



# System Administrator Essentials: Monitoring Log Files

Cybersecurity

5.3 Archiving and Logging Data Day 3



## 5.3: Linux - Logs

---

### Class Preparation

1. Check into BCS
2. Update your git repository with ``git pull``
3. Launch and login to your Ubuntu VM, and bring up the Terminal

### Homework

- Unit 4 (Linux SysAdmin): due Sunday October 25
- Unit 5 (Linux Arch/Log): due Sunday November 1
- “Incomplete” = Permissions issue or wrong link

### Upcoming Units

- Unit 6: *Bash Scripting/Programming* (10/26 - 10/31)
- Unit 7: *Windows Admin and Hardening* (11/2 - 11/7)
- Unit 8: *Networking Fundamentals* (11/9 - 11/14)

### Schedule Notes

#### *Thanksgiving Break - No Class*

- Off: Wed 11/25 & Sat 11/28
- Return on Monday 11/30

#### *Winter Break - No Class*

- Last class on Sat 12/19
- Off: Mon 12/21 - Sat 1/02
- Return on Monday 1/04

# Class Objectives

---

By the end of today's class, you will be able to:



Use `journalctl` to filter cron and boot log messages.



Perform log size management through the use of Logrotate.

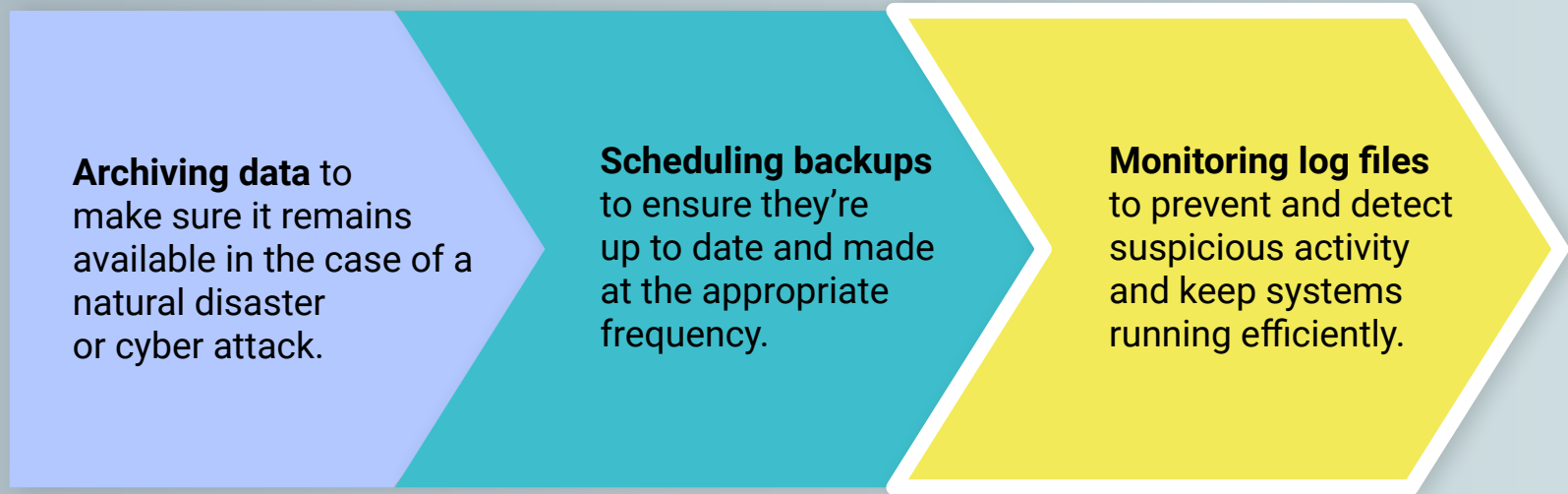


Install and configure audit rules using `auditd` and write audit logs to disk.

# Monitoring Log Files

---

Today, we will continue our overview of logging and further explore the importance and security implications of properly managing logs.



# Let's Recap

Logs are very valuable to an organization's technical and security teams.

They provide an enormous amount of information on various aspects of a network, including security, server performance, and system errors.

Logs are a valuable source of data that contains Personally Identifiable Information (PII). This information can be exploited and therefore must be protected.



# Let's Recap

The importance of these resources means we need **proper log management**:

Ensuring logs are protected through detailed recordings of changes.

Storing logs for a sufficient amount of time.

Omitting unnecessary data to avoid excessive and gratuitous logs.



# Log Management Security Implications

---

When we properly manage our logs, we are able to better analyze and review them regularly, letting us rapidly pinpoint threats, regulatory violations, and fraudulent activity.



# What Does Log Management Look Like?

---

Throughout this class, we'll cover the following steps and tools used by sysadmins to manage logs.

## Investigate an Issue

**For example:** Applying log filters during log reviews to scope out past or current events.

A log filter is a tool for extracting specific information from a log.

## Size Management

Creating a log size management system that rotates logs to preserve log entries and keep log file sizes manageable.

Log rotation is closing, dating, and moving logs to another location, and replacing them with empty files.

## Audit

Installing and configuring a log system that audits system file changes and records those changes to disk as audit records.



# Overview of Logs

# Overview of Logs

---

Linux stores all log files in a centralized repository located in **/var/log**.  
For example:



**/var/log/auth.log** stores authentication related events. Used to:

- Detect failed login attempts.
- Detect other vulnerabilities related to user authorization mechanism and brute force attacks.



**/var/log/cron.log** stores information related to cron jobs. Used to:

- Log information when a cron jobs runs recording successful execution of applications as well errors or failures.
- Check error messages when a cron job fails.

# Four Categories of Logs

---

Most log directories can be grouped into four categories:

## Application Logs

Store alerts generated by software being used by the user, including when it's launched, how long it's in use, when it's closed, etc.

## Event Logs

Contain information regarding security related events. E.g., a user succeeds or fails to log onto a host, or tries to install unauthorized software.

## Service Logs

Contain information related to system services such as cron jobs and print jobs.

## System Logs

Contain information regarding system events such as boot messages, kernel errors, or anything related to the system hardware.

**System Startup**

**Printing Documents**

**Cron Job Activity**

**Improper Shutdown**

**Successful Logins**

**Failed Logins**

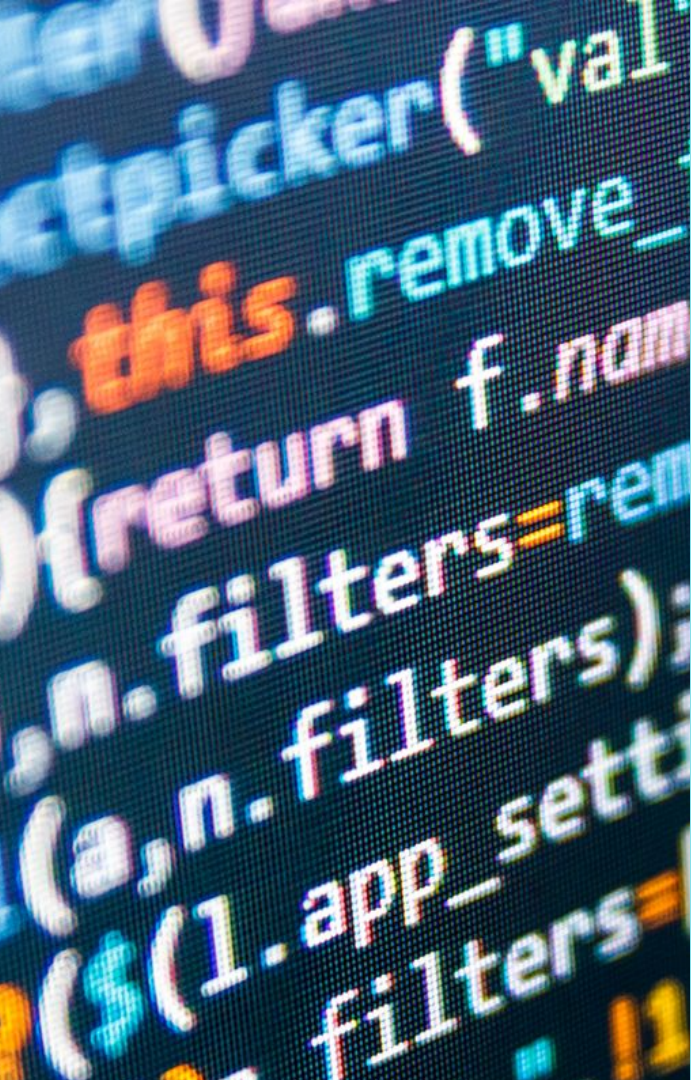
How can we  
manage and filter through  
the overwhelming amount  
of information produced  
by logs?



**journalctl** is designed to filter through enormous system logs and return specific results.



journalctl



# Introducing journalctl

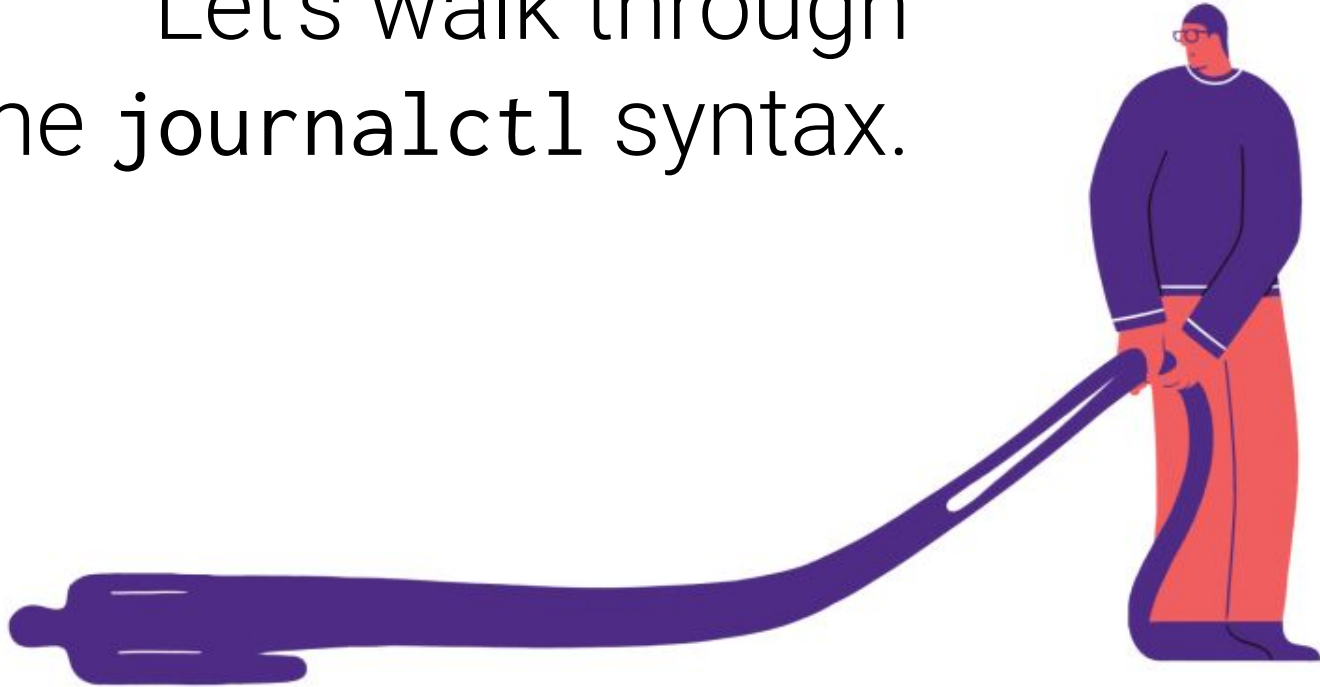
---

Before looking into `journalctl`, let's look at the underlying system responsible for tracking this information:

- `systemd` is a daemon that can be used for logging system-wide events and providing information to other tools.
- `systemd` does not provide reader-friendly display of log information.
- `journald` collects and stores log information in a structured, indexed format.
  - `journald` is often referred to as `systemd-journald`

`journalctl` allows us to access the `systemd-journald` journal and filter out desired information.

Let's walk through  
some `journalctl` syntax.







# journalctl Syntax

---

`journalctl [options] [information being filtered]`



`journalctl` without filters returns the entire (massive) contents of the system log.



`journalctl --list-boots` displays lines for each individual boot.

```
sudo journalctl --list-boots
```

```
-2 915e5048b12b4b79b71ee3d0f71ce6ca Thu 2019-11-07 21:03:23 EST-Thu 2019-11-07 23:49:16 EST
-1 69f1499b462946baab1bc26c593690cc Thu 2019-11-07 23:49:33 EST-Fri 2019-11-08 00:22:53 EST
 0 edb3c812a22d43d390c393d18ba207f1 Fri 2019-11-08 12:24:26 EST-Fri 2019-11-08 13:04:01 EST
```

# journalctl Syntax

---

`journalctl [options] [information being filtered]`



`journalctl -ef`

- `-e` displays the end of the journal
- `-f` enables **follow mode** to keep the journal screen open and displaying real-time messages in order of occurrence.



`journalctl _UID`

- View systemd-journald logs for a user.
- We can see a list of user IDs by running `cat /etc/passwd`.

## Demo Scenario

---

Next, we'll demonstrate `journalctl` using the following scenario:

- The IT administrator recommended that you enable log persistence due to a huge loss of log data that resulted from this unplanned system reboot and improper shutdown.
- Log persistence is the process of saving system logs across reboots to ensure they are not lost.
- In this demonstration, we'll also show how to use `journalctl` to identify suspicious activity, such as malicious user account creation and unauthorized login attempts.

# Demo Steps

---

We need to properly trace a series of booting events with the following steps:

01

Use **journalctl** to query the **systemd** journal.

02

Edit **/etc/systemd/journald.conf** to establish persistent storage

03

Use **-ef** to displays real-time messages in order of occurrence.

04

Use **\_UID=** display journal data for specific users.



# Instructor Demonstration

journalctl Demo



## **Activity:** Log Filtering

In this activity, you will filter through log files to investigate suspicious activity and determine if a system breach occurred.

**Suggested Time:**  
25 Minutes



# Log Size Management





# Log Size

---

Log files preserve information regarding system events for a fixed period of time. But logging daemons cannot control file size.

**If unchecked, log files can grow to unmanageable sizes that potentially consume all available space.**

- Imagine you were asked to check system logs for any signs of a possible breach.
- Now, imagine that the server has been logging data non-stop since the system started running two and a half years ago.
- Querying a log file with that much data would be daunting. It would take the time and resources of both the server and administrator.



# Managing Log Sizes

---

**Log rotation** is the process of archiving a log once it reaches a specific size or a point in a set schedule, and rotating it out with a new, empty log.



Today, we will do this using a command line tool known as **Logrotate**.

# Log Rotation

---

Some of the uses and benefits of log rotation:

01

Scheduling the creation of new log files.

02

Compressing log files to save hard drive space.

03

Executing commands prior to and after a log is rotated.

04

Time stamping old logs and renaming them during rotation.

05

Log file archive pruning to maintain only a certain number of backlogs.

06

Smaller archives mean faster transfer times.

# Logrotate Configurations

---

Logrotate works through a series of configuration files that indicates the log files to rotate and the specific parameters to apply to those files

- The main configuration file **/etc/logrotate.conf** contains default options and parameters that logrotate will use. It also indicates the specific files that these default parameters apply to.
- Some applications will require unique configurations that do not fit the default parameters found within **/etc/logrotate.conf**. Those files are handled in other configuration files, which we will look at in the upcoming demo.

The following is the content of a typical /etc/logrotate.conf file.

- weekly: Rotate out existing logs every week
- rotate 4: The number of rotations before a log is removed or emailed.
- create: Create new (empty) log files after rotating old ones
- #dateext: Will add date to end of rotated log
- #compress: Uncomment this to compress rotated log.

```
# see "man logrotate" for details
```

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4
```

```
# create new (empty) log files after rotating old ones
create
```

```
# use date as a suffix of the rotated file
#dateext
```

```
# uncomment this if you want your log files compressed
#compress
```

```
# packages drop log rotation information for this
directory
include /etc/logrotate.d
```

```
# system-specific logs may also be configured here.
```

The following is the content of a typical /etc/logrotate.conf file.

We'll take a look at # packages drop log rotation information for this directory and the /etc/logrotate.d directory in the upcoming demo.

```
# see "man logrotate" for details
```

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4
```

```
# create new (empty) log files after rotating old ones
create
```

```
# use date as a suffix of the rotated file
#dateext
```

```
# uncomment this if you want your log files compressed
#compress
```

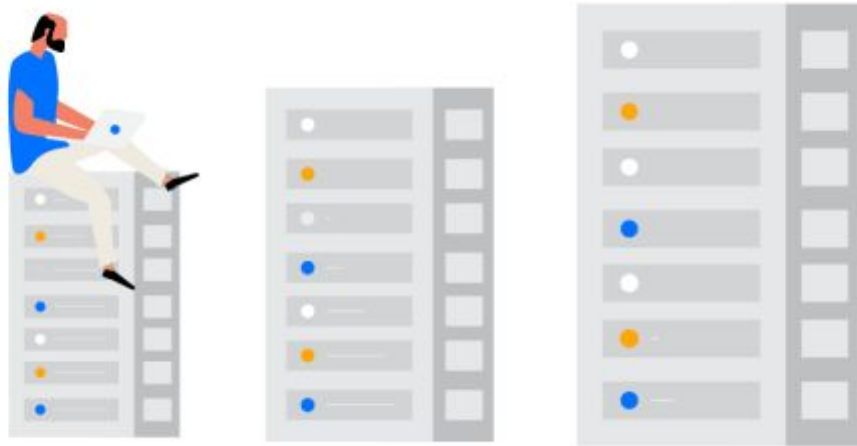
```
# packages drop log rotation information for this
directory
include /etc/logrotate.d
```

```
# system-specific logs may also be configured here.
```

# Logrotate Demo Scenario

---

We will use Logrotate for mail.log that will:



- Keep eight weeks of backlogs.
- Rotate logs daily.
- Create new empty logs after rotating out old ones.
- Not rotate empty logs.

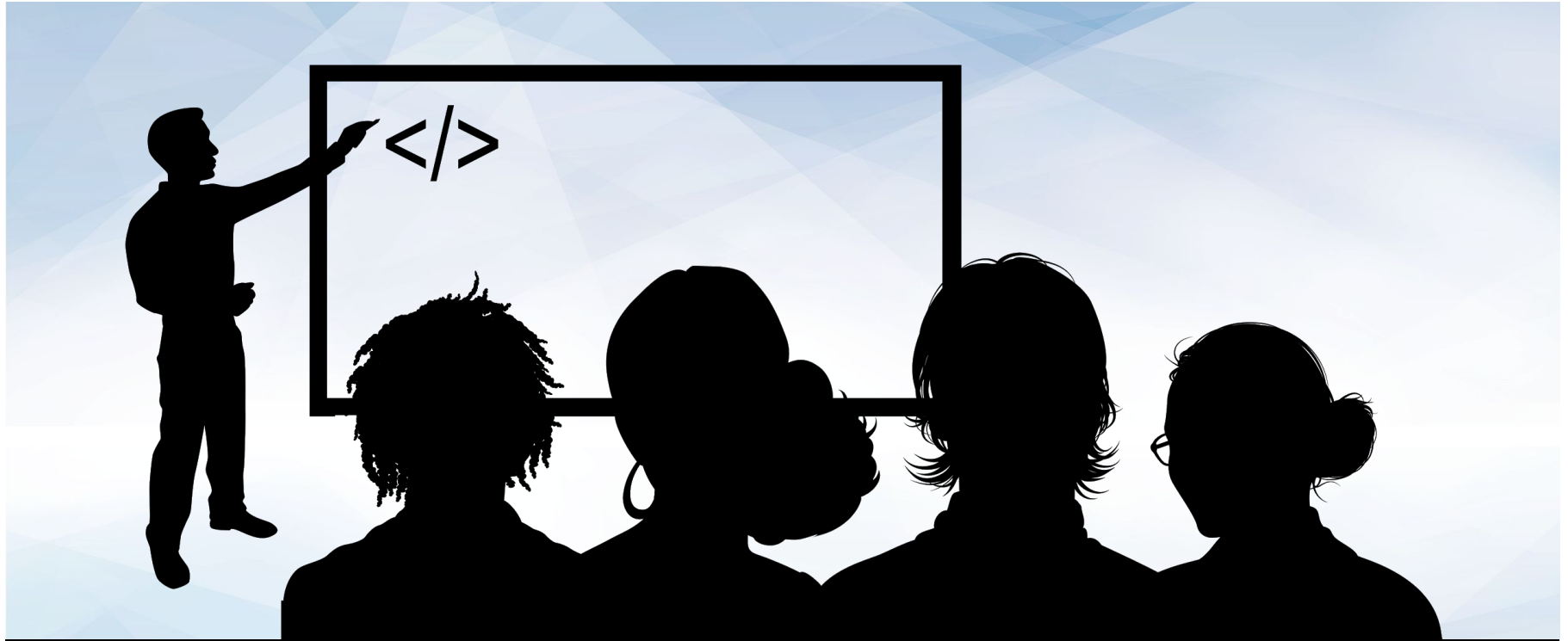
# Logrotate Demo Scenario

---

**In order to complete this task, we will:**

1. List the directories in logrotate.d to display the configuration files for installed applications.
  - If the config file exists, we can edit it.
  - If the config file doesn't exist, we need to add a configuration file to /etc/logrotate.d.
2. Use nano to create a file called mail that sets up the following logrotate configurations:
  - Use the rotate setting to keep the most recent eight weeks of backlogs.
  - Use the create setting to create a new log every time the old log is rotated.
  - Use the notifempty setting to not rotate empty logs.

Test the configuration changes by performing a manual test rotation



# Instructor Demonstration

Logrotate





## **Activity:** Log Size Management

In this activity, you will use Logrotate to minimize log size.

*Activity file shared by the instructor.*

**Suggested Time:**  
20 minutes



Now, we will  
learn **how to perform**  
**log audits** to track  
violations on a system.



# Log Auditing

## Consider the Following Situation:

---

An organization experienced a breach. The organization knows they've been breached, but don't have a way of knowing what changes the attackers made to the system.

- That information would offer insight into the tactics used by the attackers, and provide crucial assistance for incident and recovery efforts.
- **auditd** fills this gap by showing modifications made to a system. It can't show every single change made, but does provide very useful information.

Linux auditing system is an excellent way for sysadmins **to create log rules for nearly every action on a data center server or user host**. This system allows you to track and record events, and even detect abuse or unauthorized activity, via log files.

# auditd Overview

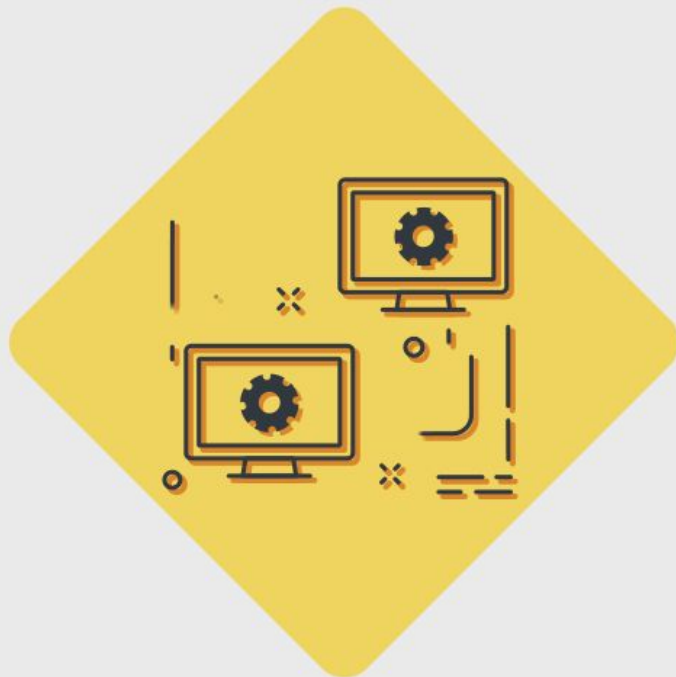
---

**auditd** is a **kernel** level subsystem that can watch every **system call** an application makes.

- **Kernel** is the core component of any OS, responsible for system memory, processes, task, and disk management. The kernel links all system hardware to the application software.
- **System call** is when any software or application makes a request for system resources.

**auditd** integration with the system kernel allows it to monitor all system operations, such as network traffic and file system access.

**auditd** does not provide any additional security actions, rather it *allows us to monitor existing violations*.



# auditd

---

Once an event is written to disk, reporting tools such as **ausearch**, **aureport**, and **aulast** are used to generate reports.

## **ausearch**

Tool designed to query **auditd** daemon logs based on different search criteria for event-driven log records.

## **aureport**

Program that summarizes various types of events.

## **auditctl**

Responsible for configuring the **auditd** system. Has the capability to enable or disable **auditd** systems, load and list rules, and generate status reports.

# auditd Demo Setup

---

Now, we'll demonstrate how to use **auditd** with the following scenario:

- There was a breach and several logs were deleted when attackers were attempting to clear their tracks.
- The security manager advised that the attackers may have created new user accounts to gain persistent network access.
- We must find out the details of any new user accounts to figure out the attackers' end goals.





Instructor Demonstration

**auditd**



# Demo Summary

---

In the previous demo, we completed the following tasks:

1. Edit `/etc/audit/auditd/auditd.conf` and specify:
  - Log file location for `auditd.log`.
  - Retain no more than 50 logs.
  - Maximum log file size of 100.
2. Use `auditctl -l` to see if any rules exist.
3. Edit `/etc/audit/rules.d/audit.rules` and add files to monitor.
4. Use `auditctl -l` to verify the new rules exist.
5. Use `systemctl restart auditd` to restart the `auditd` daemon.
6. Use `auditctl -w` as an alternative way to add a new rule.
7. Use `auditctl -l` to verify the new rule was added.
8. Use `aureport -au` to perform log search for user authentications.
9. Use `aureport -m` to search for account modifications.



## **Activity:** Event Monitor Log

The local server in your organization was hit with MedusaLocker, a nasty ransomware attack that left all of the organization's hard drives crypto-locked.

You need to enact an event monitoring system that writes audit records to disk and creates audit log reports.

*Activity file shared by the instructor.*

**Suggested Time:**  
**25 Minutes**

