Log Management: Monitoring and Making Sense of Logs

License

Contents

Preparation

These tasks can be completed on the LinuxZ IMS image, use the *Launch VM* desktop icon to access the VMs as they are required.

This lab could be completed on most Linux systems with these tools installed: tree, logger, rsyslog, accton. However, if you are not using an openSUSE distro of Linux, the config files may vary slightly.

Log basics

Open a terminal console (such as Konsole from KDEMenu → System → Terminal → Konsole).

Start by having a look at the logs that are available on a Unix/Linux system. These are typically stored in /var/log.

List the various log files by running:

```
tree /var/log
```

> (If you do not have the command "tree" installed, you can use "ls -R /var/log" instead.)

Have a look through the various files present, and try to identify the purpose of as many as possible. You may wish to open these using "less /var/log/"... followed by a filename.

> Hint: try typing "less /var/log/" then without pressing enter, press TAB twice, this will list all the files in /var/log that you could complete the command argument with.

Depending on the version of Unix or distribution of Linux you are using, and the system's configuration, there will be slightly different log files present. Most Unix systems will have a /var/log/messages file (or /var/log/syslog), which is the main location for various logs received by Syslog (and newer Syslog replacements such as Rsyslog). Syslog-like loggers are one of the most common log mechanisms deployed.

View the contents of your messages log file:

```
sudo less /var/log/messages
```

(press ‘Q’ to exit, when you are done)

Depending on how long your system has been running for (and, as will you will soon see, how long ago the logs were rotated), this log file may be very long, with many various kinds of events recorded. Obviously not all of these logs describe security events; however, these logs can be very useful when troubleshooting many kinds of system behaviour, including in the case of investigating a security breach.

Using your own judgement, attempt to find security relevant details within your messages log file.

As an example, my own /var/log/messages file contains the lines:

```
Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: Setting up
rules from /etc/sysconfig/SuSEfirewall2 ...

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: using default
zone 'ext' for interface vmnet1

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: using default
zone 'ext' for interface vmnet8

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: Firewall
rules successfully set

Jan 21 17:16:26 linux-leedsmet pppd[32076]: Script /etc/
ppp/ip-down finished (pid 342), status = 0x0

Jan 21 17:16:26 linux-leedsmet pppd[32076]: Exit.
```

This activity was triggered when I disabled the VPN I was using, which triggered a reload of the firewall rules. On an openSUSE system, configuration for the firewall is stored in /etc/sysconfig/SuSEfirewall2.

As is typical, each Syslog entry starts with a timestamp: for example, “Jan 21 17:16:26”. This is followed by the name of the computer, in my case “linux-leedsmet”. The next part is the name of the service that sent the log event. In the above example this includes logs sent by “SuSEfirewall2”, and “pppd”. In the case of pppd, a process id (pid) is also included. Following this is the actual log event (message) that was sent to Syslog.

Watching for file changes

Since keeping an eye on log files is a common security and sys-admin task, it is worth exploring a few methods.

Tail is a command that prints the last few lines of a text file:

```
sudo tail /var/log/messages
```

A simple method of keeping an eye on a file is to follow the end of a text file.

Run:

```
sudo tail -f /var/log/messages
```

This will block waiting for input, and when new lines are added, they will be displayed.

Similarly, the less program can "follow" the end of a file:

Run:

```
sudo less /var/log/messages
```

Then press Shift+F.

(Press Ctrl-C, Q to exit when you are done.)

Another useful technique is to watch for changes in the output of a command using "watch". Run:

```
watch -d ls -la /var/log/messages
```

This will continuously display the output of the ls command, and highlight any changes, such as, for example, if the file size changes because new entries are logged.

Note that openSUSE is typically configured to also log most log events to virtual terminal 10. View the virtual terminal:

(Read the whole sentence before following the instructions:) Press Ctrl-Alt-F10 to view the log via the terminal, and to return to the desktop environment press Ctrl-Alt-F7.

Writing to Syslog from the commandline

It is important to realise that most local programs (that is, programs running on your computer) can send messages to the Syslog daemon to log.

From the shell prompt run:

```
logger Hello, world!
```

Now, have another look at the end of your messages file. You will find your message, as reported by your username.

Any program can decide how to name its Syslog entries.

```
logger -t some-program Hello, World!
```

(where some-program is replaced something including your name: for example, “logger -t cliffes-program Hello, World!”)

As this implies, you can not necessarily trust all log entries, since an attacker on your system could craft their own log entries, to make them look like they are coming from other programs on your system.

Take a screenshot showing the end of your /var/log/messages file, as evidence that you have completed this part of the task. It should include a Syslog message evidently sent by a program named after you.

Label it or save it as “Logs-1”.

Understanding Syslog

Not all Syslog messages end up in /var/log/messages.

Run:

```
logger -t NetworkManager Network doing something
interesting!
```

On an openSUSE system, this message will not typically end up in /var/log/messages, since NetworkManager has its own log file at /var/log/NetworkManager.

You will find your new message here:

```
sudo tail /var/log/NetworkManager
```

Also, emergencies and warnings are treated differently.

Each Syslog event has a priority, which is made up of a facility (auth, user, etc), and a level (alert, crit, etc). See "man logger" for a list of facilities and levels.

Run this to generate an emergency event:

```
logger -p user.emerg Oh no!
```

This message will typically be sent to *every* terminal, and will generally even generate a popup notification.

While this kind of event will also be logged to /var/log/warn:

```
logger -p user.warn Warning: something is not right...
```

Writing Syslog configuration rules

The behaviour is configured in /etc/rsyslog.conf (or /etc/syslog, depending on the version of Syslog you are using).

Open the Syslog configuration file for editing:

```
sudo vi /etc/rsyslog.conf
```

(Remember: editing using vi involves pressing "i" to insert/edit text, *Esc*, then ":wq" to write changes and quit.)

Read through the configuration file, to understand how the behaviour is configured. Try to understand how the logs are sent to virtual terminal 10 (as you previously accessed via Ctrl-Alt-F10).

Add a rule that sends all messages from the program "mymonitor" to /var/log/mymonitor, by adding these lines:

```
if ($programname == 'mymonitor') \
then    /var/log/mymonitor
```

Restart Syslog so that it re-reads its configuration:

```
sudo /sbin/service syslog restart
```

Write to Syslog to test your new rule:

```
logger -t mymonitor Hello, World!
```

And check that it has written to your new config file:

```
sudo tail /var/log/mymonitor
```

Now, use what you have learned to add a rule so that any sudo command generates a message to all users (which would typically be sent on terminals and as a popup notification).

Hint: look at how the existing emergency messages rule works, and combine that method of output with the rule above, modified to be triggered by sudo. Remember to reload Syslog.

Take a screenshot showing an alert generated from the use of sudo, as evidence that you have completed this part of the task. The screenshot should include a popup notification and terminal output.

Label it or save it as "Logs-2".

After saving your evidence, remove the rule you added above, before continuing.

Network logging

Note that it is not uncommon for a Syslog log file (such as /var/log/messages) to contain entries originating from other systems on the network. There are many benefits of logging to a central server: if an attacker compromises one of the systems (so long as the computer that has been broken into is not the log server!), they cannot delete or alter existing logs; also, it simplifies log management and monitoring if the various logs can be accessed from a central location. Of course, consequently the security of the log server becomes increasingly important.

You will now configure your local host OS to accept Syslog events from other computers.

Note: you can complete this task by working with a classmate, by logging into two computers with the LinuxZ image, or if you prefer you can use a Linux VM as the remote computer.

On openSUSE, Syslog network logging is configured in /etc/rsyslog.d/remote.conf. If you are using another version of Linux, the rules discussed in this section can be added to the Syslog config file, such as /etc/syslog.

On your local system (the soon-to-be log server), edit the configuration file:

```
sudo vi /etc/rsyslog.d/remote.conf
```

(Remember: editing using vi involves pressing "i" to insert/edit text)

Traditionally, Syslog uses UDP, but we will use TCP (as has become common), since it is more reliable. Uncomment (remove the #) from these two lines:

```
$ModLoad imtcp.so

$InputTCPServerRun <port>
```

(Many Unix config files use '#' to denote that a line is a comment that is not to be processed, so by removing those from the beginning of the line you are putting them into effect.)

And change the *<port>* to *10514*.

Save your changes, and exit vi (press *Esc*, then ":wq" to write and quit).

Restart Syslog, for your changes to take effect:

```
sudo /sbin/service syslog restart
```

If you like, you can port scan your own machine, to confirm that the port is now open:

```
nmap localhost -p 10514
```

Now, note your own IP address of your newly configured Syslog server:

```
/sbin/ifconfig
```

Configure this first system's firewall to accept connections to TCP port 10514:

Hint: on openSUSE (if you are not using openSUSE, you can instead use IPTables), start YaST, then Security->Firewall, Custom Rules->Add

Set the Source Network. For the simplest approach, "0/0" will allow connections from any computer. The security can (and should) be improved by only allowing connections from specific IP addresses or ranges.

Set the Destination Port to "10514"

From a second computer (such as a Linux VM, or another LinuxZ openSUSE image). If you like you can get a classmate to complete this from their system:

If the two computers you are using have the same hostname (as will happen if you are both using the LinuxZ image), the resulting logs will be hard to make sense of. (To find out the local hostname, run “hostname”.)

If you are using the LinuxZ image on both computers, change the hostname of this computer, so that it is different to the first computer:

```
sudo vi /etc/HOSTNAME
```

Edit “linuxz” to something else (no spaces).

Restart the second computer.

Check that the second computer can see the newly opened Syslog TCP port. Run:

```
nmap -Pn -p 10514 IP-address-of-new-syslog-server
```

(Where *IP-address-of-new-syslog-server* is as noted earlier.)

You should see the port state as “open”. If not, check the servers firewall rules, and recheck that the service is running.

Configure the second “remote” computer to send Syslog messages to the IP address of the new Syslog server...

```
sudo vi /etc/rsyslog.d/remote.conf
```

Uncomment (remove the #) from this line:

```
*.* @@remote-host
```

(Hint: note the two ‘@’ symbols which denotes TCP, not one, which would indicate UDP)

And change *remote-host* to *IP*:10514, where IP is the previously noted IP address of the server. For example, “*.* @@192.168.204.109:10514”.

Still on the second system, restart Syslog, for your changes to take effect:

```
sudo /sbin/service syslog restart
```

From that second system, all Syslog messages will now be automatically forwarded to the remote server. Run:

```
logger “my-name’s remote system”
```

(replace *my-name* with your actual name)

Confirm that that message was logged to the first computer, and that the other computer's name is shown in the logs.

```
sudo tail /var/log/messages
```

One of the benefits of logging to a remote system is that you have a backup of your logs, since they will be recorded on both machines. It is also wise to maintain a separate backup, incase both systems are compromised. Also, note that a spool can be configured, so that if there are network problems a system sending log messages will re-attempt to send them.

Take a screenshot showing that your /var/log/messages file includes messages sent from remote computers.

Label it or save it as "Logs-3".

Logrotate

A busy server may have a very large number of log entries. For this reason Unix has a "logrotate" tool, which moves old log events into a separate file, and compresses them so that they take up less space.

Logrotate typically runs daily, and as such a script for running logrotate may be present in:

```
ls /etc/cron.daily/
```

(Note the presence of a logrotate script.)

Open /etc/logrotate.conf:

```
sudo less /etc/logrotate.conf
```

As you can see, this file refers to the /etc/logrotate.d directory, which contains rules for managing logs. Press "Q" to quit less.

Look at the logrotate configuration for syslog:

```
less /etc/logrotate.d/syslog
```

Check whether there are any compressed messages files on your system:

```
ls /var/log/messages*
```

If you only see the file "/var/log/messages", then force logrotate to happen now:

```
sudo /usr/sbin/logrotate -f /etc/logrotate.d/syslog
```

> Now try "ls /var/log/messages*" again

Depending on how long your system has been running, you will probably notice the presence of many compressed log files over time.

Figure out how to decompress one of the previous log files, and inspect its contents.

Command history

Related to log management, is command history; which can be a useful source for identifying what happened during/after a security incident.

Process accounting involves logging every single command run by users. It can be activated using a simple command that logs the names of programs run as they are terminated (when the programs end). This works by activating a feature in the Linux kernel.

First create an empty log file with:

```
sudo touch /var/log/processaccounting
```

Now start the logging with:

```
sudo /usr/sbin/accton /var/log/processaccounting
```

> If the accton command is not available, install it. On openSUSE this can be done by first running "cnf accton", to find the name of the package containing the program, then run the install command it gives you, such as "sudo zypper install acct". Once the required software is installed, try the above again.

Run some commands on the system, then look at what has been logged:

```
sudo /usr/sbin/dump-acct /var/log/processaccounting
```

Note that the output is detailed, but only includes the name of the programs, not the path or arguments, so these logs may be misleading.

If you were to use process accounting in production, you would need to ensure that this command is run each time the system starts. (For example, using init.)

Bash (the most popular Unix interactive shell), also stores a history of commands run.

Run:

```
history
```

(for details: “man history”)

This file is written to each time a Bash shell terminates.

As you can see, a detailed record is kept. However, it is trivial to modify:

```
vi ~/.bash_history
```

If you start a new Bash instance your modified history will be loaded:

```
bash
```

```
history
```

Clear your history with:

```
rm ~/.bash_history
```

(this deletes the file)

```
history -c
```

(this clears the list from memory, so when you exit, the blank history is written)

Regular expressions

As you have seen, log files can be large and detailed. Looking for the details you are interested in can be tedious. However, use of regular expressions (AKA regexp or regex) can be very helpful, and make this much easier – they can help you to cut through the noise.

A regex is a pattern describing some text. A regex can be used to filter for matching text, or capture parts of matching text. There are lots of tools that accept regex patterns, such as the Unix commands grep, awk, sed, less, and vi. Regex are a powerful way of processing text, and as such regex are available in most programming languages, to make processing text much easier.

For the sake of explanation, the following examples will show how regex can be used to search log files.

As mentioned earlier, my messages log file includes:

```
Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: Setting up
rules from /etc/sysconfig/SuSEfirewall2 ...

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: using default
zone 'ext' for interface vmnet1

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: using default
zone 'ext' for interface vmnet8

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: Firewall
rules successfully set

Jan 21 17:16:26 linux-leedsmet pppd[32076]: Script /etc/
ppp/ip-down finished (pid 342), status = 0x0

Jan 21 17:16:26 linux-leedsmet pppd[32076]: Exit.
```

The simplest form of regex can search for standard characters, so the following regexp:

    leedsmet

would find matches in all of those lines. We can see this in action by using the command grep, which is a standard Unix command that searches through files for lines containing matches to a regex. The "--color" argument asks grep to highlight in colour the part of the lines that match. So on my computer I can search for the text "leedsmet", and the output would include those lines described previously:

```
sudo grep --color 'leedsmet' /var/log/messages

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: Setting up
rules from /etc/sysconfig/SuSEfirewall2 ...

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: using default
zone 'ext' for interface vmnet1
```

```
Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: using default
zone 'ext' for interface vmnet8

Jan 21 16:52:22 linux-leedsmet SuSEfirewall2: Firewall
rules successfully set

Jan 21 17:16:26 linux-leedsmet pppd[32076]: Script /etc/
ppp/ip-down finished (pid 342), status = 0x0

Jan 21 17:16:26 linux-leedsmet pppd[32076]: Exit.
```

Try searching through your own log file, for matches to a specific query. Run:

```
sudo grep --color ‘kernel’ /var/log/messages
```

The above should output all of the log entries that were sent by the kernel.

If we want to find something more complicated, we can use *regex special characters* to create patterns that match multiple texts. Run:

```
sudo grep --color 'sudo:\|su:' /var/log/messages
```

Tip: if at this point you see a long list of messages sent to the screen, you likely forgot to remove the Syslog rule you created earlier.

This will show all of the log entries created by a user attempting a su or sudo. This would be a security sensitive action that would be worth checking for.

This is achieved by the use of the “|” (pipe) character, which describes *alternative* patterns. Note that (somewhat confusingly) grep required a “\” (which is the escape character) before the special character. Unfortunately, this is not the case with most regex tools, usually it is the other way around, where special characters always have their special meaning ***unless*** they are escaped.

Another way of searching for the same thing would be:

```
sudo grep --color 'su\(do\)*:' /var/log/messages
```

This works by grouping with “( )” (parentheses); the “*” character states that the proceeding group must appear zero-or-more times. So “su” or “sudo” both match the above regex.

These special characters are used to state how many times the proceeding “atom” (section) occurs:

- *: zero or more
- +: one or more
- ?: zero or one
- {n}: n times: for example, a{2} would match aa
- {n,m}: min n, max m times

Some other special characters:

- [a-z]: a range of characters (defined in the square brackets), in this case any *one* lowercase character
- **.**: any character
- |: as in the example above, allows alternative values to match

Note that there are a few subtle varieties of regex (as we have seen, grep requires escape slashes before certain special characters), but for the most part the above rules apply.

So for example, run:

```
sudo grep --color 'su\(do\)*:.*USER=root.*grep' /var/log/
messages
```

Figure out what the above is matching, and make sure you understand how every section of the above regex works.

You can also do all kinds of neat tricks such as matching repeating text (using backreferences), and changing how "greedy" your matches are.

There are plenty of good tutorials available online. You can learn more about regex matching from sources such as:

man grep

http://gnosis.cx/publish/programming/regular_expressions.html

Using what you have learned, write a grep command that performs a regex on /var/log/messages for log entries sent by the kernel.

Once that is working, extend your regex to only match log events sent in the afternoon (12:00) today or yesterday.

Take a screenshot showing a grep command with it successfully returning log entries sent by the kernel, in the afternoon today or yesterday.

Label it or save it as “Logs-4”.

Windows and Logs

Although Windows does not natively support Syslog (it uses what is known as “Event Logs”, which can be accessed via the Event Viewer), it is possible to install software that makes this possible, allowing all of your logs to be centralised with Unix logs.

This resource provides some more information:

> http://www.syslog.org/logged/windows-syslog/

Note the additional challenge below, relating to Windows log management.

Additional challenges

New versions of Syslog (such as rsyslog) include TLS encryption for authentication and integrity. Your mission, should you choose to accept, is to correctly use encryption/PKI to protect against man-in-the-middle attacks and spoofing. Configure the Syslog server to only accept logs from your other computer, using PKI.

You may also be interested in eavesdropping on normal networked Syslog messages, to see whether you can sniff log traffic. Think about the security consequences.

Take screenshots of configuration changes and use, as evidence that you have completed this part of the task.

Label it or save it as “Logs-A1”.

Use logger to send a log message that would fool a sysadmin into thinking a sudo command was successful.

Take screenshots of the logger command and the tail of /var/log/messages, as evidence that you have completed this part of the task.

Label it or save it as “Logs-A2”.

On Windows, open Event Viewer, view the logs, and browse through the Security logs. Use the command line tool “eventcreate” to add a log event. Browse to the event you have created.

Take a screenshot your added log event in Windows, as evidence that you have completed this part of the task.

Label it or save it as “Logs-A3”.

Use two or more Windows systems, and configure remote logging, where all logs are forwarded to a central log server.

Take screenshots of configuration changes and use, as evidence that you have completed this part of the task.

Label it or save it as “Logs-A4”.

Download and install a log monitoring tool of your choice, and determine how to monitor sudo commands.

Take screenshots of configuration changes and use, as evidence that you have completed this part of the task.

Label it or save it as "Logs-A5".

What to show your tutor for XP rewards

Show your tutor each of the above (in red) evidences. This will be used to allocate XP for the module. Further details of the XP rewards and requirements are available on the *My XP* site.