System Administrator Essentials: Monitoring Log Files

5.3

Cybersecurity

Archiving and Logging Data Day 3

Class Objectives

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

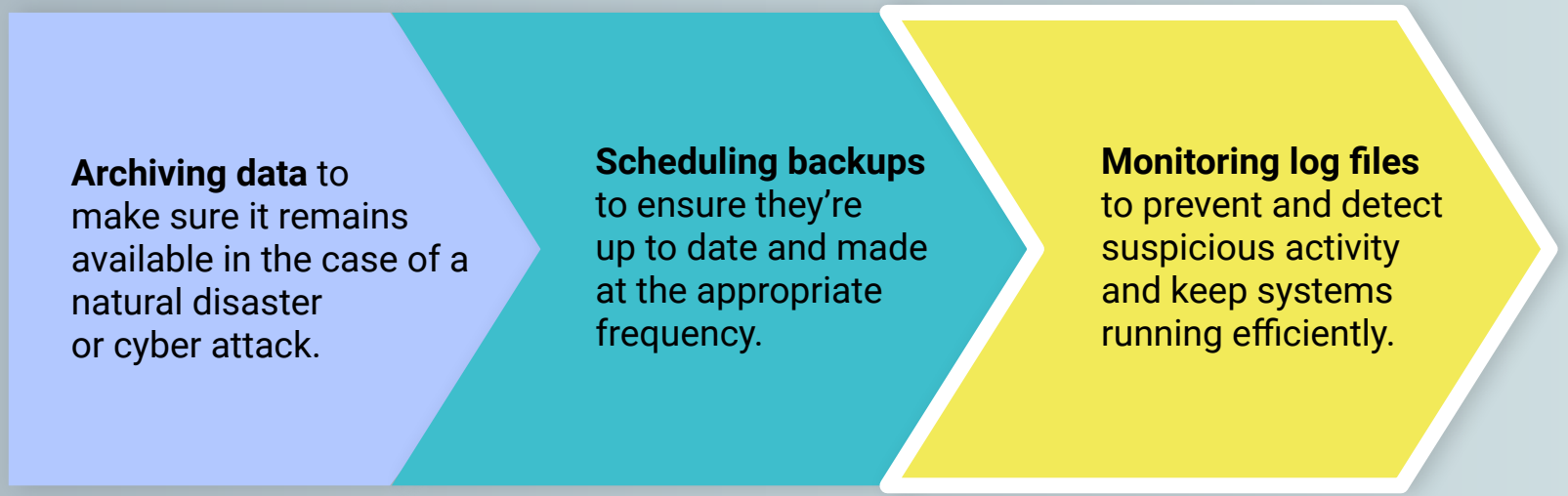
Filter cron and boot log messages using **`journalctl`** and **`ryslog`**.

Perform log size management through the use of Logrotate.

Install and configure audit rules using **`auditd`** to 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

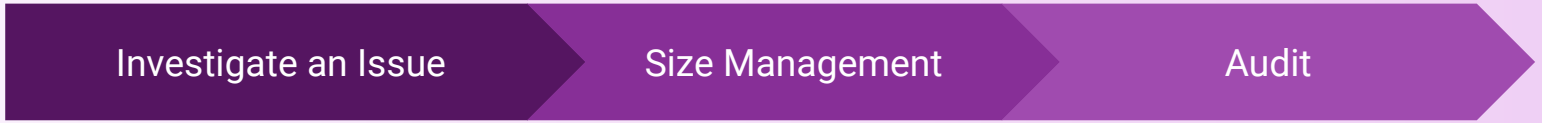
When we properly manage our logs, we are able to better analyze and review them regularly, letting us rapidly pinpoint threats, regulatory violations, and fraudulent activity.

From properly managed logs, we can learn of:

- **Indicator of Attack (IOA):** Attacks in progress.
- **Indicator of Compromise (IOC):** Attacks that already happened**.**

What Does Log Management Look Like?

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



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.

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.

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.

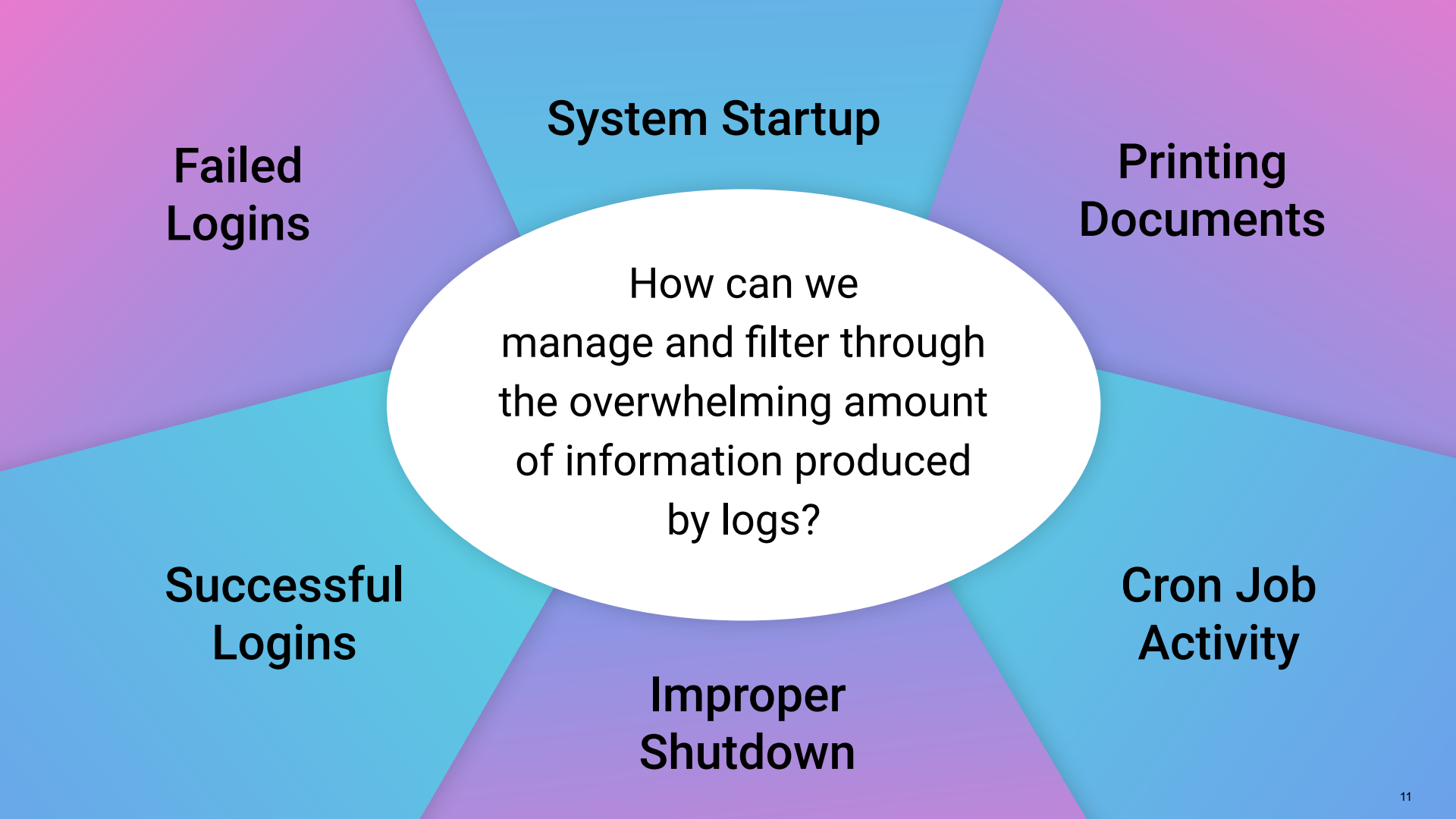
`/var/log/cron.log` stores information related to **`cron`** jobs. Used to:

- 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.

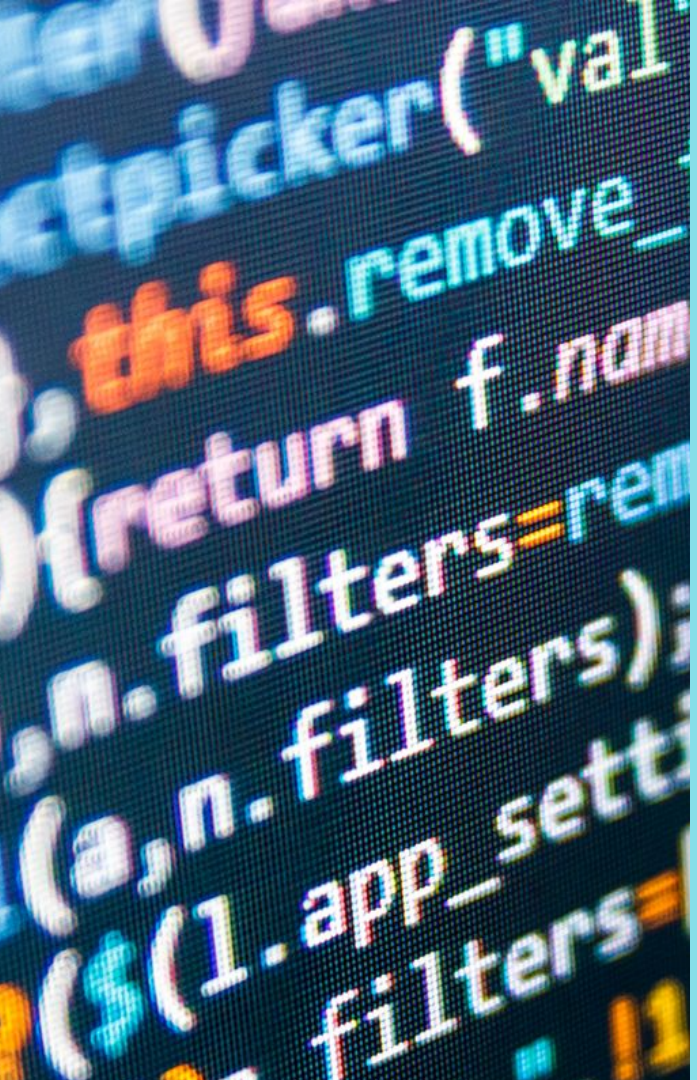
Four Categories of Logs

Most log directories can be grouped into four categories:

Application Logs	Event Logs	Service Logs	System 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.	Contain information regarding security related events. E.g., a user succeeds or fails to log onto a host, or tries to install unauthorized software.	Contain information related to system services such as cron jobs and print jobs.	Contain information regarding system events such as boot messages, kernel errors, or anything related to the system hardware.


System Startup
Failed Logins
Printing Documents
How can we manage and filter through the overwhelming amount of information produced by logs?
Successful Logins
Cron Job Activity
Improper Shutdown

journalctl



Introducing `journalctl`

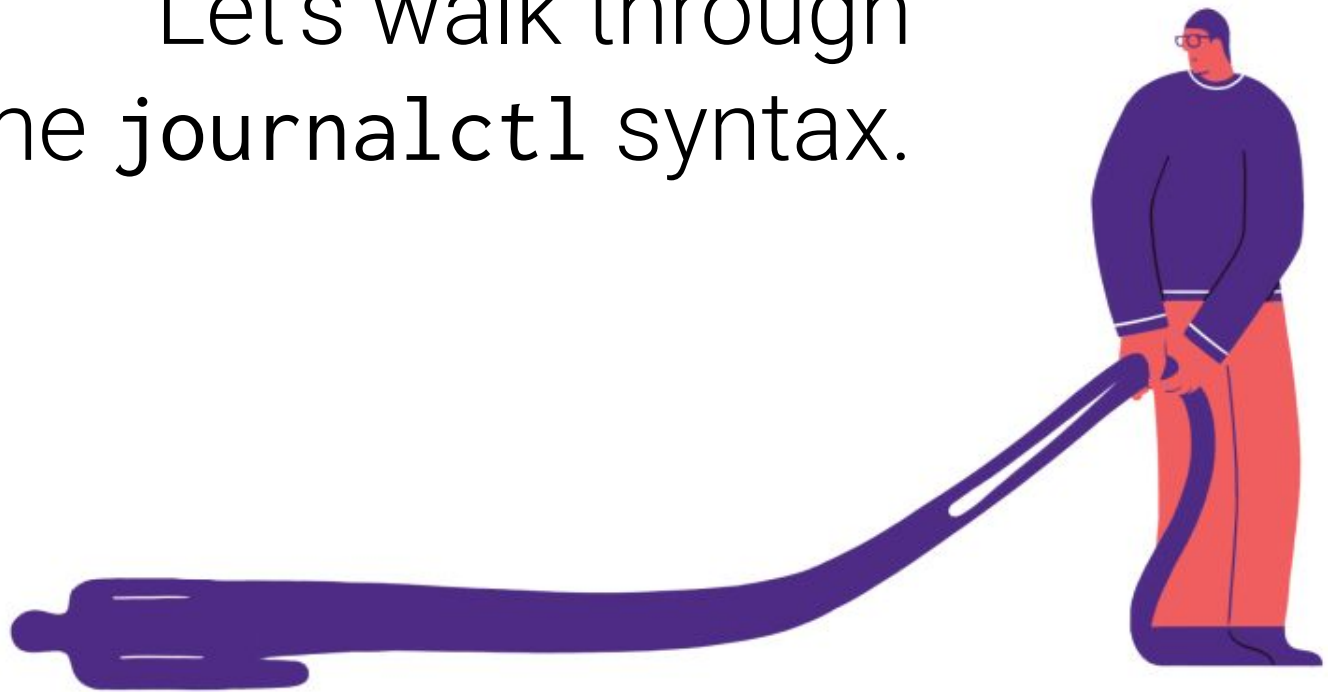
`journalctl` is designed to filter through enormous system logs and return specific results.

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 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 -b: Displays messages from specific boots.

- Boots are “anchor points” during the initial stages of incident response.
- If left empty, it will display the most recent log.

journalctl --since yesterday: Filters results starting from the time specified

journalctl --until “2 hours ago”: Filters results before the specified time.

journalctl -u cron: Filters for specified services.

`journalctl` Syntax examples

We can use the following syntax with the journalctl command:

`journalctl -p err -b -10` filters log messages based on priority message.

- `-p` is the message priority. Log messages will be filtered based on their priority level.
- `err` is the priority level being queried. Will only return results with a priority level of error and lower.
- `-b` shows messages from a specific boot, which will add a match for the boot ID.
- `-10` is the boot ID record taken from the `--list-boots` command.

Different Types of Priorities		
`emerg`	`0`	System is unstable
`alert`	`1`	Action must be taken immediately
`crit`	`2`	Critical conditions
`err`	`3`	Error conditions
`warn`	`4`	Warning conditions
`notice`	`5`	Normal but significant conditions
`info`	`6`	Informational messages
`debug`	`7`	Debug-level messages

`journalctl` Walkthrough

We can use the following syntax with the journalctl command:

`journalctl --vacuum-size=5M`: Removes archived logs until occupied disk space falls below the specified file size (5 MB). Sizes are measured in bytes:

- `K`: kilo
- `M`: mega
- `G`: giga
- `T`: tera

`journalctl --vacuum-time=2years`: Removes archived logs until the journal contains no log older than the specified time frame.

`journalctl --vacuum-files=50`: Reduces the number of journal files so they stay below a specified number.

Demo Scenario

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

- You need to investigate an unexpected system reboot at 8:07 p.m., November 8th.
- IT admin asked you to enable **log persistence** to avoid future loss of log data.
- They also advised you to increase log buffer size to 10 GB to account for sharp spikes in the log buffer and avoid potential DoS attacks.
- The security manager also wants you to implement a log management scheme. It requires the removal of archived log journals older than one year, except the most recent 10, in order to free up disk space.

Demo Steps

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

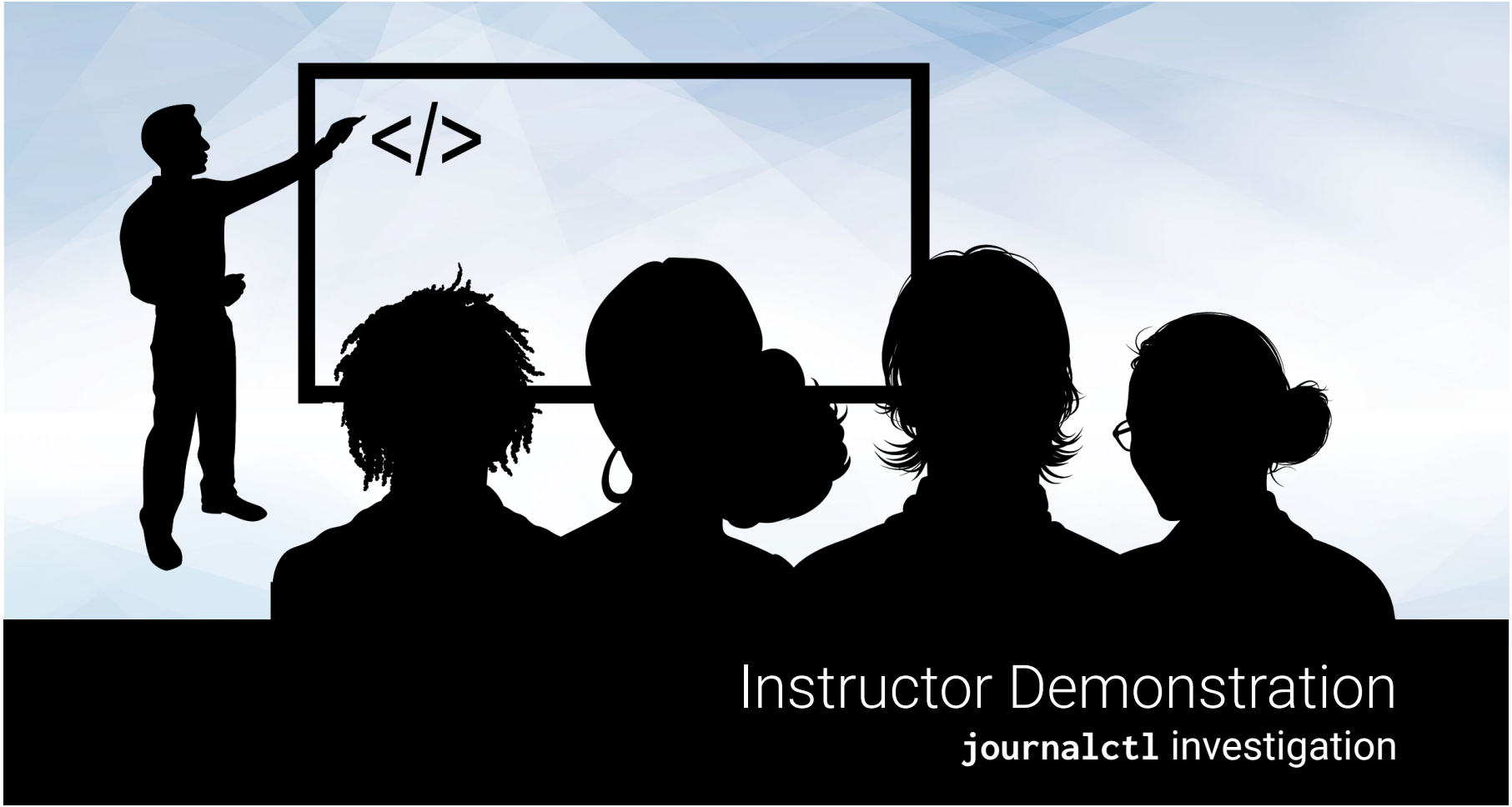
01 Use **`journalctl`** to query the **`systemd`** journal.

02 Apply a filter to **`journalctl`** that returns a list of system boots.

03 Use **`journalctl`** command options to extract boot events from the system log.

04 Configure **`journalctl`** to persist log data.

05 Configure disk usage to remove all archived log journals except the most recent 10, remove any older than one year, and increase log buffer size to 10 GB.



Instructor Demonstration

`journalctl` investigation

rsyslog

Limitations of `syslog`

While **`journalctl`** provides the security administrator the capability to filter through massive logs based on specific search criteria, it also has its limitations:

- **Cannot create new logs.**
- **Does not set priorities for which alerts should be logged.**

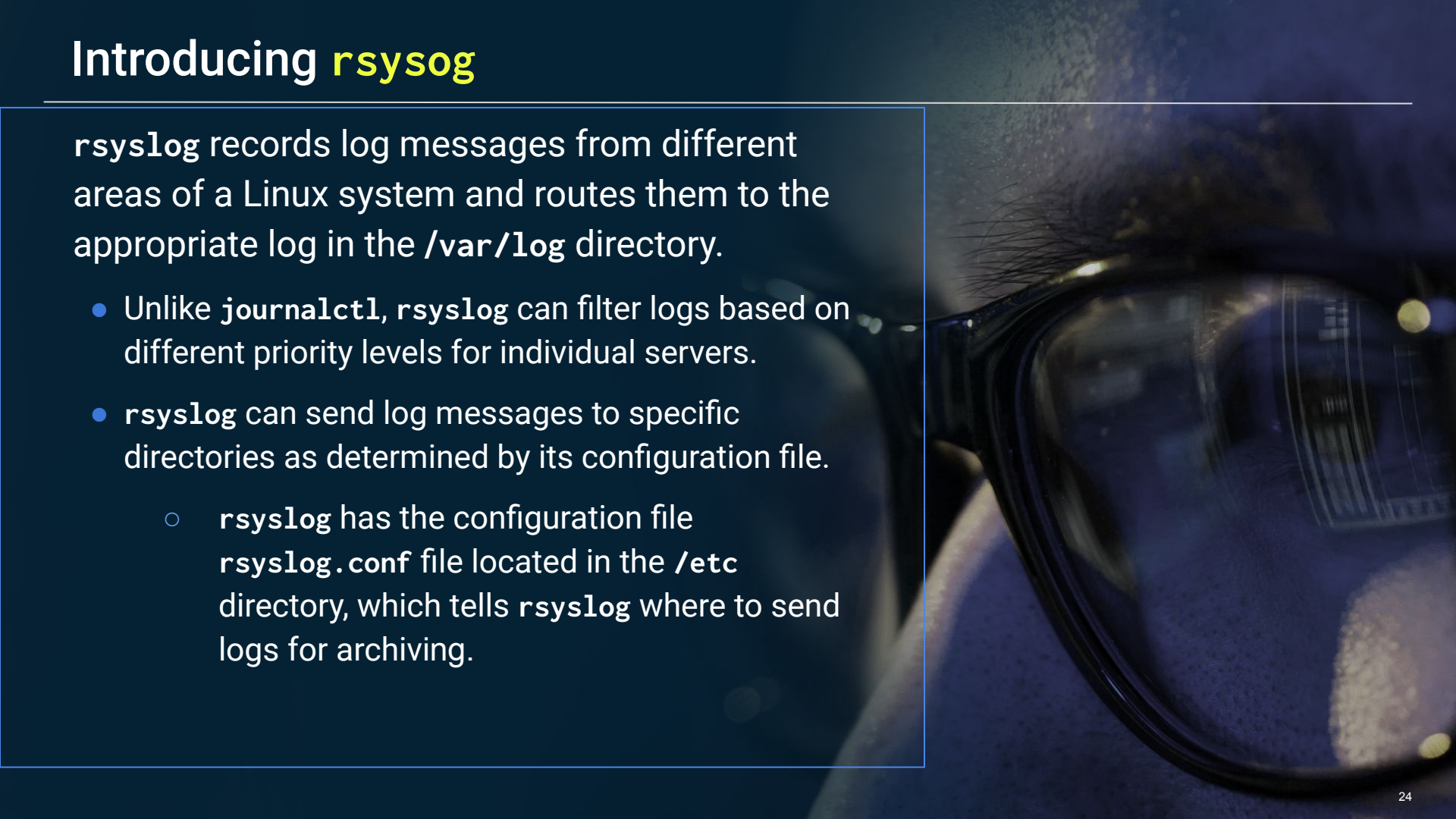
We can use **`rsyslog`** to do things that **`journalctl`** cannot. We can use them together to form a more cohesive log management system.

`rsyslog` = remote system log, and can be used to gather logs from different machines on a network, and store them on a central machine

Introducing rsysog

rsyslog records log messages from different areas of a Linux system and routes them to the appropriate log in the **/var/log** directory.

- Unlike **journalctl**, **rsyslog** can filter logs based on different priority levels for individual servers.
- **rsyslog** can send log messages to specific directories as determined by its configuration file.
    - **rsyslog** has the configuration file **rsyslog.conf** file located in the **/etc** directory, which tells **rsyslog** where to send logs for archiving.



Example `rsyslog` Configuration File

```
#  /etc/rsyslog.conf    Configuration file for rsyslog.
#
#           For more information see
#           /usr/share/doc/rsyslog-doc/html/rsyslog_conf.html
...
...
auth,authpriv.*         /var/log/auth.log
*.*;auth,authpriv.none      -/var/log/syslog
#cron.*             /var/log/cron.log
daemon.*            -/var/log/daemon.log
kern.*              -/var/log/kern.log
lpr.*               -/var/log/lpr.log
mail.*              -/var/log/mail.log
user.*              -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info           -/var/log/mail.info
mail.warn           -/var/log/mail.warn
  mail.err            /var/log/mail.err
```

`rsyslog` Syntax

Configuration logs are made up of two-part commands:

```
sshd.emerg                    /var/log/sshd.log
```

The **selector** indicates where messages are from and priority of the log message.

- **Facility**, located to the left of the dot, indicates the origin of the message.
- **Priority**, located to the right of the dot, indicates the severity of the message.
- In this case, the command is telling `rsyslog` to log messages from `sshd` that are of the `emerg` priority and lower.

The **action** indicates where the message is going.

- In this case, the message will be sent to `/var/log/sshd.log`.

`rsyslog` Syntax — Selector

```
cron.*                          var/log/cron.log
```

`cron` is the facility.

- The facility tells **`rsyslog`** to record the message generated by **`cron`**.

The asterisk (**`*`**) is the priority.

- The asterisk specifies that **all** message priorities will be logged.

rsyslog Syntax — Selector

Facility Types	
auth	Security/Authorization messages
kern	Kernel messages
mail	Mail system messages
cron	Clock daemon related messages
daemon	System daemon messages
lpr	Printing-related messages
user	User-level messages
security	Security/authorization related messages

Priority Types	
emerg	System is unstable
alert	Action must be taken immediately
crit	Critical conditions
err	Error conditions
warn	Warning condition
notice	Normal but significant condition
info	Informational messages
debug	Debug-level messages

`rsyslog` Syntax — Selector

Facilities and priorities can be related in the following ways:

```
mail.warn /var/log/mail.log
```

`.warn` priority tells `rsyslog` to log any messages with a `warn` priority or lower.

```
auth.!info /var/log/mail.log
```

`!info` tells `rsyslog` to **not** log informational messages.

```
auth.!=info /var/log/mail.log
```

`!=info` tells `rsyslog` to log all messages above the `info` priority.

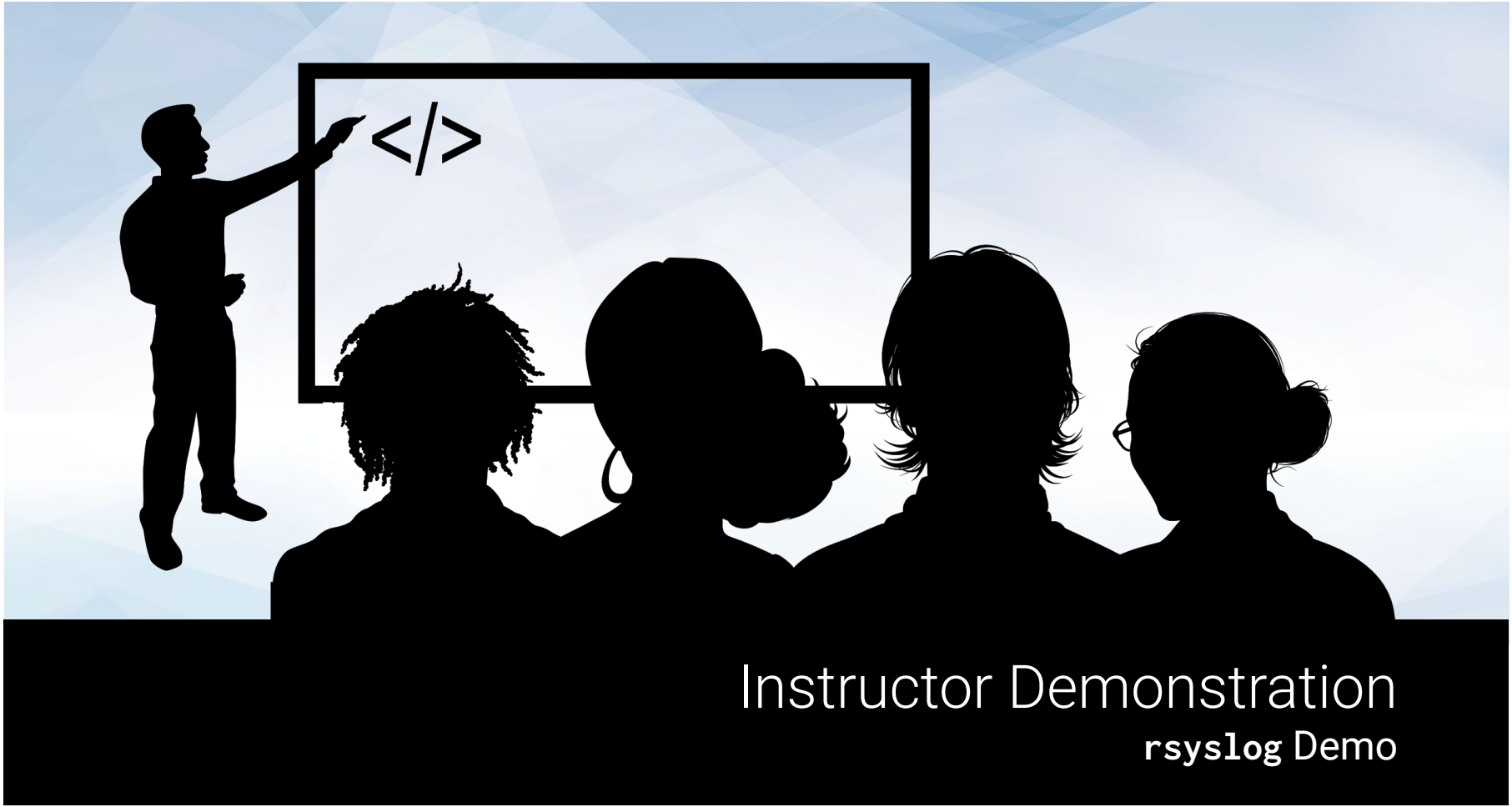
Demo Scenario

In the next demo, we will configure a fully functional `rsyslog` service that will:

- Record journal messages based on a specific priority for the **`lpr`** daemon.
- Record journal messages to the **`/var/log/lpr.log`** directory.

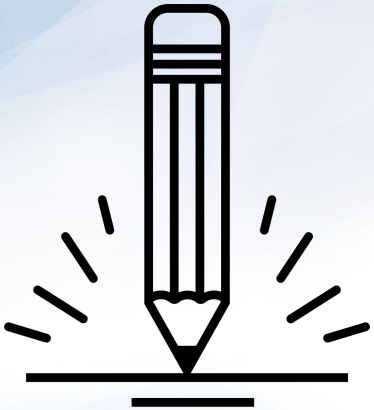
To achieve this, we will complete the following steps:

1. Verify the **`rsyslog`** installation
2. Configure **`rsyslog`** to indicate the specific directories to save log data.
3. Assign priorities to alerts before they are logged.
4. Restart the **`syslog`** daemon to activate configuration changes.



Instructor Demonstration

`rsyslog` Demo



Activity: Log Filtering

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

Activity file shared by the instructor.

Suggested Time:
15 Minutes

Time’s Up! Let’s Review.

Activity Review: Log Filtering

Completing this activity required the following steps:

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

02 Apply filters to **`journalctl`** queries.

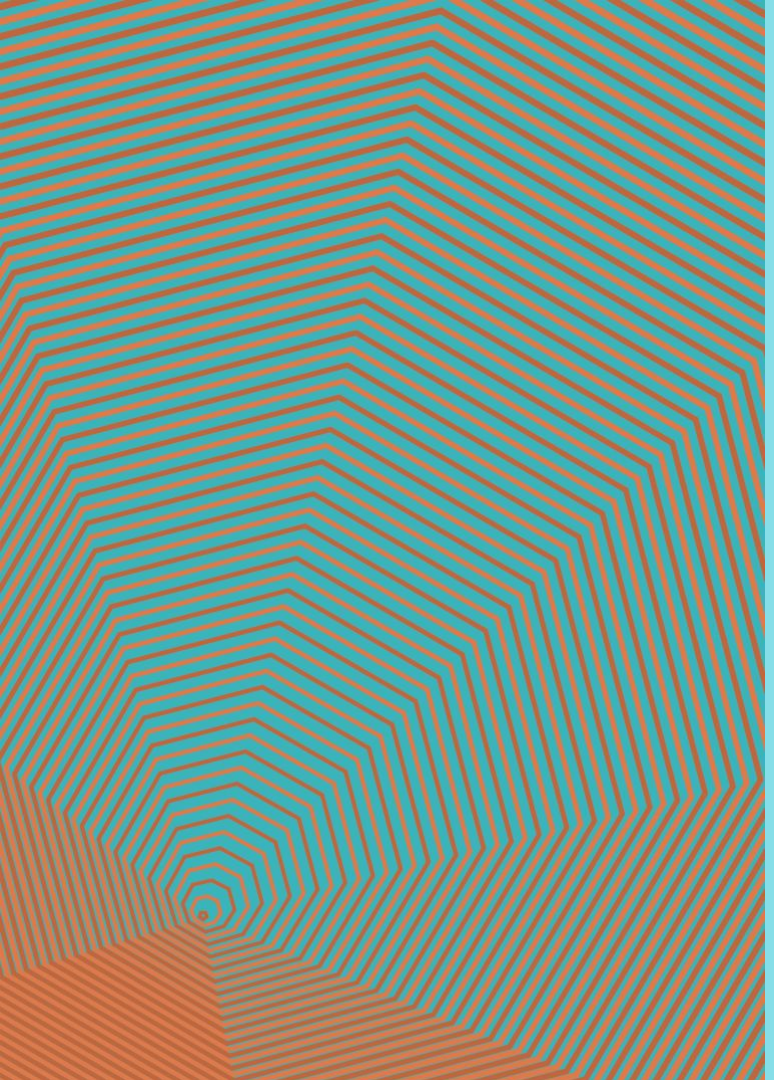
03 Use **`rsyslog`** to specify directories to save log data to.

04 Start and stop the **`rsyslog`** daemon.

05 Edit the **`rsyslog`** config file to create new log files.

06 Apply filters to **`rsyslog`**.

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.

Lograte Configurations

Logrotate uses the path **`etc/logrotate.conf`**, which is a file that contains configuration settings such as: options, parameters, and specifications of which logs to rotate.

```
/var/log/apache2/*.log {
  daily
  missingok
  rotate 14
  compress
  delaycompress
  notifempty
  endscript
 }
```

This configuration determines which actions will be logged and the location of these logs.

- All logs (`*.log`) in the `/var/log/apache2` directory.

Everything contained in the braces `{}` are configuration options, telling Logrotate what actions to take during rotations.

Logrotate Configurations

```
/var/log/apache2/*.log {
  daily
  missingok
  rotate 14
  compress
  delaycompress
  notifempty
  endscript
 }
```

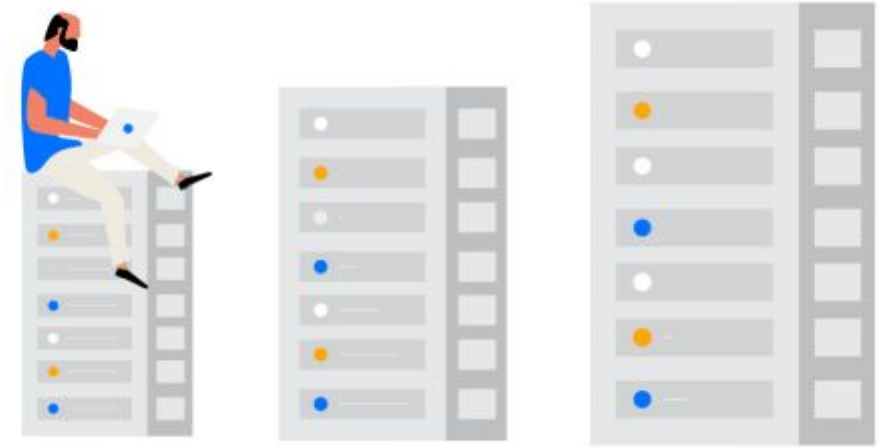
- **`daily`**: Frequency that existing logs logs are rotated out.
- **`missingok`**: If the log file is missing, go on to the next one without issuing an error message.
- **`rotate`**: Specifies number of rotations before a log is removed or emailed.
    - When set to **`0`**, old versions of logs are removed.
- **`compress`**: gzip is default compression used for log files, but other types may be used.
- **`delaycompress`**: Postpone compression of previous log file to next rotation cycle.
- **`notifyempty`**: Do not rotate the log if it is empty (overrides **`ifempty`** option).

Logrotate Demo Scenario

We will use Logrotate with the following scenario:

Every four years, the IT administrator transfers log files to a remote server, resulting in existing logs growing to an unmanageable size.

We must implement a log size management process that will:



For `mail.log`:

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

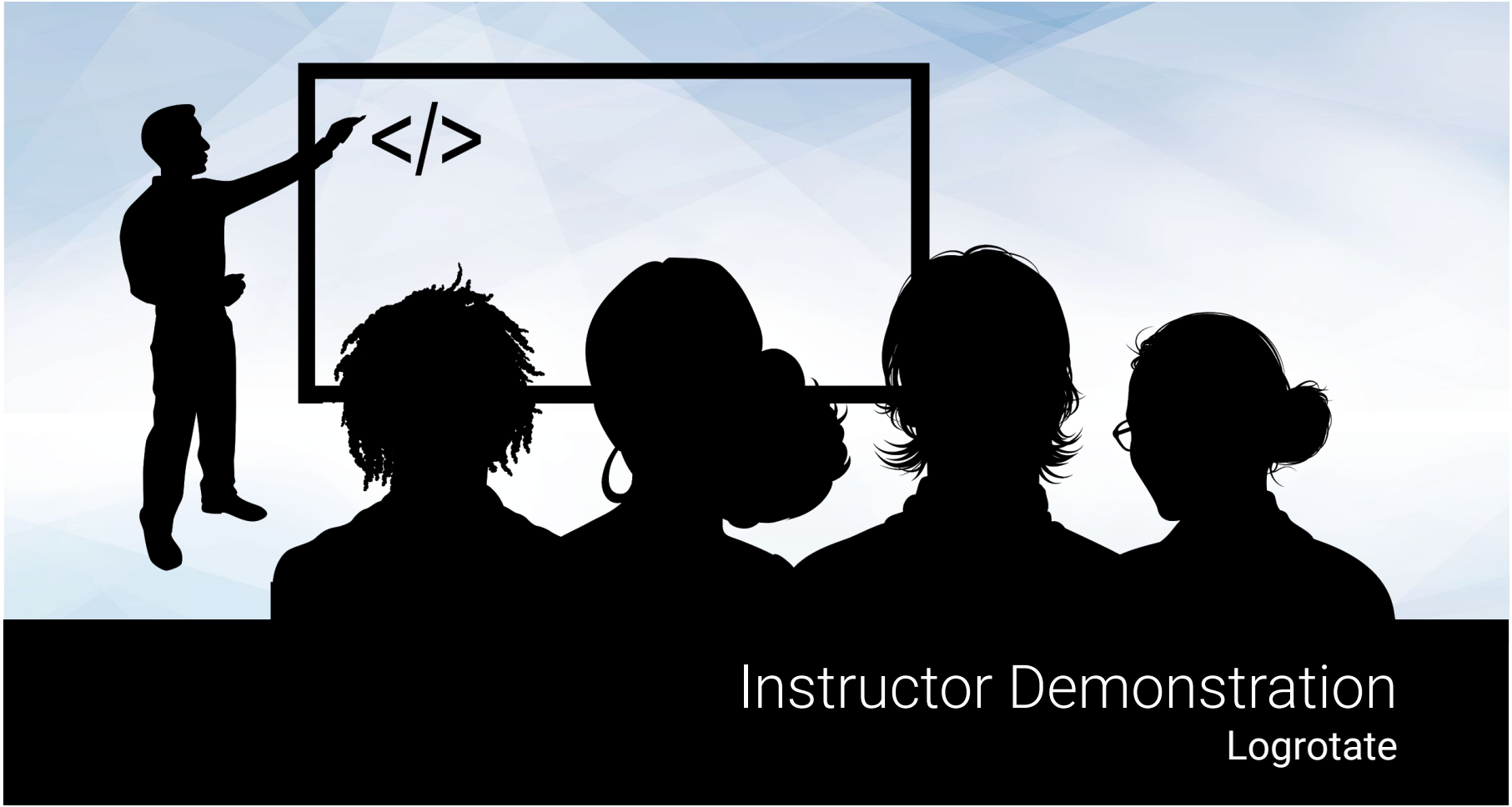
For `dmesg log`:

- Keep 10 weeks of backlogs.
- Rotate logs weekly.
- Create new empty logs after rotating out the old ones.
- Compress logs before moving them to save disk space.

Logrotate Demo Scenario

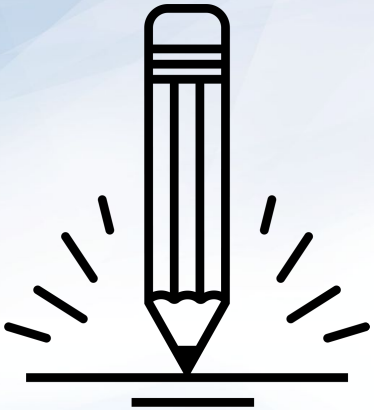
In order to complete this task, we will:

1. List directories for `logrotate.d` to display default configuration files.
    a. If configuration file exists, edit it.
    b. If configuration file does not exist, add it to `/etc/logratate.conf`.
2. Configure by editing the `/etc/logrotate.conf` file with the following settings:
    a. `rotate` to keep the most recent eight weeks of backlogs.
    b. `create` to create a new log every time the old log is rotated.
    c. `notifyempy` to avoid rotating empty logs.
    d. `compress` to compress logs during rotation.
3. Test the configuration changes with 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:
15 minutes

Time’s Up! Let’s Review.

Activity Review: Log Size Management

Completing this activity required the following steps:

01 List the contents of the **`logrotate.d`** file to display a list of Logrotate configurations for any installed package that needs log rotation.

02 Check the installed version of Logrotate for version control purposes.

03 Edit the **`/etc/logrotate.conf`** file and modify the system configuration.

04 Test the Logrotate scheme by forcing a manual rotation.

Let's Recap

So far, we've covered the following tools of proper log management:

- **`journalctl`** filters through massive logs by narrowing search results to specific search criteria.
- **`rsyslog`** performs priority-based log filtering and creates new logs for new services via a configuration file.
- **`logrotate`** manages log file sizes to make managing logs easier, saving disk space and decreasing file transfer times.


15:00
Break

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 TTPs 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.

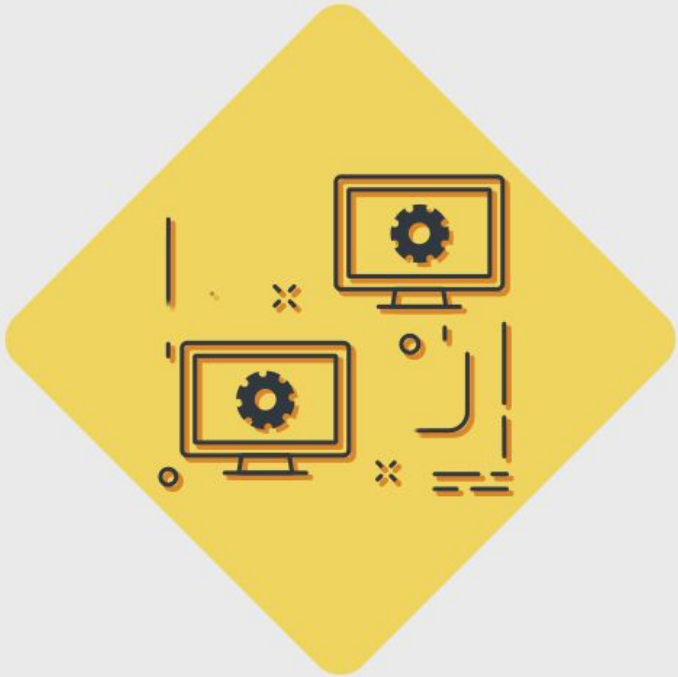
`auditd` Overview

`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.

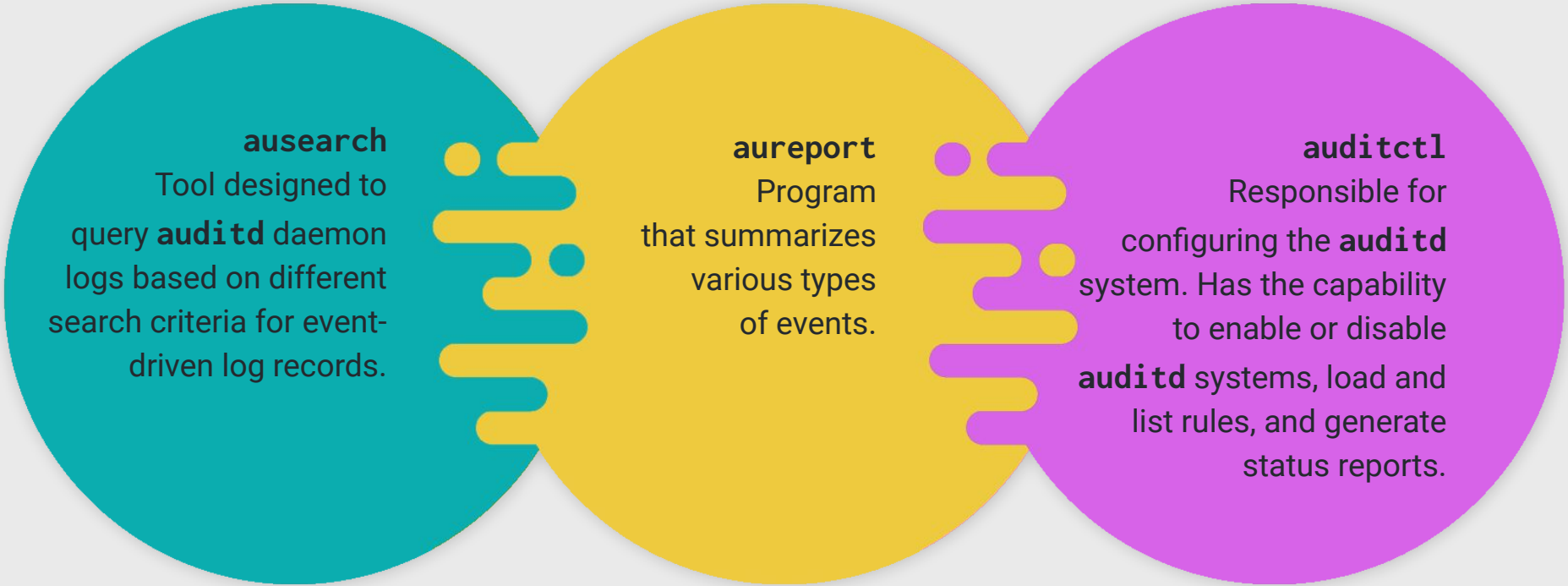
`auditd` integration with the system kernel allows it to monitor all system operations, such as network traffic and file system access.

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



auditd

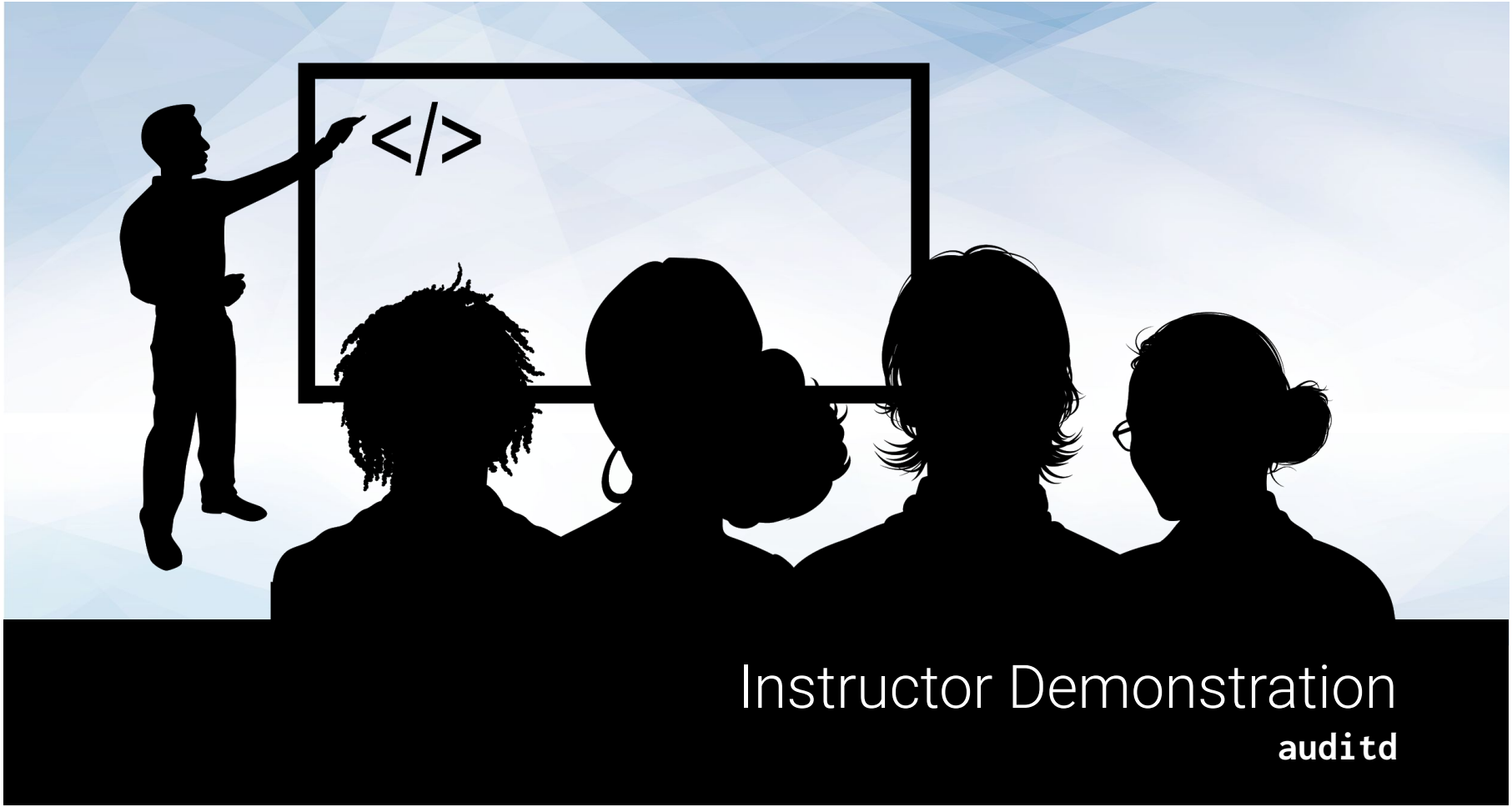
Once an event is written to disk, reporting tools such as **ausearch**, **aureport**, and **aulast** are used to generate reports.



`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.



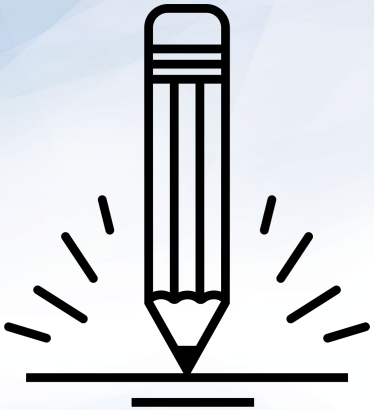
Instructor Demonstration

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:
20 Minutes

Time’s Up! Let’s Review.

Activity Review: Event Monitor Log

Completing this activity required the following steps:

01. Use the **`apt`** package manager to install **`auditd`**.
02. Edit **`/etc/audit/audit.conf`** file and make modifications as the root user.
03. Use **`auditctl`** using the **`-l`** option to list existing rule sets.
04. Edit **`/etc/audit/rules.d/audit.rules`** to add new rules.
05. Use **`auditd`** with the **`-w`** option to audit directories.
06. Perform log searches using **`auditd`** with the **`-au`** option.
07. Test **`auditd`** by creating a user account using **`useradd`**.
08. Create a report for modifications of **`auditd`** using the **`-m`** option.

Any Questions?