LINUX SECURITY

part_3.sh

Author: aimery de La Roncière @ aimerydlr
University: ECE Paris – Ing5 CYB
YEAR: 2025-2026

Table of contents

I. Audit and Monitoring

Effective audit and monitoring are critical components of a robust security posture. They provide visibility into system activities, enable the detection of suspicious behaviors, and support forensic investigations when incidents occur. This chapter explores key aspects of audit and monitoring focusing on user login monitoring, the comprehensive handling of logs and events, and the role of advanced tools like Security Information and Event Management (SIEM) systems.

I.1. User Login Monitoring

Monitoring user login activities is essential for security auditing and detecting unauthorized access attempts. Linux provides several command-line tools to inspect login records, user sessions, and authentication failures.

w

Shows who is currently logged in and what they are doing, including the login time, idle time, and current processes.

Important files:

- /var/run/utmp : Stores information about current logins.

```
w           # Display all current logged-in users and their activity
w username  # Show what the specified user is doing
w -h        # Show the same output without the header
```

who

Lists users currently logged in, with details such as terminal, login time, and originating IP address.

Important files:

- /var/run/utmp : Tracks logged-in users.

```
who                 # List all logged-in users
who -a              # Show all available information (including dead processes)
who am i            # Show information about the current terminal session
```


last

Displays the history of user logins, reboots, and shutdowns from the system log.

Important files:

- /var/log/wtmp : Binary file storing all login and logout events.

```
last                    # Show the most recent logins and system reboots
last -n 10              # Show the last 10 login/logout records
last -f /var/log/wtmp.1 # Read from an archived wtmp file
```

lastlog

Shows the most recent login time of all users or a specified user.

Important files:

- /var/log/lastlog — Binary file containing last login times.

```
lastlog               # Display last login times for all users
lastlog -u username   # Show last login time of a specific user
lastlog -t 30         # Show users who haven't logged in for 30 days or more
```


faillog

Reports failed login attempts and can reset the failure count.

Important files:

- /var/log/faillog — Stores failed login attempt records.

```
faillog                # Show failed login attempts for all users
faillog -u username    # Show failed login attempts for a specific user
faillog -r             # Reset all failed login counters
```

faillock

Manages authentication failure records and can lock accounts after a number of failed attempts.

Important files:

- /var/run/faillock/ — Directory storing per-user failure records.

```
faillock --user username           # Display failure record for a user
faillock --user username --reset   # Reset failure counter for a user
faillock --reset                   # Reset all failure counters
```


I.2. Logging, Analysis, Auditing, Collection, and forwarding of Events

Centralized logging and auditing are key components for system monitoring, troubleshooting, and security. Linux provides several protocols and tools to collect, store, analyze, and forward logs generated by the kernel, services, and applications.

System Logging Protocols and Daemons

journald (systemd journal)

journald is the native systemd logging daemon, designed to collect structured, indexed logs from the kernel, system services, and user applications. Unlike traditional syslog, it stores logs in a binary format enabling efficient filtering, querying, and integrity verification.

- Key Features:
  - Structured logs with metadata (e.g., PID, UID, SELinux context)
  - Supports both persistent (disk) and volatile (RAM) storage
  - Integration with systemd and coredump
    Logs stored in binary files (not plain text)
- Important files/directories:
  - */var/log/journal/* : persistent journal files; created only if persistent storage enabled)
  - */run/log/journal/* : volatile journal files in RAM; logs lost at reboot)
  - */etc/systemd/journald.conf* : configuration file

- Example configuration (/etc/systemd/journald.conf):

```
[Journal]
Storage=persistent          # Enables persistent log storage on disk
Compress=yes                # Compress archived journal files
MaxRetentionSec=1month      # Retain logs for one month
RateLimitIntervalSec=30     # Throttle log flood to 1 message per 30s
ForwardToSyslog=yes         # Forward journal logs to traditional syslog daemon
```


journalctl

journalctl is a command-line tool to view and query logs managed by systemd's journald. It uses a binary format for fast searching and filtering by service, time, or priority, providing a unified way to analyze system and application events.

```
journalctl                                          # Show all logs from all boots
journalctl -u sshd.service                          # Show logs only from SSH daemon
journalctl -b                                       # Show logs from current boot
journalctl --since "2025-07-01" --until "2025-07-02"   # Logs in date range
journalctl -p err                                   # Show only error and more critical logs
```

Syslog Protocol and Daemons

Syslog is a long-standing standard protocol (RFC 5424) for transmitting event messages, originally over port 514. It enables logging messages from different devices and applications in a centralized fashion.

- Common syslog daemons:
  - *syslogd*: Basic legacy syslog daemon, limited features, rarely used now.
  - *rsyslog*: Advanced and widely-used syslog daemon with modular inputs/outputs, supports TCP/UDP, TLS encryption, database storage.
  - *syslog-ng*: Alternative syslog implementation offering extensive filtering, rewriting, and transport options.

- Typical files and folders:
  - */etc/rsyslog.conf* and */etc/rsyslog.d/* : rsyslog configs
  - */var/log/* : default directory for syslog logs, e.g., /var/log/syslog, /var/log/auth.log
  - */etc/syslog-ng/* : syslog-ng configuration folder

- rsyslog Configuration

It is recommended not to modify the main configuration file /etc/rsyslog.conf directly. Instead, configuration changes should be made by creating separate files within the /etc/rsyslog.d/ directory. This approach improves maintainability and avoids conflicts, as it allows modular management of rsyslog settings, making upgrades and troubleshooting easier.

- Emitter configurations (send logs to remote syslog server)

```
# /etc/rsyslog.d/90-remote.conf
# Legacy syntax

*.* @192.168.1.100:514          # Send all logs over UDP to remote server IP 192.168.1.100


#/etc/rsyslog.d/30-audit.conf
# RainerScript syntax

# Load the imfile module to monitor specific log files with a polling interval of 10 seconds
module(load="imfile" PollingInterval="10")

# Configure an input to monitor the auditd audit.log file
input(type="imfile"
      File="/var/log/audit/audit.log"    # Path of the file to monitor
      Tag="audit:"                       # Prefix added to logs for identification
      Facility="local6"                  # Syslog facility used for these logs
      Severity="debug")                  # Severity level of the logs

# Forward logs with facility local6 to a remote server using TCP on port 514
local6.* @@<remote_ip>:514
```


- Receiver configuration (listen on UDP port 514 and save logs):

```
# /etc/rsyslog.d/90-incoming.conf

# Load the TCP module
module(load="imtcp")

# Enable listening on TCP port 514
input(type="imtcp" port="514")

# Template for remote logs storage path
template(name="RemoteLogs" type="string" string="/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log")

# Action to write logs to local files based on the template
action(
  type="omfile"
  dynaFile="RemoteLogs"
  createDirs="on"
  dirCreateMode="0755"
  fileCreateMode="0640"
)

# Prevent other rules from processing this message further
stop
```

- Configuration syntax

A syntax in rsyslog defines the set of rules and format used to write configuration files that control how logs are processed, filtered, and forwarded.

- The *Legacy syntax* is the original, simpler format that uses straightforward directives and statements. It is easy to read and configure but offers limited flexibility for complex logging needs.
- The *RainerScript syntax* is a more recent, powerful scripting language designed for rsyslog. It provides advanced features such as conditional statements, loops, variables, and enhanced control over log processing. This makes RainerScript the preferred choice for modern and complex rsyslog configurations.

For more information, see the official documentation:

- Legacy syntax: https://www.rsyslog.com/doc/v8-stable/tutorials/legacy_syntax.html
- RainerScript syntax: https://www.rsyslog.com/doc/configuration/index_directives.html

logger

logger is a command-line utility used to send custom log messages to the system log (syslog) or the systemd journal (journald). It allows users and scripts to manually generate log entries, which can be useful for testing, debugging, or adding custom event records.

```
logger "System maintenance started"               # Logs a simple informational message
logger -p local0.notice "Custom alert message"    # Logs a notice to local0 facility
logger -t MyScript "Script execution completed"   # Logs a message tagged with "MyScript"
```


Audit System

auditd

auditd is the Linux Audit daemon, specifically designed for security auditing. It logs detailed information about system calls, file access, user authentication events, and more.

- Important files/directories:
  - */etc/audit/auditd.conf* : daemon configuration
  - */etc/audit/audit.rules* : audit rules specifying what to log
  - */var/log/audit/audit.log* : main audit log file
  - */etc/audit/rules.d/* : directory for modular audit rules (depending on distro)

- Example auditd configuration (/etc/audit/auditd.conf):

```
log_file = /var/log/audit/audit.log
log_format = RAW
flush = INCREMENTAL
max_log_file = 30          # Maximum log file size in MB before rotation
max_log_file_action = ROTATE
space_left = 75            # Warning threshold for disk space (in MB)
action_mail_acct = root    # Send mail to root when disk space is low
```

- Basic audit rule example (/etc/audit/rules.d/50-audit.rules):

```
# Monitor all execve system calls
-a always,exit -F arch=b64 -S execve -k exec_commands
# Log all changes to /etc/passwd
-w /etc/passwd -p wa -k passwd_changes
# Monitor usage of sudo command
-w /usr/bin/sudo -p x -k sudo_usage
```

- Application of new audit rules:

```
augenrules --load
```


aureport

aureport is a powerful utility that reads audit logs and generates summarized reports. aureport provides aggregated data on security events, making it ideal for compliance checks and incident overviews.

It reads from:

- */var/log/audit/audit.log* : the main audit log file.
- */etc/audit/auditd.conf* : defines log rotation and storage rules.
- Uses audit events generated by auditd and kernel rules defined in */etc/audit/rules.d/*.rules*.

```
# Show a summary of all audit events (logins, exec, file access, etc.)
aureport

# Display only authentication-related events (logins, sessions)
aureport -au     # "au" for authentication

# Show executed commands (useful for incident review)
aureport -x
```


ausearch

ausearch is a command-line utility used to query and filter audit logs in a detailed and flexible way. It is ideal for forensic analysis and deep inspection of individual security events logged by the Audit subsystem.

It reads from:

- */var/log/audit/audit.log*: main audit log file.
- */etc/audit/auditd.conf*: log configuration file.

Key features:

- Search by UID, PID, SYSCALL, event type, file path, time range, or key.
- Supports complex filtering with logical conditions (e.g., ausearch -x /usr/bin/passwd -ts yesterday).
- Can output results in raw or human-readable format.

```
ausearch -x /usr/bin/passwd      # Search all events triggered by passwd command
ausearch -k failed-login         # Search events tagged with a specific audit key
ausearch -ua 1001 -ts recent     # Search by UID (1001) with recent timestamps
```


I.3. SIEM (Security Information and Event Management)

A SIEM is a centralized platform that allows organizations to collect, analyze, and correlate security events and log data from across their entire IT environment.

It plays a key role in threat detection, incident response, compliance auditing, and log management by providing unified visibility over diverse systems (servers, network devices, endpoints, applications).

It also ensures log retention over a defined period of time, which is essential not only for complying with regulatory standards such as PCI-DSS, HIPAA, or GDPR, but also for conducting forensic analysis, troubleshooting technical issues, and supporting effective incident response.

Objectives of a SIEM

- Centralized log aggregation: Collect logs from multiple sources (Linux, Windows, firewalls, IDS/IPS, antivirus, applications).
- Real-time monitoring and alerting: Identify suspicious behaviors and generate alerts on anomalies or rule-based patterns.
- Correlation of events: Combine multiple low-level events into high-level security incidents (e.g., login failure followed by privilege escalation and data access).
- Retention and compliance: Store logs securely for long durations (months/years) to meet legal and regulatory requirements (GDPR, PCI-DSS, HIPAA...).
- Incident investigation and forensics: Enable searching, filtering, and timeline reconstruction of past events.

Global Workflow of a SIEM

- Data Collection (Agents / Syslog / APIs). SIEMs ingest data through various methods, including:
  - Agents installed on endpoints (e.g., Wazuh agent, NXLog, Beats) to forward logs securely and in real-time.
  - Syslog (UDP/TCP, RFC 5424/3164), a common protocol used by Unix/Linux systems and network devices.
  - API integrations with cloud services and SaaS platforms (e.g., AWS CloudTrail, Microsoft 365).
  - Log shipping tools such as Filebeat, Fluentd, or Logstash for structured parsing and transport.
  - While SNMP is primarily used for network monitoring, its traps can also be forwarded to a SIEM for alert correlation, although it is less commonly used for detailed log ingestion.

- Normalization and Parsing: All incoming data is transformed into a structured format (JSON, key-value, etc.) to enable correlation across sources.
- Enrichment (optional): SIEMs may add context to raw logs, such as geolocation, threat intelligence, or user identity data.
- Correlation Engine. Rules or machine learning models detect abnormal patterns, such as:
  - Multiple failed logins from different locations
  - Access to sensitive files outside of business hours
- Alerting and Notifications: Alerts are triggered and sent to security teams (email, dashboards, ticketing systems).
- Storage and Indexing: All events are indexed and stored in a queryable format (e.g., Elasticsearch, SQL, proprietary DB).
- Dashboards and Reports: Provide visual insights and automated reports for both operational and compliance use.

SIEM examples

Wazuh

An open-source security platform that integrates host-based intrusion detection, log collection, file integrity monitoring, vulnerability detection, and compliance reporting. Wazuh agents installed on endpoints collect security data and forward it to a central server for analysis. It provides real-time threat detection and can be integrated with Elastic Stack or OpenSearch for indexing, visualization, and alerting.

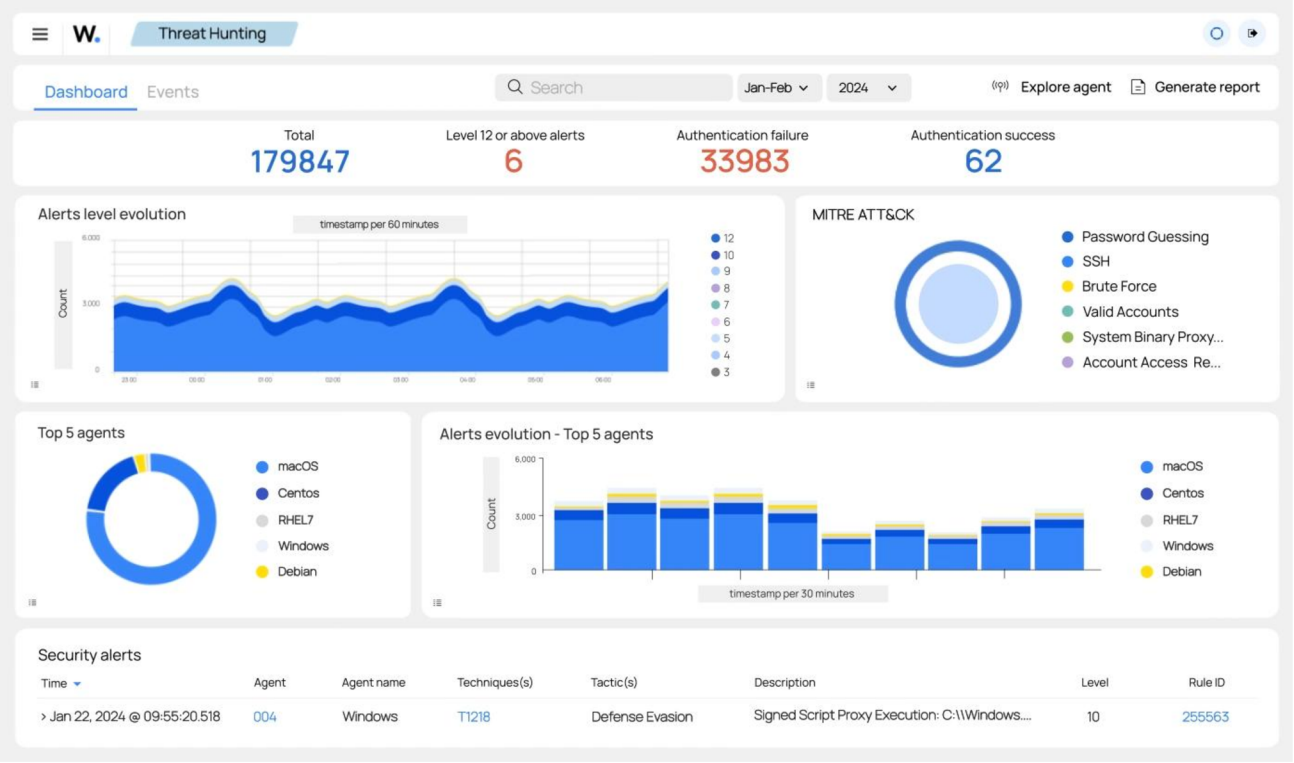

Figure 1 - Wazuh SIEM

Source: https://wazuh.com/

Graylog

A powerful log management and analysis platform that supports real-time log ingestion, structured search, and customizable dashboards. Graylog can collect logs from various sources using protocols like Syslog, GELF, and Beats. Its correlation and alerting features make it well-suited for both security monitoring and operational troubleshooting. It is often chosen for its ease of use and scalability in managing large volumes of log data.

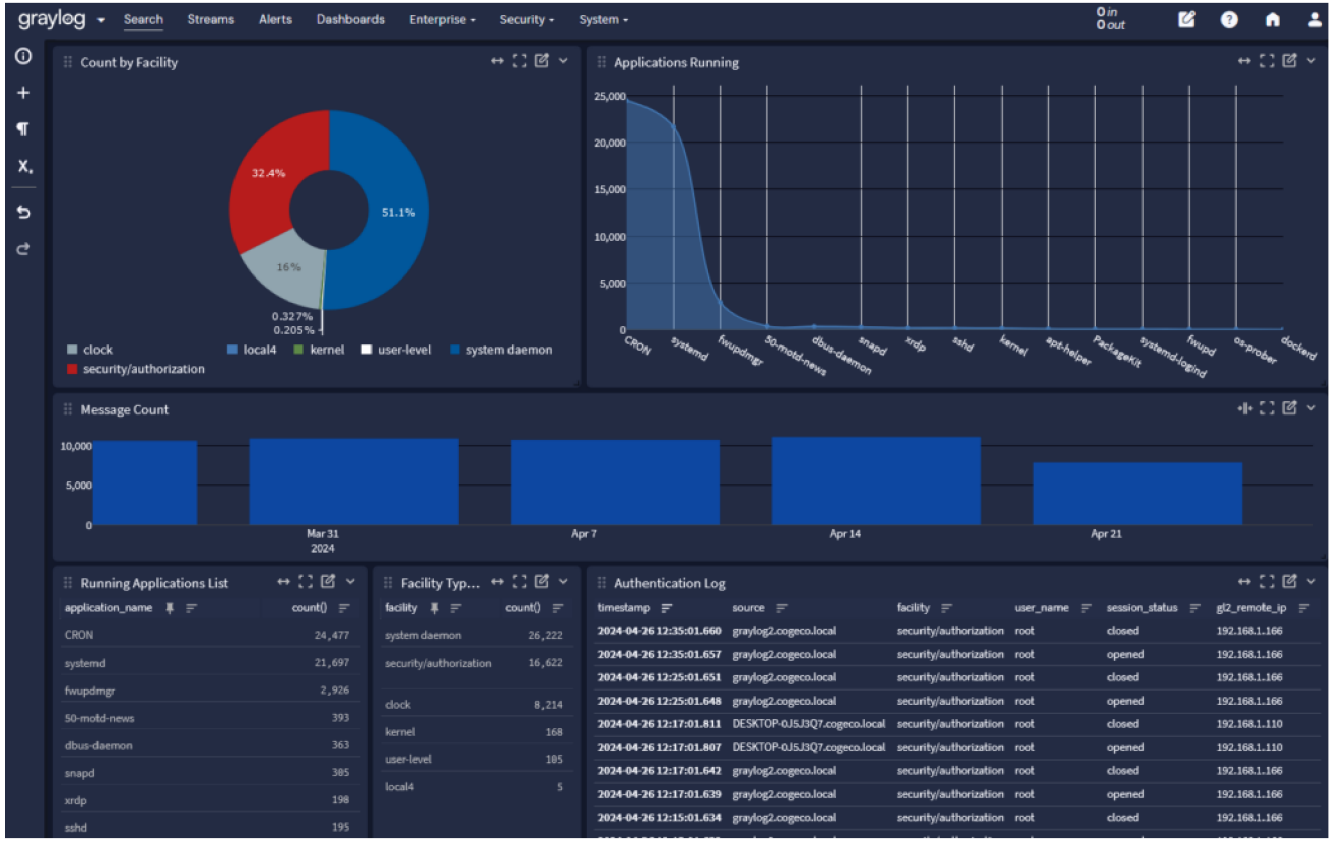

Figure 2 - Graylog SIEM

Source: https://graylog.org/

Splunk

A commercial SIEM and data analytics platform known for its powerful search capabilities and scalability. Splunk collects and indexes machine data from virtually any source, enabling advanced correlation, real-time alerts, dashboards, and automated response. Widely adopted in enterprise environments, Splunk supports ingestion via agents (like the Universal Forwarder), Syslog, APIs, and cloud connectors, making it suitable for complex, heterogeneous infrastructures.

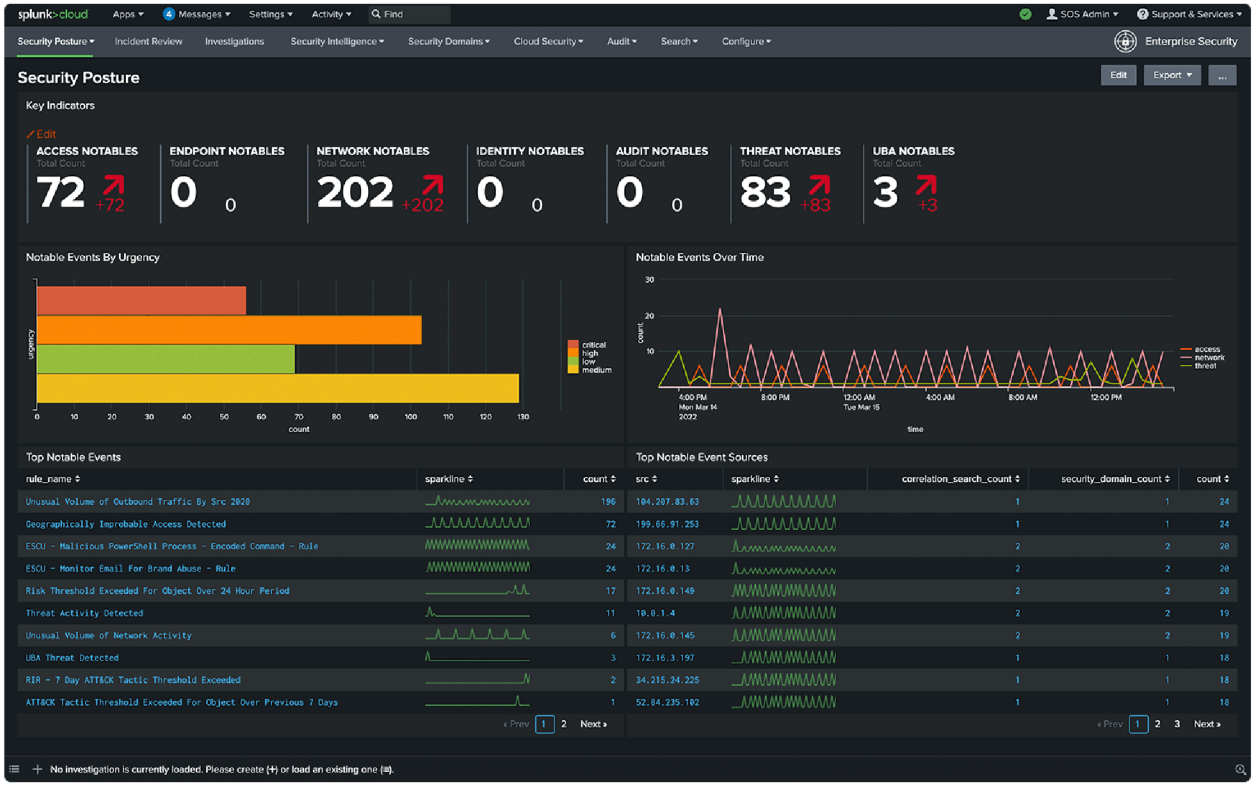

Figure 3 - Splunk SIEM

Source: https://www.splunk.com/