:54:45 secOps ovs-vswitchd[319]: 2017-03-31T13:54:45Z|00001|ovs_numa|INFO|Discovered 1 CPU cores on NUMA node 0
Mar 31 09:54:45 secOps ovs-vswitchd[319]: 2017-03-31T13:54:45Z|00002|ovs_numa|INFO|Discovered 1 NUMA nodes and 1 CPU cores
Mar 31 09:54:45 secOps ovs-vswitchd[319]: ovs|00001|ovs_numa|INFO|Discovered 1 CPU cores on NUMA node 0
Mar 31 09:54:45 secOps ovs-vswitchd[319]: ovs|00002|ovs_numa|INFO|Discovered 1 NUMA nodes and 1 CPU cores
Mar 31 09:54:45 secOps ovs-vswitchd[319]: 2017-03-31T13:54:45Z|00003|reconnect|INFO|unix:/run/openvswitch/db.sock: connecting..
Mar 31 09:54:45 secOps ovs-vswitchd[319]: 2017-03-31T13:54:45Z|00004|reconnect|INFO|unix:/run/openvswitch/db.sock: connected
Mar 31 09:54:45 secOps ovs-vswitchd[319]: ovs|00003|reconnect|INFO|unix:/run/openvswitch/db.sock: connecting...
Mar 31 09:54:45 secOps ovs-vswitchd[319]: ovs|00004|reconnect|INFO|unix:/run/openvswitch/db.sock: connected
Mar 31 09:54:45 secOps ovs-vswitchd[319]: 2017-03-31T13:54:45Z|00005|ovsdb_idl|WARN|Interface table in Open_vSwitch database la
Mar 31 09:54:45 secOps ovs-vswitchd[319]: 2017-03-31T13:54:45Z|00006|ovsdb_idl|WARN|Mirror table in Open_vSwitch database lacks
<some output omitted>


[analyst@secOps ~]$ sudo journalctl —-since '1 day ago'
-- Logs begin at Fri 2014-09-26 13:22:51 EDT, end at Fri 2017-03-31 10:26:48 EDT. --
Mar 30 05:54:43 secOps systemd-journald[169]: Time spent on flushing to /var is 849us for 0 entries.
Mar 30 05:54:43 secOps kernel: Linux version 4.8.12-2-ARCH (builduser@andyrtr) (gcc version 6.2.1 20160830 (GCC) ) #1 SMP PREEM
Mar 30 05:54:43 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
Mar 30 05:54:43 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
Mar 30 05:54:43 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
Mar 30 05:54:43 secOps kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
Mar 31 05:54:43 secOps kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.
Mar 30 05:54:43 secOps kernel: x86/fpu: Using 'eager' FPU context switches.
Mar 30 05:54:43 secOps kernel: e820: BIOS-provided physical RAM map:
Mar 30 05:54:43 secOps kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
Mar 30 05:54:43 secOps kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
Mar 30 05:54:43 secOps kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
```

```
<some output omitted>
```

f. journalctl also allows for displaying log entries related to a specific service with the –u option. The command below displays logs entries related to nginx:

```
[analyst@secOps ~]$ sudo journalctl –u nginx.service
-- Logs begin at Fri 2014-09-26 13:22:51 EDT, end at Fri 2017-03-31 10:30:39 EDT. --
Oct 19 16:47:57 secOps systemd[1]: Starting A high performance web server and a reverse proxy server...
Oct 19 16:47:57 secOps nginx[21058]: 2016/10/19 16:47:57 [warn] 21058#21058: conflicting server name 'localhost' on 0.0.0.0:80,
Oct 19 16:47:57 secOps systemd[1]: nginx.service: PID file /run/nginx.pid not readable (yet?) after start: No such file or dire
Oct 19 16:47:57 secOps systemd[1]: Started A high performance web server and a reverse proxy server.
Oct 19 17:40:09 secOps nginx[21058]: 2016/10/19 17:40:09 [error] 21060#21060: *1 open() '/usr/share/nginx/html/favicon.ico' fai
Oct 19 17:40:09 secOps nginx[21058]: 2016/10/19 17:40:09 [error] 21060#21060: *1 open() '/usr/share/nginx/html/favicon.ico' fai
Oct 19 17:41:21 secOps nginx[21058]: 2016/10/19 17:41:21 [error] 21060#21060: *2 open() '/usr/share/nginx/html/favicon.ico' fai
Oct 19 17:41:21 secOps nginx[21058]: 2016/10/19 17:41:21 [error] 21060#21060: *2 open() '/usr/share/nginx/html/favicon.ico' fai
Oct 19 18:36:33 secOps systemd[1]: Stopping A high performance web server and a reverse proxy server...
Oct 19 18:36:33 secOps systemd[1]: Stopped A high performance web server and a reverse proxy server.
-- Reboot --
Oct 19 18:36:49 secOps systemd[1]: Starting A high performance web server and a reverse proxy server...
Oct 19 18:36:49 secOps nginx[399]: 2016/10/19 18:36:49 [warn] 399#399: conflicting server name 'localhost' on 0.0.0.0:80, ignor
Oct 19 18:36:49 secOps systemd[1]: nginx.service: PID file /run/nginx.pid not readable (yet?) after start: No such file or dire
Oct 19 18:36:49 secOps systemd[1]: Started A high performance web server and a reverse proxy server.
<some output omitted>
```

Note: As part of systemd, services are described as units. Most service installation packages create units and enable units during the installation process.

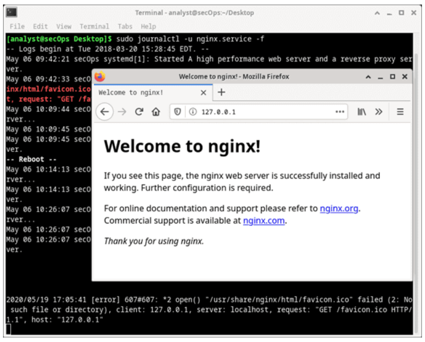
g. Similar to **tail –f, journalctl** also supports real-time monitoring. Use the –f option to instruct journalctl to follow a specific log. **Press Ctrl + C** to exit.

```
[analyst@secOps ~]$ sudo journalctl -f
[sudo] password for analyst:
-- Logs begin at Fri 2014-09-26 13:22:51 EDT. --
Mar 31 10:34:15 secOps filebeat[222]: 2017/03/31 14:34:15.077058 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:34:40 secOps sudo[821]: pam_unix(sudo:session): session closed for user root
Mar 31 10:34:45 secOps filebeat[222]: 2017/03/31 14:34:45.076057 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:35:15 secOps filebeat[222]: 2017/03/31 14:35:15.076118 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:35:45 secOps filebeat[222]: 2017/03/31 14:35:45.076924 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:36:15 secOps filebeat[222]: 2017/03/31 14:36:15.076060 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:36:45 secOps filebeat[222]: 2017/03/31 14:36:45.076122 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:37:15 secOps filebeat[222]: 2017/03/31 14:37:15.076801 logp.go:232: INFO No non-zero metrics in the last 30s
Mar 31 10:37:30 secOps sudo[842]:  analyst : TTY=pts/0 ; PWD=/home/analyst ; USER=root ; COMMAND=/usr/bin/journalctl -f
Mar 31 10:37:31 secOps sudo[842]: pam_unix(sudo:session): session opened for user root by (uid=0)
<some output omitted>
```

h. journalctl also supports mixing options to achieve the desired filter set. The command below monitors nginx system events in real time.

```
[analyst@secOps ~]$ sudo journalctl -u nginx.service -f
-- Logs begin at Fri 2014-09-26 13:22:51 EDT. --
Mar 23 10:08:41 secOps systemd[1]: Stopping A high performance web server and a reverse proxy server...
Mar 23 10:08:41 secOps systemd[1]: Stopped A high performance web server and a reverse proxy server.
-- Reboot --
Mar 29 11:28:06 secOps systemd[1]: Starting A high performance web server and a reverse proxy server...
Mar 29 11:28:06 secOps systemd[1]: nginx.service: PID file /run/nginx.pid not readable (yet?) after start: No such file or directo
Mar 29 11:28:06 secOps systemd[1]: Started A high performance web server and a reverse proxy server.
Mar 29 11:31:45 secOps systemd[1]: Stopping A high performance web server and a reverse proxy server...
Mar 29 11:31:45 secOps systemd[1]: Stopped A high performance web server and a reverse proxy server.
-- Reboot --
Mar 31 09:54:51 secOps systemd[1]: Starting A high performance web server and a reverse proxy server...
Mar 31 09:54:51 secOps systemd[1]: nginx.service: PID file /run/nginx.pid not readable (yet?) after start: No such file or directo
Mar 31 09:54:51 secOps systemd[1]: Started A high performance web server and a reverse proxy server.
```

i. Keep the command above running, open a new web browser window and type 127.0.0.1 (default configuration) or 127.0.0.1:8080 (custom_server.conf) in the address bar. journalctl should display an error related to a missing favicon.ico file in real-time. Use Ctrl+C to quit journalctl.



Terminal - analyst@secOps:~/Desktop
File Edit View Terminal Tabs Help
Welcome to nginx! - Mozilla Firefox
Welcome to nginx!
127.0.0.1

```
[analyst@secOps Desktop]$ sudo journalctl -u nginx.service -f
-- Logs begin at Tue 2018-03-20 15:28:45 EDT. --
May 06 09:42:21 secOps systemd[1]: Started A high performance web server and a reverse proxy ser
ver.
May 06 09:42:33 secO
inx/html/favicon.ico
t, request: "GET /fa
May 06 10:09:44 secO
rver...
May 06 10:09:45 secO
May 06 10:09:45 secO
ver.
-- Reboot --
May 06 10:14:13 secO
rver...
May 06 10:14:13 secO
ver.
May 06 10:26:07 secO
rver...
May 06 10:26:07 secO
May 06 10:26:07 secO
ver.


2020/05/19 17:05:41 [error] 607#607: *2 open() "/usr/share/nginx/html/favicon.ico" failed (2: No
 such file or directory), client: 127.0.0.1, server: localhost, request: "GET /favicon.ico HTTP/
1.1", host: "127.0.0.1"
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Reflection

Log files are extremely important for troubleshooting.

Log file location follows convention but ultimately, it is a choice of the developer.

More often than not, log file information (location, file names, etc.) is included in the documentation. If the documentation does not provide useful information on log files, a combination of web research, and system investigation should be used.

Clocks should always be synchronized to ensure all systems have the correct time. If clocks are not correctly set, it is very difficult to trace back events.

It is important to understand when specific events took place. In addition to that, events from different sources are often analyzed at the same time.

Show All Answers Clear All Responses