UNIX Logging: Section 1 Transcript

Introduction

Welcome to the UNIX Logging module. A wealth of information about system configuration, system usage, user behavior, and application usage can be obtained from log files. This module will cover the various processes that create logs and the commands necessary to discover and view log messages. Interpreting log messages, examining the underlying mechanisms for filtering and storing log files, and log file rotation will be discussed. Finally, you will have the opportunity to use the Linux Audit service and learn about Security-Enhanced Linux.

Throughout this module, you will be presented with opportunities to assess and apply what you've learned.

At the end of this module, you will be able to:

- Explain system logging and how it is used,
- Recognize application logging and identify where application log files are stored,
- Discover alternate logging locations using pfiles and lsof,
- Analyze user login records to determine login history,
- Interpret log files to determine what actions are taking place on the system,
- Describe how syslog configuration determines how logging messages are handled,
- Use the auditd service to log system events, and
- Describe Security-Enhanced Linux (SELinux).

In addition to these topics, as you are traversing through this module, you will be expected to conduct Internet research to discover and learn about a variety of UNIX commands. Let's take a quick moment to review a few of the UNIX commands that you should become familiar with during this module.

Prerequisites

As you are traversing through this module, you are also expected to use Internet search engines and UNIX "man" pages to discover and learn about a variety of UNIX commands and syntax. Some of the commands that you will need to research and become proficient with are: `ausearch`, `journalctl`, `last`, `lastb`, `logadm`, `logger`, `logrotate`, `lsof`, `pfiles`, `sestatus`, and `setenforce`.

Before continuing on, take the time to research the UNIX commands displayed - especially any that may be unfamiliar to you. You may print a copy of the commands from the Resources.

Click Next when you are ready to continue.

Bypass Exam Introduction

If you are already familiar with the subject matter presented in this module, you can choose to take a

Bypass Exam to skip this module.

The Bypass Exam option provides a single opportunity to successfully demonstrate your competence with the material presented within the module. If you pass, you'll receive credit for completing the module, unlocking the content within, and you will be free to proceed to the next module. If you do not pass, you will need to successfully complete the module, including all exercises and the Module Exam, to receive credit.

Click the Next Section button to continue.

UNIX Logging: Section 2 Transcript

What is System Logging? 1/11

System logging is the recording of events that occur while the operating system is running. The idea behind system logs is similar to the use of a black box on an airplane that records what goes on with the airplane in case there is a problem. Most system logs are recorded chronologically and located within `/var/log/` (in Linux) or `/var/adm/` (in Solaris). However, they could be saved anywhere or nowhere, depending on the system configuration.

System logs are used to keep track of events that occur on a server. If there is a problem, system logs can be used to assist with tracing the source of the problem. System logs are also used for computer system management and security auditing, as well as generalized informational, analysis, and debugging messages.

System logs are supported by a wide variety of devices such as printers and routers across multiple platforms. On UNIX systems, the system log messages are received by the system log daemon.

System Message Logging Daemons 2/11

The syslog daemon is a service that implements system logging. The syslog daemon is controlled by a configuration file called `/etc/syslog.conf` in which you define logging rules and destinations for messages. Logging rules are defined by using a facility name and the severity level of the message. The facility name and severity level are passed to the syslog daemon from an application when it wants to log a message.

Additional logging daemons, which offer more robust features, have been developed since the release of syslog in the early 1980s. Let's briefly look at two of them.

rsyslog and syslog-ng 3/11

One successor to syslog is rsyslog, which stands for Rocket-fast system log. It offers many more features than syslog, including enhanced security and modular design. In particular, rsyslog configuration options, located in `/etc/rsyslog.conf`, include a directive to use additional configuration files in the directory specified by the `$IncludeConfig` variable, which is often set to `/etc/rsyslog.d/*.conf`. Thus, in addition to the main configuration file, you can use supplemental drop in configurations by adding them to this directory. Rsyslog also has many more options for sorting, saving, and sending messages. It can deliver over a million messages a second to local destinations, making it much faster than syslog.

Another alternative to syslog is syslog-ng, an open source implementation of the syslog protocol for UNIX. It extends the syslog model with content based filtering and adds more configuration options such as filtering messages according to the host that sent the message.

You can see which logging daemon a system is running by executing `ps -ef | grep syslog`.

Before we look at how to configure syslog daemons, let's look at how a typical syslog message is formatted.

System Log Message Format

Understanding the format of syslog messages and how they are filtered makes finding logs related to system problems possible and easier to troubleshoot. While log message fields may differ depending on the log message configuration and the logging daemon being used, they typically include the following four fields: a timestamp, hostname or IP address, the name of the program or process that generated the message, and the message itself.

This example shows the log file from a machine that has been rebooted. Notice the format of the messages. Each starts with a date and timestamp. Listed next is the host name. In some cases, log files list the IP address. This is followed by the process that generated the message.

Additional identifiers in the log message include the facility and severity levels which can be used to filter messages but are not written in the log file by default. Let's take a closer look at them.

Facilities and Severity

The facility is an identifier used to describe the application or process (i.e. source) that generated the log message. For example, a facility of "kern" or "0" indicates a kernel message, and a facility of "ftp" or "11" indicates an FTP message. Local facilities (local0 - local7) are reserved for custom use.

The severity level indicates the importance of the message. Severity levels range in importance from zero to seven where 0 indicates an emergency while 7 is simply a debug message.

Combining facility and severity scores can determine the priority of a message and affects the way it is handled based on the system log daemon configuration. An imminent system failure naturally has a different severity than a debugging message and will trigger different actions. The rules that determine how messages are handled are set in the daemon config file.

Refer to the Resources for a complete set of facilities and severity definitions. Next, we are going to look at a syslog configuration file.

System Log Configuration

The system log configuration files set the rules on how log messages are handled, based on their facilities and severity. The system log daemon reads its configuration files when started and loads them into a running configuration. Applications and processes send their log entries to the system log daemon with their corresponding facility and severity. The system log daemon then handles the logs received based on the daemon's running configuration. When a message matches a loaded rule, the log message is processed according to the rule.

Messages can be processed in numerous ways, such as writing them into a file, sending them to another host, or saving them in a database. The running configuration controls what messages are written where, and what actions, if any, are taken when certain messages are received.

In this `/etc/syslog.conf` file, let's look at the line that writes messages to `/var/log/messages`.

The line can be broken down into two main sections: the type of message and what should be done with the message. Since this line ends with `/var/log/messages`, it is writing the matching message to this file. This line has four facility dot severity clauses each separated by a semicolon (;).

Looking at the first clause, `*.info`, the asterisk (*) means match any facility, while `info` means match any severity of informational or worse. It's worth noting that the lower the numeric identifier for a severity level, the worse it is - "0" being the lowest and "7" being the highest. Matching messages could have a severity of Emergency, Alert, Critical, Error, Warning, Notice, or Informational.

The `mail.none` clause means that the action on this line should not be taken for messages with a facility of mail. 'None' is not an actual severity level; it serves to avoid the associated action on messages with the specified facility. As you can see from the last two clauses, `authpriv` and `cron` messages are not included in `/var/log/messages`. As you look at other lines further down in the file, you will notice that `mail`, `authpriv`, and `cron` messages are being saved into their own files.

Now let's explore the different locations where syslog messages can be recorded.

Message Recording for System Logging

Because many actions that take place on a server are logged, log files can get very big very quickly. Rules in the system log configuration files filter messages and send them to different specified locations.

On Linux and Solaris systems, the configuration specifies how and where the system log messages are recorded. There are many different options for where to write the messages such as in a file, in a database, or by sending them to a remote logging server. You need to examine the configuration file to determine where messages are being recorded.

By default, most Linux distributions send their standard log messages to `/var/log/messages` and send their security and authorization messages to `/var/log/secure`. Solaris systems, by default, send their standard log messages to `/var/adm/messages`, but their authorization messages are not logged.

It is common for hosts to send their syslog messages to a remote syslog server for storage and for possible analysis by a security tool for malicious activity. You can tell system log messages are being sent to a remote server by looking in the configuration file for a line that ends with one or two "@"s followed by a hostname or IP address.

For example, to send the same log messages that are being written to `/var/log/messages` on linux.ops.local to the webserver www.ops.local via UDP, add the line displayed on the screen: "`*.info;mail.none;authpriv.none;cron.none @www.ops.local`" to the configuration file.

When a system is configured to send log messages to another host, the messages are sent almost instantly. By default, log messages for most system logging daemons (e.g. syslog and rsyslog) are

sent via port 514 over UDP. However, the system may be configured to send log messages using other methods, ports, and protocols. For example, system log messages can be sent encrypted, over TCP (indicated by "@@" in the configuration file), or using protocols such as RELP, The Reliable Event Logging Protocol.

Now that we know where messages are recorded, let's discuss how to search through system logs.

Searching Through System Logs

System logs are typically written in human-readable format. Even so, system administrators are unlikely to read each log line-by-line every day; it is more likely that system log files are parsed for specific events that may have occurred (e.g. kernel panics, mail errors, or excessive failed login attempts). You can use `grep` to find entries that match a regular expression that describes what you are looking for. For instance, if you wanted to isolate all of the password related entries in `/var/log/secure`, you could execute the command `grep password /var/log/secure`.

You could further refine the results to isolate a certain date by using the pipe symbol to redirect the output of the grep program to the input of another. This command `(grep password /var/log/secure | grep "Nov 17")` returns password related entries from November 17th.

You could then pipe that output to yet another program, such as `wc -l`, which would return a count of the lines in `/var/log/secure` that contain "password" and "Nov 17".

Conversely, it is sometimes useful to filter out expected log entries in order to highlight unexpected log entries. For example, if you want to show any entries that do not contain "192.168" in `/var/log/secure`, you could execute `(grep -v "192.168." /var/log/secure)` the command displayed on the screen.

Next, let's look at systemd logging.

systemd Logging

The systemd initialization system provides `systemd-journald`, a daemon that is used for system logging. Other system logging daemons are not required for system log messages to be recorded. The systemd journal is configured by modifying the `/etc/systemd/journald.conf` configuration file.

By default, the systemd journal logs all system messages to the same location in memory in chronological order. Thus, when the system is rebooted, the log messages prior to the reboot are no longer accessible. Journald can be configured to store logs on disk persistently. If configured, the default location to store the journal logs is `/var/log/journal`. This directory does not typically exist unless the journal is configured to log persistently to disk.

The systemd journal can also be used in conjunction with other system logging daemons, such as rsyslog. If both are used, log messages are sorted by facility and severity using the same configuration files that we previously discussed. In fact, by default, CentOS and RHEL 7.0 use rsyslog and systemd-journald to process the system log messages. Although the journal is not configured to persistently log to disk by default, the rsyslog daemon intercepts log messages passed

to it via the systemd journal and persistently logs them to files in ASCII text format based on the rsyslog configuration.

journalctl

Irrespective of the configuration for persistent systemd journal logging, you must use the `journalctl` command to read the binary-formatted file. Without any options, the `journalctl` command displays the full contents of the journal, starting with the oldest entry. The `journalctl` output displays messages of severity `notice` and `warning` in bold text. Messages in red text have a severity level of 0-3 (emerg, alert, crit, err).

Due to the verbosity of logging, you may wish to limit the output by passing options with the `journalctl` command. Some useful `journalctl` options are:

- `-p`, or `--priority`. This option filters messages using traditional syslog severity levels (i.e. emerg, alert, info, etc.)
- `--since` and `--until`. These options filter messages either since or until the time specified using the format `"YYYY-MM-DD HH:MM:SS"`

Note: If the time is omitted, it will assume "00:00:00"; if seconds are omitted, it will assume ":00". If the date is omitted, the current date is assumed. Alternatively, you can also use the strings "today", "tomorrow", "yesterday" or "now". Also, relative times can be used by prefixing the time with a "-" or "+", referring to the times before or after the current time, respectively.

- `-u, --unit`. This option filters messages based on service unit.

For more information, refer to `man 1 journalctl, man 7 systemd.journal-fields`, or `man 7 systemd.time`.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

UNIX Logging: Section 3 Transcript

Application Logging

Just like the system logs messages, applications running on a machine may also create logs. Each application can have a different place to store its log files or it can perform logging through the system logging daemon. It's not always easy to locate these application log files. Whether an application logs or not, and where it stores log files is completely dependent on the developer of the application.

If the application is using the system logging daemon to create its log files, the location of these files may be determined by looking at the system log configuration files. By default, the CentOS system logger writes the following application logs: `/var/log/maillog`, `/var/log/cron`, `/var/log/spooler` and `/var/log/boot.log`. Often times, if an application is written to use the system logging daemon, it will typically be configured to log to one of the local facilities, which are `local0 - local7`. Which facility it logs to, if any, can be difficult to determine.

If the application is directly writing its logs, it can be more difficult to find the log files. You can see if the application has a configuration file that may contain the location of its log file. A good example of this is the Apache webserver. Apache's configuration file is usually named `httpd.conf` (although its location on the system may vary, it's typically located within `/etc/http`). Its configuration file contains a directive named ErrorLog which specifies the location for its error messages to be written.

When a configuration file is not found or you think the application may create other files, you will need to see what files the application has open for writing. On Solaris, the `pfiles` command is used to show this information. On Linux, the `lsof` command is used. These commands provide information related to a running process, including the files that it has open.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

UNIX Logging: Section 4 Transcript

Logfile Rotation

Log files can become very large and take up an increasing amount of disk space. To limit the space that log files consume, they can be archived at set periods of time or when they reach a set size. This process is typically referred to as logfile rotation. This is an automated process that renames the current log file and creates a new one, but it can also be configured to archive or compress the old files.

A common method used for rotating log files is to append a number to the end of the file name, although this is configurable. Each time a new log file is started, the numbers in the old file names are increased by 1, so the files "rotate" through the numbers. A threshold can be configured so that when the oldest file name is increased to a certain number, the file is deleted to conserve space. Some Linux operating systems append the date of the log file rotation to the end of the filename. When a set number of files, age, or filesize is reached, the oldest log file is removed.

Many options are available for log rotation, such as custom file naming and file compression in many different formats. The Solaris service that rotates log files is called `logadm` and its configuration file is `/etc/logadm.conf`. Linux uses a script named `logrotate` to run its file rotation program. By default, `logrotate` is run by the `cron` service daily from the `/etc/cron.daily/logrotate` script. The main configuration file for `logrotate` is `/etc/logrotate.conf`; however, additional files may be located inside the `/etc/logrotate.d` directory.

Logfile rotation can be configured to rotate system logs or any application log on the system.

Next, let's discuss some other logs that contain data that will be of interest for gathering information about a server.

User Login Records

In addition to the system log files and the application log files, there are other log files that contain information regarding successful and unsuccessful user access to the system.

An accounting of the current status of user logins, user logouts, system boot time, and system events is contained in `/var/run/utmp` on Linux and `/var/adm/utmpx on Solaris`.

A historical record of the information contained in the `utmp` file is recorded in the `wtmp` log. The location of this file on Linux is `/var/log/wtmp`. On Solaris, this information is contained in `/var/adm/wtmpx`.

Linux systems also maintain a log of unsuccessful login attempts in `/var/log/btmp`. This information is useful to determine what users are trying to get into the system and from where they are trying to get in. This log file could be used to discover brute force login attempts.

Unlike most system and application log files, these logs are in binary format. You must use the `last`

or `lastb` commands to read their contents. By default, the `last` command reads `/var/log/wtmp` on Linux and `/var/adm/wtmpx` on Solaris. `lastb` is the same as `last`, except that by default it reads `/var/log/btmp` on Linux. That said, `last` can display the contents of any of these files, including rotated versions of them, using the command-line option displayed on the screen. `(last -f /var/log/wtmp.1)`

Linux Auditing

The Linux auditing system is an access monitoring and accounting service which is integrated with the kernel and typically runs as the `auditd` service. It is capable of tracking security-relevant information and logging information about events happening on the system. The audit system provides the option to write a log when a specified command, system call, or event occurs. It can monitor access to any object, such as a file, a network socket, or a device. `Auditd` is used to monitor and log violations of security policies, but does not provide additional system security.

`Auditd` logging is enabled by default for login/logout, boot/reboot, screen lock/unlock, and sudo/su. The configuration file for `auditd` is located in `/etc/audit.conf` and the rules it uses are located in `/etc/audit/audit.rules` and possibly also in files within the `/etc/audit/rules.d` directory.

The audit logs are written by default to `/var/log/audit/audit.log`. These logs are clear text, however the format of these logs is very different from typical system logs. In particular, date/timestamps are recorded in epoch time, which can be cumbersome to convert to local time. It is much easier to read audit logs using the `ausearch` command. An option is available for audit logs to be written to another system logging daemon such as rsyslog. Because of the integration with the kernel and detailed logging provided by `auditd`, it is typically used as the primary logger for SELinux.

Security Enhanced Linux

SELinux is a Linux kernel security module that provides mandatory access controls (MAC). SELinux is a set of kernel modifications and tools added to Linux distributions. The origins of its key concepts can be traced to National Security Agency projects.

SELinux controls the access a system allows each user, process, and daemon to have; it is used to confine these by limiting what objects they can access on the system. This minimizes the amount of damage resulting from a system compromise. SELinux has three modes of operation.

The first mode is Enforcing, which is the default in CentOS/RHEL 7.0. In enforcing mode, the security policy is enforced, blocking applications from working if there is no policy in place to allow access. A denial is logged by `auditd` with a type of Access Vector Cache (AVC). The `auditd` logs for SELinux always begin with `"type=AVC"` and when a block occurs, the log contains the word `"denied"`. One other item in the log you need to pay attention to is `"comm="`, which is followed by the command that SELinux is blocking. In this case, SELinux is blocking a shared object needed for the Apache webserver.

The second mode is Permissive. Permissive mode does not prevent any access operations from

occurring, however it will log an `auditd` AVC message. Permissive mode is commonly used to verify the policy is working correctly without blocking operations. Once the policy has been verified, the system is typically placed into Enforcing mode.

The last SELinux mode is Disabled. In disabled mode, SELinux does not produce any logs or block any processes. SELinux is shut down.

The configuration file `/etc/selinux/config` determines the default mode for SELinux on boot.

SELinux Management Commands

SELinux has several commands that are used for management. We will discuss a few of the commands that you'll need to know: `sestatus`, `getenforce`, and `setenforce`.

The command `sestatus` is used to determine the current mode that SELinux is running. The output of this command displays a block of information which includes the "current mode". If a process you are trying to use is not working and you see that it is blocked in the AVC audit logs, you will need to change the mode SELinux is running to allow your process to work.

The command `getenforce` simply returns the current SELinux mode with no additional information.

Use the command `setenforce` to switch the currently running mode between permissive and enforcing. The argument for `setenforce` is the mode you want SELinux to be running. The mode argument is specified as Enforcing (1) or Permissive (0). After changing the mode, you will see the value of Current mode change in the output of `sestatus`.

In order to switch into or out of Disabled mode, the system must be rebooted. No command can enable SELinux when it is disabled. The configuration file will need to be changed and the system rebooted. Alternatively, a kernel parameter for SELinux could be set during boot.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

UNIX Logging: Section 5 Transcript

Summary

You have completed the UNIX Logging module. During this module we discovered how much information about system configuration, system usage, user behavior, and application usage can be obtained from log files. You should now be able to:

- Explain system logging and how is it used
- Recognize application logging and identify where application log files are stored
- Discover alternate logging locations using pfiles and lsof
- Analyze user login records to determine login history
- Interpret log files to determine what actions are taking place on the system
- Describe how syslog configuration determines how logging messages are handled
- Use the auditd service to log system events, and
- Describe SELinux

To receive credit and advance to the next module, you must achieve a passing score on the Module Exam. Click Next to begin the Module Exam.