Security Operations Center (SOC)

Tasks:1

1. SOC Fundamentals and Operations :-

Purpose of a SOC :-

The primary purpose of a SOC is to protect an organization's IT environment by ensuring security threats are identified, analyzed, and handled effectively.

- **Proactive Threat Detection :-**
  - Continuously identify potential security threats before they cause damage.
  - Use SIEM, threat intelligence, and behavioral analysis to detect anomalies.
  - Reduce the attack surface by early identification of malicious activity.
- **incident response :-**
  - Quickly investigate and confirm security incidents.
  - Contain, eradicate, and recover from attacks.
  - Minimize business impact and data loss.
- **continuous monitoring :-**
  - 24/7 monitoring of networks, systems, applications, and logs.
  - Correlate events from multiple sources to gain full visibility.
  - Ensure ongoing security posture awareness.

- **Roles :-**

A Security Operations Center (SOC) is structured into roles to ensure efficient detection, investigation, and response to security incidents**:**

- **Tier 1 – SOC Analyst (L1)**
  - Monitor SIEM alerts and dashboards
  - Perform initial alert triage
  - Identify false positives vs true positives
  - Escalate confirmed threats to Tier 2
- **Tier 2 – SOC Analyst (L2)**
  - Conduct in-depth investigations
  - Analyze logs, network traffic, and endpoints
  - Identify root cause and scope of incidents
  - Recommend containment actions
- **Tier 3 – SOC Analyst (L3/Incident Responder)**
  - Handle advanced and high-severity incidents
  - Perform malware analysis and threat hunting
  - Lead containment, eradication, and recovery
  - Develop detection rules and response playbooks

- **Threat Hunters**
  - Proactively search for hidden or advanced threats
  - Use hypotheses and behavioral analytics
  - Identify new attack patterns and improve detections
  - Reduce dwell time of attackers
- **SOC Manager**
  - Oversee SOC operations and strategy
  - Manage analysts, workflows, and escalations
  - Ensure compliance and reporting
  - Track KPIs (MTTD, MTTR) and improve SOC efficiency

- **Key functions**
- **Log Analysis :-**

Collecting and examining logs from multiple sources to detect suspicious activity.

- **Log sources include:**
  - Servers (Linux/Windows)
  - Firewalls, routers, switches
  - IDS/IPS (Snort, Suricata)
  - EDR tools (Defender, CrowdStrike)
  - Applications & databases
- **What analysts do**
  - Search logs for failed logins, malware indicators, unusual traffic
  - Correlate events across different systems
  - Identify patterns like brute-force attacks or lateral movement
- **Alert Triage :-**

  Reviewing and prioritizing security alerts generated by SIEM, IDS, or EDR tools.
- **Steps in alert triage:**
  - Validate – Is the alert real or false positive?
  - Prioritize – Based on severity, asset value, and impact
  - Investigate – Check logs, endpoints, IPs, hashes
  - Escalate or Close – Send to L2/L3 or mark as benign
- **Threat Intelligence Integration**

  Enriching alerts with external and internal threat data to improve detection accuracy.

  Threat intel sources:
  - IP/domain reputation feeds
  - Malware hash databases
  - CVE and vulnerability feeds
  - MITRE ATT&CK framework
- **Identifies known malicious IPs/domains**
  - Adds context to alerts
  - Speeds up investigations

Study Of Security Operations Centers Frameworks

SOC Framework : -

A SOC framework is a structured set of guidelines, best practices, processes, and standards that helps a Security Operations Center (SOC) effectively design, operate, and improve its security operations. It provides a clear approach for monitoring, detecting, analyzing, responding to, and recovering from cybersecurity incidents in a consistent and efficient manner.

- Detect attacks
- Respond to incidents
- Understand how attackers work

- **NIST Framework :-**
  NIST (National Institute of Standards and Technology) provides a framework that explains how an organization should manage and respond to cybersecurity incidents**.**
  There are five frameworks in NIST:-
  - **Identify :-** Understand and manage cybersecurity risks by knowing your systems, assets, users, and data.
    Helps prioritize protection efforts based on business impact and risk.
  - **Protect :-** Implement safeguards to prevent or limit the impact of cyber incidents.
    Includes access control, security policies, training, and protective technologies.
  - **Detect : -**Identify cybersecurity events and incidents in a timely manner.
    Uses continuous monitoring, SIEM, and detection processes to spot threats early.
  - **Respond : -**Take action to contain and mitigate the impact of a detected incident.
    Includes incident response planning, communication, and threat containment.
  - **Recover :-** Restore affected systems and services after an incident.
    Focuses on recovery planning, backups, and improving defenses to prevent recurrence.
- **NIST Incident Response Lifecycle**
  - **Definition:**
    A step-by-step process SOC teams follow during an incident.
    - **Preparation** – Set up tools and policies
    - **Detection & Analysis** – Identify and analyze attack
    - **Containment, Eradication & Recovery** – Stop and fix the attack
    - **Post-Incident** – Lessons learned
- **Example:**
  Phishing attack → detect email → block sender → reset passwords → improve email filters.

- **MITRE ATT&CK Framework** :-
  MITRE ATT&CK is a knowledge base that shows how attackers perform attacks, based on real-world incidents.

ATT&CK stands for:
Adversarial Tactics, Techniques, and Common Knowledge

Examples :-

Tactic	Technique	Simple Example
Initial Access	Phishing	Fake email steals password
Execution	PowerShell	Attacker runs malicious script
Persistence	Startup entry	Malware runs on every reboot
Credential Access	Keylogging	Steals passwords
Lateral Movement	Remote Desktop	Moves to another system

- **How SOC Analysts Use MITRE ATT&CK**
  - This process helps SOC teams **identify the attacker's tactics and the exact stage of the attack lifecycle** by mapping SIEM alerts to known techniques such as **MITRE ATT&CK**.
    Example: A PowerShell execution alert mapped to **T1059 (Command and Scripting Interpreter)** indicates an execution phase, helping analysts confirm malicious behavior instead of a normal admin task.
  - It enables **better detection rule tuning and reduced false positives** by using standardized threat models and real attack patterns.
    Example: If repeated PowerShell alerts are benign admin activity, SOC teams can refine SIEM rules to alert only when encoded commands or suspicious parameters are used.
  - It also supports **proactive threat hunting** by uncovering related malicious activities across the environment.
    Example: After detecting T1059, analysts hunt for related techniques like **credential dumping (T1003)** or **lateral movement**, identifying hidden threats before they cause damage.

- **Case Study :-**
  - **Microsoft SOC Case Study (Azure Sentinel) :-**
    A leading technology company in South Africa wanted to improve its cybersecurity because traditional tools like firewalls and antivirus weren't enough for modern threats. They needed real-time monitoring, detection, and response across their entire business network.
  - **Solution :-**
    The company partnered with a Security Operations Centre service powered by **Microsoft Sentinel**, a cloud-native SIEM (Security Information and Event Management) and SOAR (Security Orchestration, Automation, and Response) platform running on Microsoft Azure.

- **24×7 security monitoring** – The SOC watched all systems continuously.
- **Log and event collection** – Millions of security events were collected daily.
- **AI/automation** – Microsoft Sentinel used analytics to detect threats faster.
- **Alert management** – Security analysts reviewed alerts and investigated true threats.

- **Results :-**
  - The SOC consistently collected and processed hundreds of millions of security events per da
  - About 25–35 medium to high severity alerts were triggered per day and 1–5 confirmed incidents were acted upon.
  - The company could focus on real threats instead of noise, improving security posture and reducing risk.
- **Key Takeaways :-**
  - Modern SOCs use **cloud-based SIEM like Microsoft Sentinel** for visibility across all systems.
  - AI and automation reduce manual work and help prioritize critical threats.
  - SOCs operate **24×7 to catch threats in real time**.

- **IBM SOC Case Study (IBM QRadar SIEM)**

  United Family Healthcare (UFH) — a large healthcare provider with many hospitals and clinics needed stronger cybersecurity to protect **patient data** and meet various compliance requirements. Before the SOC, they **didn't have a central security view** and found it hard to detect or manage incidents effectively.
- **Solution**

  UFH chose **IBM Security QRadar SIEM** as the backbone of its SOC. QRadar collects, analyzes, and correlates logs from:
  - Network devices
  - Servers
  - Endpoints
  - Applications
  - Security tools

  All this data gives analysts a **centralized dashboard** that shows threats and priority alerts in one place.

  UFH also used additional features like:
  - **User Behavior Analytics (UBA)** to detect insider threats
  - **Network Insights** to see traffic patterns
  - **Automated reporting** for audits and compliance

- **Results**
  - After going live with the SOC, UFH could **detect and contain ransomware in ~30 minutes**, which previously took **days**.

- Event processing and incident reporting that used to take **weeks** were done in **a single day**.
- Security visibility improved dramatically — executives could see risk status instantly on dashboards.

- **Key Takeaways:-**
  - A well-implemented SOC (powered by tools like **IBM QRadar**) gives **centralized view** of security across an enterprise.
  - Automation and analytics reduce detection and response time.
  - SOCs help with **compliance reporting** and audit requirements.

- **Summary :-**
  A SOC is where cybersecurity data is monitored 24×7, alerts are investigated, and threats are responded to. In real case studies:
  - Microsoft SOC with Azure Sentinel showed how cloud SIEM helps companies handle massive log volumes and focus on real risks.
  - IBM SOC with QRadar demonstrated how centralized SIEM and automated analytics can shorten incident response and improve overall security in large enterprises like healthcare.

- **<u>Practice Workflow Simulation using Splunk Phantom</u>**

- **Step 1: Alert Detection**
  Definition:
  Security alerts are generated by SIEM or EDR tools.
  These alerts indicate suspicious or malicious activity.
  Alerts are sent to Splunk Phantom for response.
  Example:
  Phishing email or malware alert detected.

- **Step 2: Playbook Trigger**
  Definition:
  A playbook is automatically triggered based on alert type.
  It contains predefined investigation and response steps.
  This ensures a standard SOC workflow is followed.
  Example:
  Phishing playbook starts running automatically.

- **Step 3: Automated Investigation**
  Definition:
  Phantom collects alert details like IPs, URLs, and hashes.

Threat intelligence is used to analyze the indicators.
The alert is classified as malicious or benign.
Example:
URL reputation checked using VirusTotal.

- **Step 4: Automated Response**
  Definition:
  If malicious, Phantom performs response actions automatically.
  Actions reduce impact and stop further damage.
  This speeds up incident containment.
  Example:
  Block URL, isolate endpoint, disable user.

- **Step 5: Analyst Review & Closure**
  Definition:
  SOC analyst reviews automated actions and findings.
  The incident is confirmed, closed, or escalated.
  Details are documented for future reference.
  Example:
  Analyst closes phishing incident after confirmation.

- **Easy Memory Formula :-**
  Detect → Playbook → Investigate → Respond → Review.

2. Security Monitoring Basics :-

- **Security monitoring** is the continuous process of collecting, analyzing, and reviewing system, network, and application activity to detect security threats and suspicious behavior in real time.
  It helps organizations identify attacks early, reduce damage, and respond quickly to incidents.
  Security monitoring is usually performed by a Security Operations Center (SOC) using tools like SIEM, IDS/IPS, and EDR**.**

- **<u>Objectives</u> :-**
  - Detect anomalies
  - unauthorized access
  - policy violations

- **Detect anomalies** :-Detecting anomalies means identifying unusual or abnormal behavior in systems or networks compared to normal activity.These behaviors may indicate attacks, misconfigurations, or system misuse.Early anomaly detection helps prevent major security incidents.
- **Detect Unauthorized Access :-**Unauthorized access is when someone tries to access systems, data, or resources without proper permission.Security monitoring helps identify failed logins, suspicious logins, and access violations.This prevents attackers from gaining control of sensitive systems.
- **policy violations :-** Policy violations occur when users or systems break organizational security rules or guidelines.Security monitoring ensures compliance with security policies and standards.It helps reduce insider threats and compliance risks.

- **Tools :-**

- **SIEM (Security Information and Event Management)** :- A SIEM is a centralized security tool that collects, stores, and analyzes logs and events from different sources across an organization. It correlates events to detect threats, generate alerts, and support incident response. SIEM tools provide real-time monitoring, dashboards, and reports for SOC teams.
- **What SIEM Monitors:**
  - User login activity
  - Server and application logs
  - Firewall and IDS/IPS logs
  - Endpoint security events
- **Example:** SIEM detects multiple failed login attempts followed by a successful login and raises a brute-force alert.

- **Network Traffic Analyzers :-**A network traffic analyzer captures and inspects network deep packet inspection and forensic analysis.packets to understand how data flows across the network.It helps detect malicious traffic, network attacks, and performance issues.These tools are mainly used for
- **What Network Analyzers Monitor:**
  - Source and destination IP addresses
  - Protocols (TCP, UDP, HTTP, DNS)
  - Packet size and payload
  - Suspicious traffic patterns
- **Example:-**Wireshark captures traffic and shows a large number of SYN packets, indicating a possible DoS attack.

- **<u>Key Metrics</u>:**
  - False Positives: A **false positive** occurs when a security tool raises an alert for activity that is **actually normal or harmless**. It increases alert noise and wastes analyst time. Reducing false positives improves SOC efficiency.

    Example:

    A legitimate user login is flagged as suspicious.

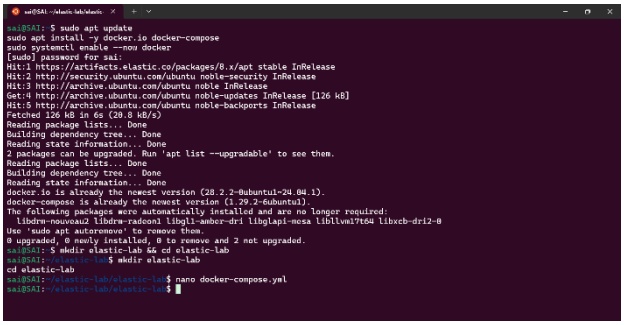
- **False Negatives:**A false negative occurs when **a** real security threat is not detected by monitoring tools.This is dangerous because attacks go unnoticed.SOC teams aim to minimize false negatives.
  Example:
  Malware infection not detected by SIEM or EDR.

- **Mean Time to Detect (MTTD)**
  Definition:-MTTD measures the average time taken to identify a security incident after it occurs.Lower MTTD means faster threat detection.It reflects the effectiveness of monitoring tools and SOC processes.
- Example:
  Attack happens at 10:00 AM, detected at 10:10 AM → MTTD = 10 minutes.

- **Set up a lab environment with open-source tools :-**
  - **Components used :-**
    - **Elasticsearch** – Data storage & search engine
    - **Kibana** – SIEM UI (Elastic Security)
    - **Filebeat** – Log shipper
    - **Sample logs / attack simulation** – To generate security events

```
Logs → Filebeat → Elasticsearch → Kibana (SIEM)
```

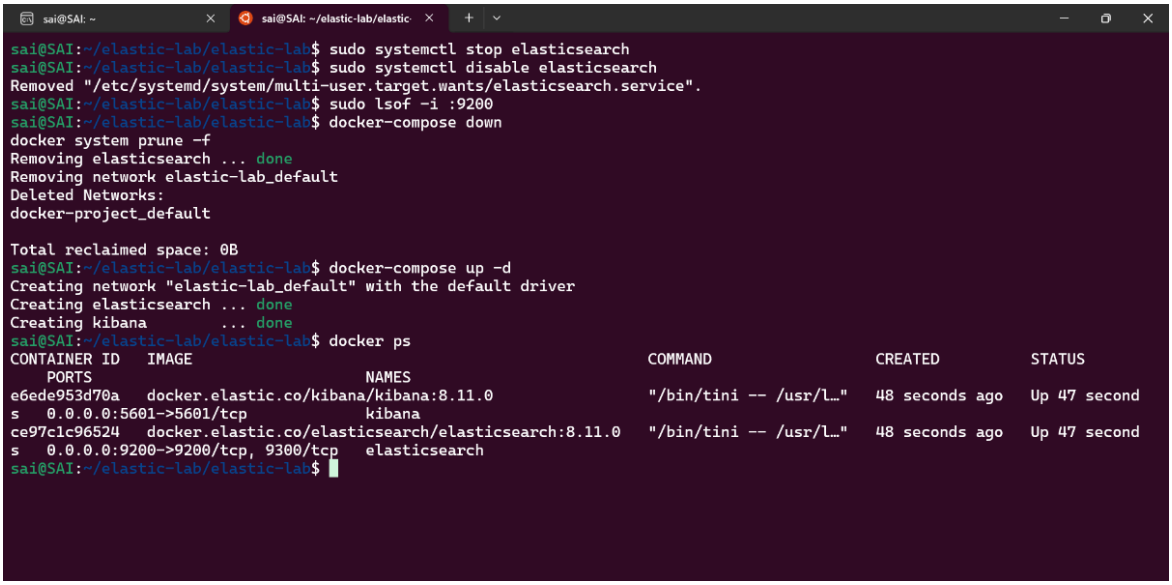
- **Prerequisites**
  - **System Requirements**
    - OS: Ubuntu 20.04+ / Windows + WSL / macOS
    - RAM: **8 GB minimum** (very important)
    - Disk: 20 GB free



Step 2 :-

Command that is used to create :-

```
version: "3.8"
 services:
 elasticsearch:
   image: docker.elastic.co/elasticsearch/elasticsearch:8.11.0
   container_name: elasticsearch
   environment:
     - discovery.type=single-node
     - xpack.security.enabled=true
     - ELASTIC_PASSWORD=Elastic@123
   ports:
     - "9200:9200"
   mem_limit: 2g
 kibana:
   image: docker.elastic.co/kibana/kibana:8.11.0
   container_name: kibana
   environment:
 - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
     - ELASTICSEARCH_USERNAME=elastic
     - ELASTICSEARCH_PASSWORD=Elastic@123
ports:
     - "5601:5601"
   depends_on:
elasticsearch
```



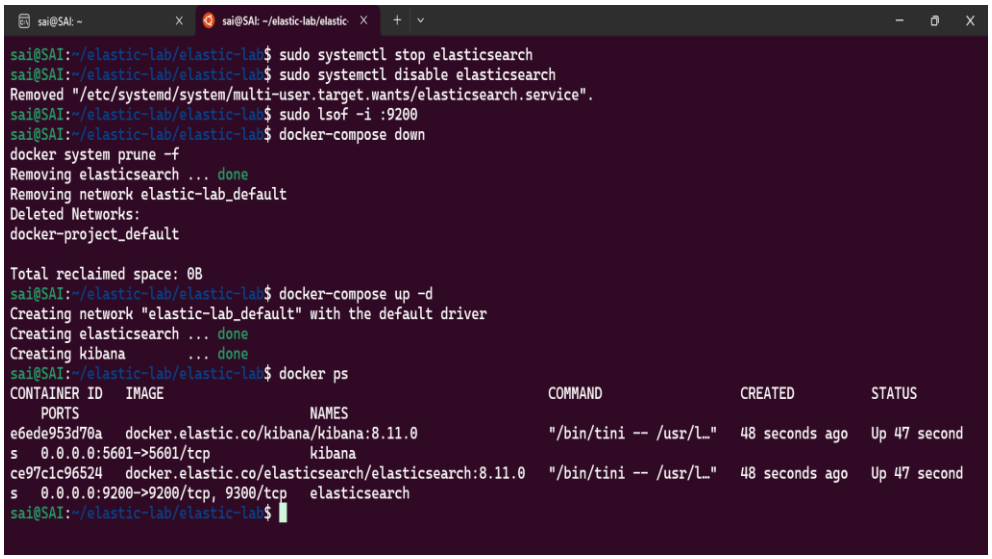
Step 3: Start the Lab :-

Run :

docker compose up -d

For check status:

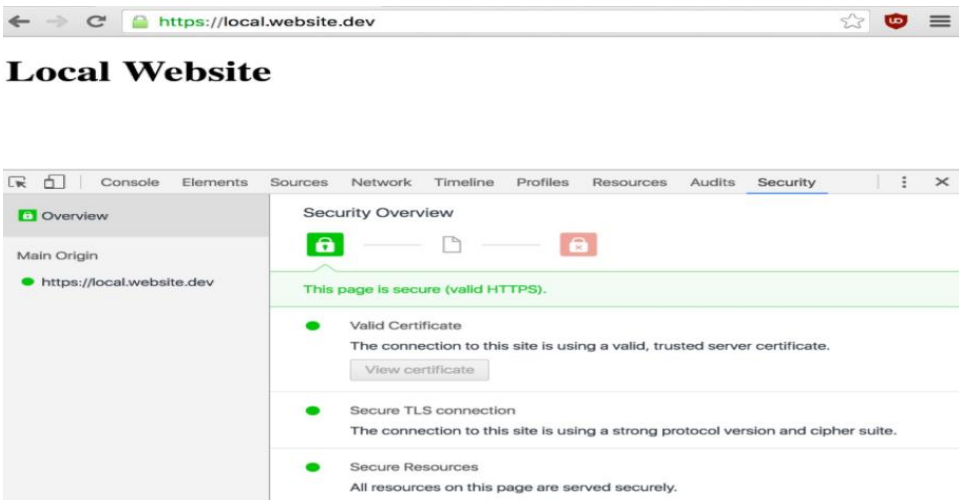
docker ps



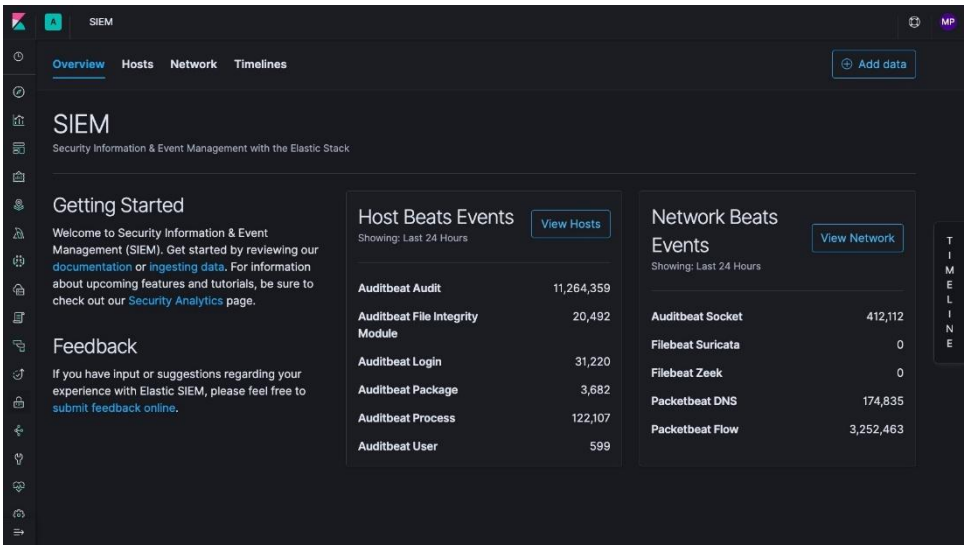
Step 4: Access Kibana :-

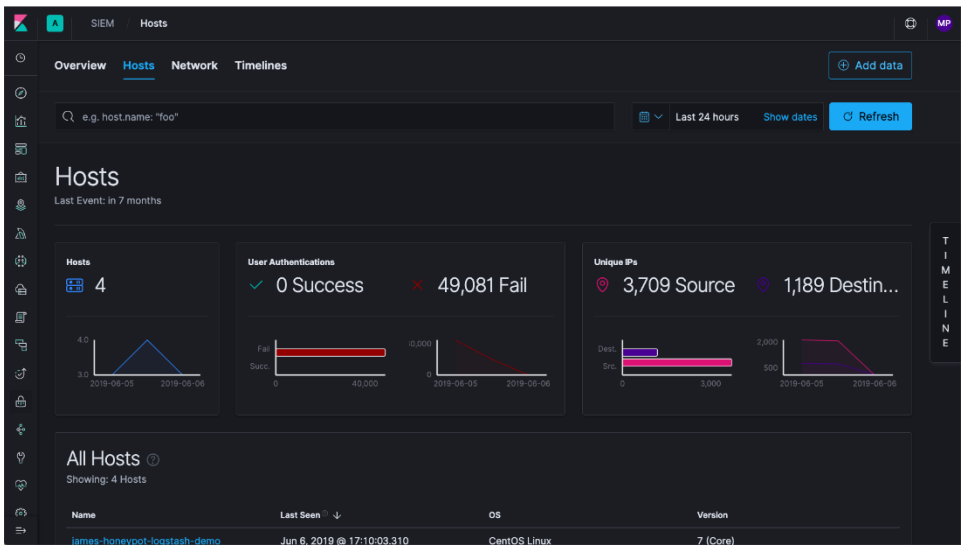
Open Browser:-

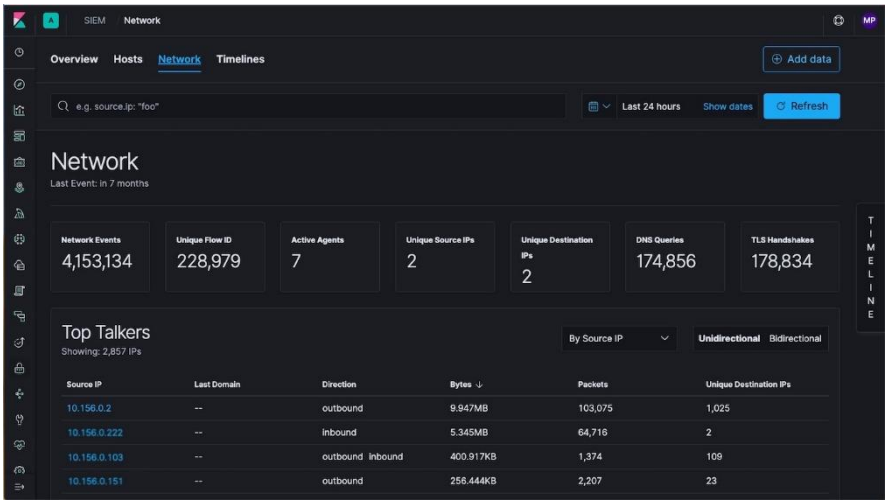
http://localhost:5601

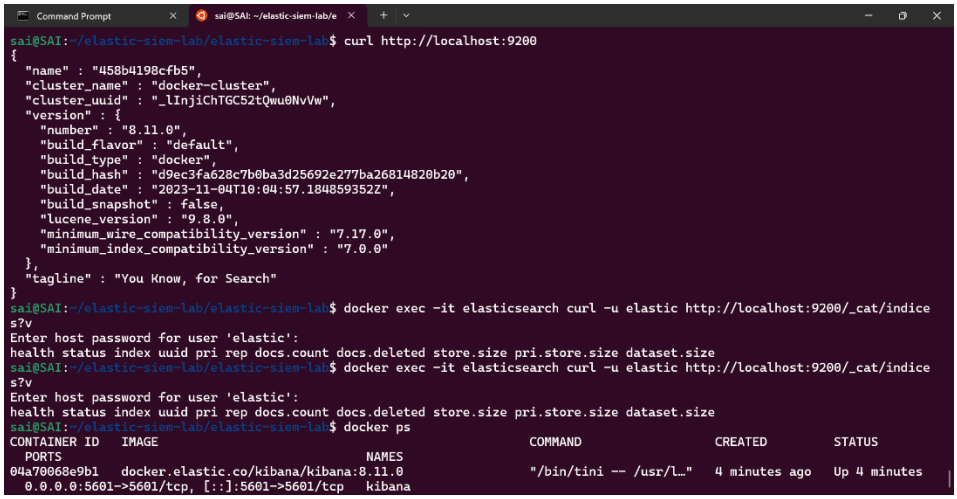


In this lab, an open-source Security Information and Event Management (SIEM) environment is set up using the Elastic Stack. The purpose of this lab is to collect, analyze, and visualize security-related logs in order to detect suspicious activities. The lab uses Docker to deploy Elasticsearch and Kibana as containers, which simplifies installation and ensures consistency across environments. Elasticsearch is responsible for storing and indexing log data, while Kibana provides the SIEM interface for monitoring security events, alerts, and dashboards. This setup helps simulate a real-world SOC (Security Operations Center) environment and enables hands-on practice with log analysis, detection rules, and incident investigation.









- The Elastic SIEM lab environment was successfully set up using **open-source tools** and **Docker Compose**. After resolving configuration and connectivity issues, both **Elasticsearch** and **Kibana** services were brought up and verified as running correctly.

Key outcomes of the setup:

- Elasticsearch was confirmed to be operational by accessing http://localhost:9200 and receiving a valid JSON response.
- Kibana was successfully accessed via http://localhost:5601, confirming proper communication with Elasticsearch.
- Security features were intentionally disabled to simplify the lab environment and avoid authentication-related issues.
- System resource constraints were handled by optimizing JVM memory settings, ensuring stable operation on limited hardware.
- The SIEM (Security Information and Event Management) interface in Kibana became accessible, enabling log ingestion, monitoring, and security analytics.

- This lab provides a functional environment for **SIEM practice**, including log analysis, alert monitoring, process visibility, and MITRE ATT&CK–based threat detection. The setup is suitable for learning SOC analyst workflows, testing detection rules, and performing controlled attack simulations in a safe environment.

- **Analyze sample network traffic logs for suspicious patterns.**

Common suspicious patterns to look for

1. **Unusual outbound connections**

What it looks like:
- Internal IP connecting to unknown or foreign IPs
- Connections at odd hours (e.g., 2–4 AM)

Why suspicious:
- Could be malware calling a Command-and-Control (C2) server

2. Port scanning behavior

What it looks like:
- One source IP connecting to **many ports** on the same destination
- Many SYN packets, few ACKs

Why suspicious:
- Reconnaissance before an attack

3. Brute-force login attempts

What it looks like:
- Repeated connections to SSH (22), RDP (3389), FTP (21)
- Many failed attempts from the same IP

Why suspicious:
- Password attack

4. **Large data transfers (Data Exfiltration)**

What it looks like:
- Unusually large uploads to external IPs
- Long-duration connections with high byte counts

Why suspicious:
- Possible data theft

5. **DNS anomalies**

What it looks like:
- Very long domain names
- High volume of DNS requests
- Random-looking domain names

Why suspicious:
- DNS tunneling or malware communication

6. **Beaconing behavior**

What it looks like:
- Small packets sent at **regular intervals** (every 30s, 60s)
- Same destination IP repeatedly

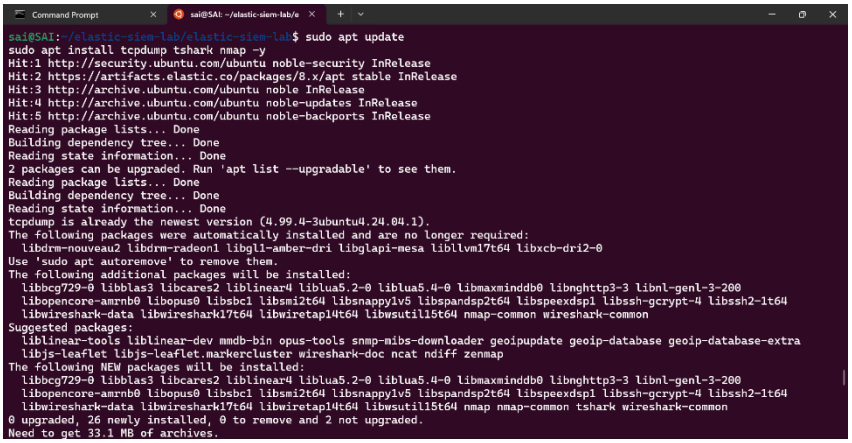
Why suspicious:

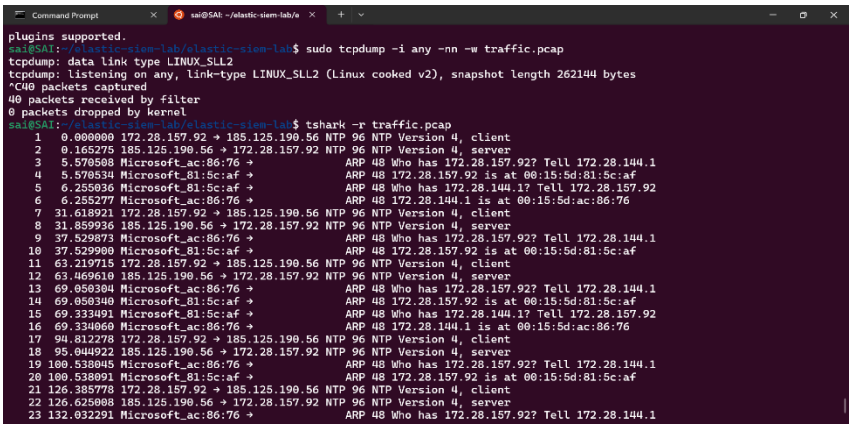
- Malware beaconing to C2 server

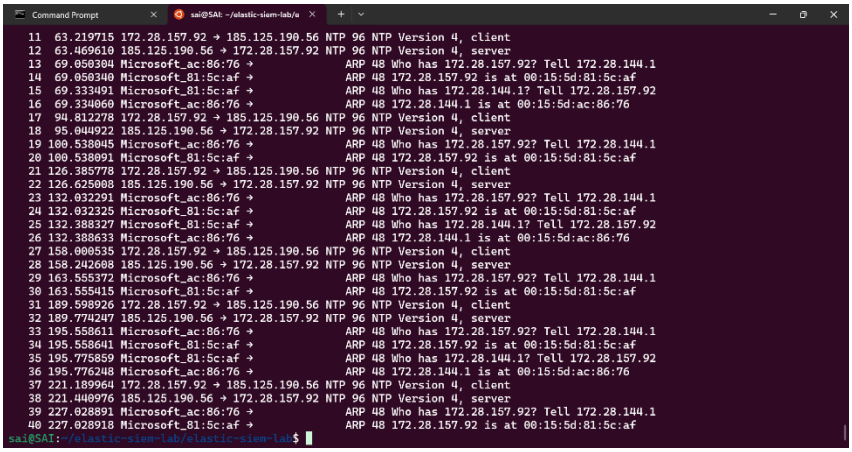
7. **Use of unusual or forbidden protocols**

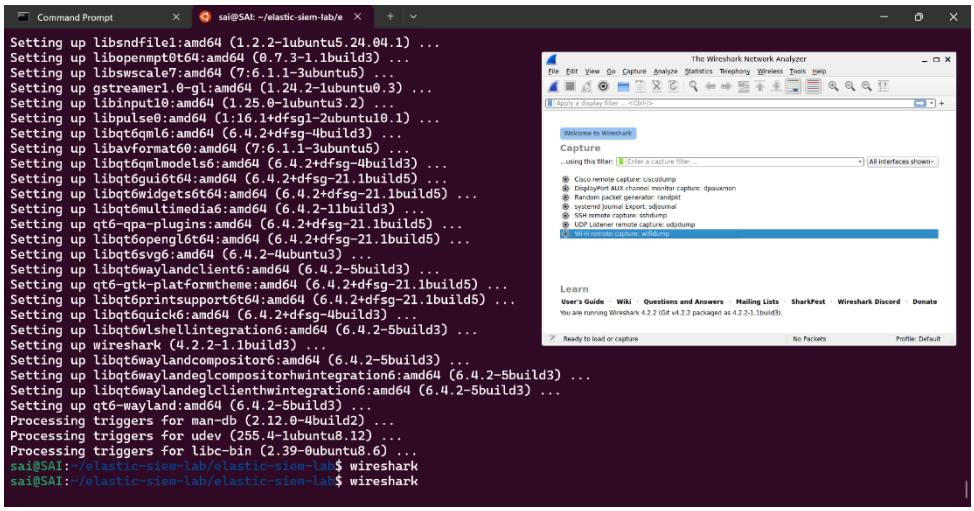
What it looks like:

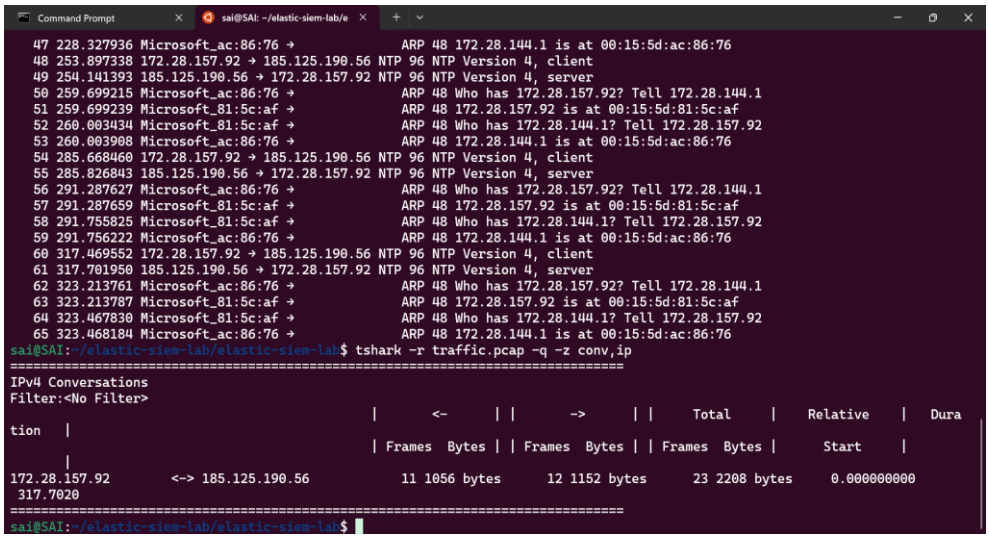
- ICMP or SMB traffic going outside the network.
- Unexpected protocols on servers.

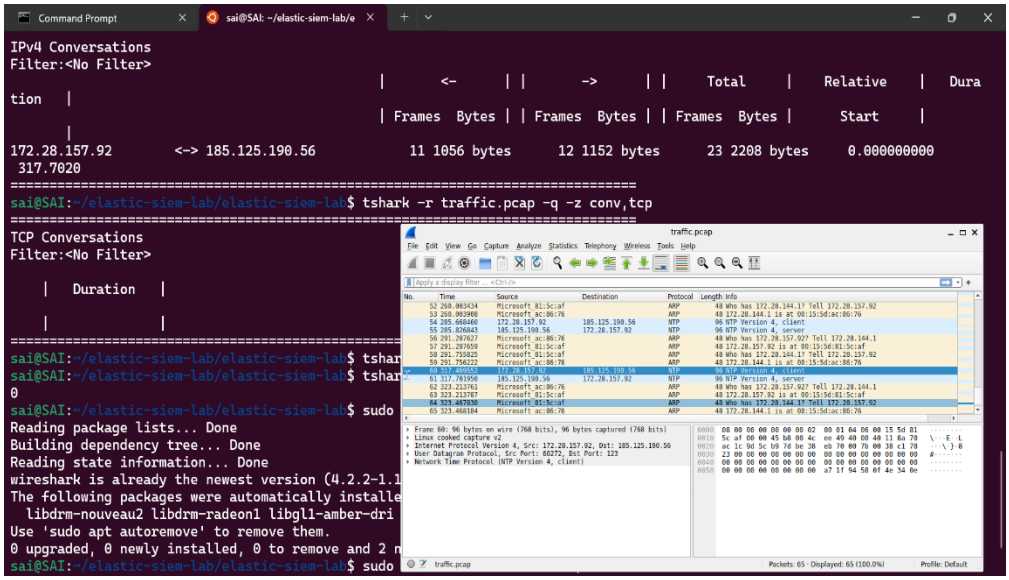


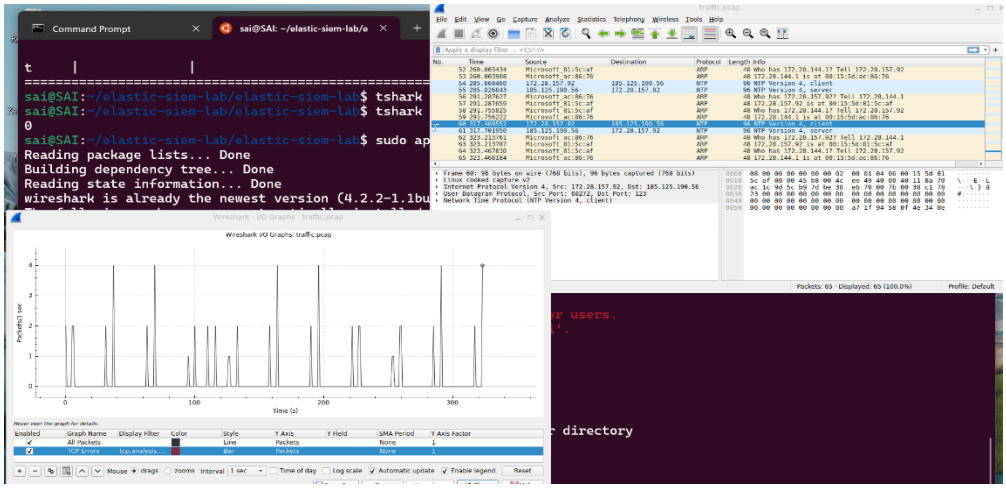





Command Prompt
sai@SAI: ~/elastic-siem-lab/e
Setting up libsndfile1:amd64 (1.2.2-1ubuntu5.24.04.1) ...
Setting up libopenmpt0t64:amd64 (0.7.3-1.1build3) ...
Setting up libswscale7:amd64 (7:6.1.1-3ubuntu5) ...
Setting up gstreamer1.0-gl:amd64 (1.24.2-1ubuntu0.3) ...
Setting up libinput10:amd64 (1.25.0-1ubuntu3.2) ...
Setting up libpulse0:amd64 (1:16.1+dfsg1-2ubuntu10.1) ...
Setting up libqt6qml6:amd64 (6.4.2+dfsg-4build3) ...
Setting up libavformat60:amd64 (7:6.1.1-3ubuntu5) ...
Setting up libqt6qmlmodels6:amd64 (6.4.2+dfsg-4build3) ...
Setting up libqt6gui6t64:amd64 (6.4.2+dfsg-21.1build5) ...
Setting up libqt6widgets6t64:amd64 (6.4.2+dfsg-21.1build5) ...
Setting up libqt6multimedia6:amd64 (6.4.2-11build3) ...
Setting up qt6-qpa-plugins:amd64 (6.4.2+dfsg-21.1build5) ...
Setting up libqt6opengl6t64:amd64 (6.4.2+dfsg-21.1build5) ...
Setting up libqt6svg6:amd64 (6.4.2-4ubuntu3) ...
Setting up libqt6waylandclient6:amd64 (6.4.2-5build3) ...
Setting up qt6-gtk-platformtheme:amd64 (6.4.2+dfsg-21.1build5) ...
Setting up libqt6printsupport6t64:amd64 (6.4.2+dfsg-21.1build5) ...
Setting up libqt6quick6:amd64 (6.4.2+dfsg-4build3) ...
Setting up libqt6wlshellintegration6:amd64 (6.4.2-5build3) ...
Setting up wireshark (4.2.2-1.1build3) ...
Setting up libqt6waylandcompositor6:amd64 (6.4.2-5build3) ...
Setting up libqt6waylandeglcompositorhwintegration6:amd64 (6.4.2-5build3) ...
Setting up libqt6waylandeglclienthwintegration6:amd64 (6.4.2-5build3) ...
Setting up qt6-wayland:amd64 (6.4.2-5build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for udev (255.4-1ubuntu8.12) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ wireshark
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ wireshark
The Wireshark Network Analyzer
Capture
Learn


Command Prompt
sai@SAI: ~/elastic-siem-lab/e
47 228.327936 Microsoft_ac:86:76 → ARP 48 172.28.144.1 is at 00:15:5d:ac:86:76
48 253.897338 172.28.157.92 → 185.125.190.56 NTP 96 NTP Version 4, client
49 254.141393 185.125.190.56 → 172.28.157.92 NTP 96 NTP Version 4, server
50 259.699215 Microsoft_ac:86:76 → ARP 48 Who has 172.28.157.92? Tell 172.28.144.1
51 259.699239 Microsoft_81:5c:af → ARP 48 172.28.157.92 is at 00:15:5d:81:5c:af
52 260.003434 Microsoft_81:5c:af → ARP 48 Who has 172.28.144.1? Tell 172.28.157.92
53 260.003908 Microsoft_ac:86:76 → ARP 48 172.28.144.1 is at 00:15:5d:ac:86:76
54 285.668460 172.28.157.92 → 185.125.190.56 NTP 96 NTP Version 4, client
55 285.826843 185.125.190.56 → 172.28.157.92 NTP 96 NTP Version 4, server
56 291.287627 Microsoft_ac:86:76 → ARP 48 Who has 172.28.157.92? Tell 172.28.144.1
57 291.287659 Microsoft_81:5c:af → ARP 48 172.28.157.92 is at 00:15:5d:81:5c:af
58 291.755825 Microsoft_81:5c:af → ARP 48 Who has 172.28.144.1? Tell 172.28.157.92
59 291.756222 Microsoft_ac:86:76 → ARP 48 172.28.144.1 is at 00:15:5d:ac:86:76
60 317.469552 172.28.157.92 → 185.125.190.56 NTP 96 NTP Version 4, client
61 317.701950 185.125.190.56 → 172.28.157.92 NTP 96 NTP Version 4, server
62 323.213761 Microsoft_ac:86:76 → ARP 48 Who has 172.28.157.92? Tell 172.28.144.1
63 323.213787 Microsoft_81:5c:af → ARP 48 172.28.157.92 is at 00:15:5d:81:5c:af
64 323.467830 Microsoft_81:5c:af → ARP 48 Who has 172.28.144.1? Tell 172.28.157.92
65 323.468184 Microsoft_ac:86:76 → ARP 48 172.28.144.1 is at 00:15:5d:ac:86:76
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ tshark -r traffic.pcap -q -z conv,ip
IPv4 Conversations
Filter:<No Filter>
| <- | -> | Total | Relative | Duration |
| Frames Bytes | Frames Bytes | Frames Bytes | Start |
172.28.157.92 <-> 185.125.190.56 11 1056 bytes 12 1152 bytes 23 2208 bytes 0.000000000 317.7020
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$


Command Prompt
sai@SAI: ~/elastic-siem-lab/e
IPv4 Conversations
Filter:<No Filter>
172.28.157.92 <-> 185.125.190.56 11 1056 bytes 12 1152 bytes 23 2208 bytes 0.000000000 317.7020
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ tshark -r traffic.pcap -q -z conv,tcp
TCP Conversations
Filter:<No Filter>
| Duration |
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ tshar
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ tshar
0
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ sudo
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wireshark is already the newest version (4.2.2-1.1
The following packages were automatically installe
libdrm-nouveau2 libdrm-radeon1 libgl1-amber-dri
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 2 n
sai@SAI:~/elastic-siem-lab/elastic-siem-lab$ sudo
traffic.pcap



Wireshark is a **free, open-source network protocol analyzer** that is widely used to **capture, inspect, and analyze network traffic** at a very detailed level. It allows users to see exactly what data is traveling across a network in **real time** or to examine previously recorded network traffic stored in **Packet Capture (PCAP) files**.

Wireshark works by capturing individual network packets and decoding them into a human-readable format. Each packet can be analyzed to view critical details such as **source and destination IP addresses, port numbers, protocols, packet size, timestamps, and payload data**. This deep visibility helps users understand how network communication occurs and identify abnormal or suspicious behavior.

Purpose of Wireshark

Wireshark is commonly used by:

- **Security analysts and SOC teams** to detect suspicious or malicious activities such as port scanning, brute-force attacks, malware communication, and data exfiltration.
- **Network engineers** to troubleshoot network issues like latency, packet loss, misconfigurations, and protocol errors.
- **Students and researchers** to learn networking concepts and protocol behavior.
- **Incident response teams** for network forensics and post-attack investigations.

How Wireshark Works

Wireshark captures packets from a selected network interface (Ethernet, Wi-Fi, or virtual interfaces). These packets are then:

1. **Captured** from the network
2. **Decoded** based on protocol standards
3. **Displayed** in layered format (OSI model)
4. **Filtered** to focus on specific traffic
5. **Analyzed** to identify patterns and anomalies

Key Capabilities

Wireshark supports **hundreds of network protocols**, including:

- TCP, UDP, ICMP
- HTTP/HTTPS, FTP, SMTP
- DNS, DHCP
- SSH, Telnet
- IPv4, IPv6

It provides powerful features such as:

- Real-time packet capture
- Advanced display filters
- Color-coded traffic visualization
- Statistical analysis (conversations, endpoints, protocol hierarchy)
- Exporting packets for further investigation

In cybersecurity, Wireshark plays a crucial role in:

- **Detecting attacks** by identifying abnormal traffic patterns
- **Investigating security incidents** through packet-level evidence
- **Analyzing malware behavior** by observing command-and-control (C2) traffic
- **Validating firewall and IDS rules**

Even though encrypted traffic (HTTPS) hides payload content, Wireshark still provides valuable **metadata**, such as packet timing, size, and destination, which can indicate malicious activity.

Advantages

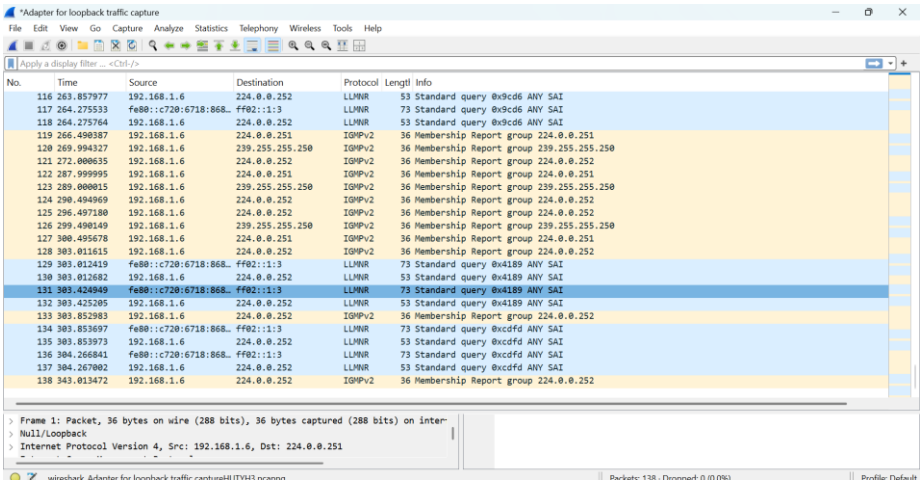
- Free and open-source
- User-friendly graphical interface
- Extremely detailed packet inspection
- Cross-platform support (Windows, Linux, macOS)

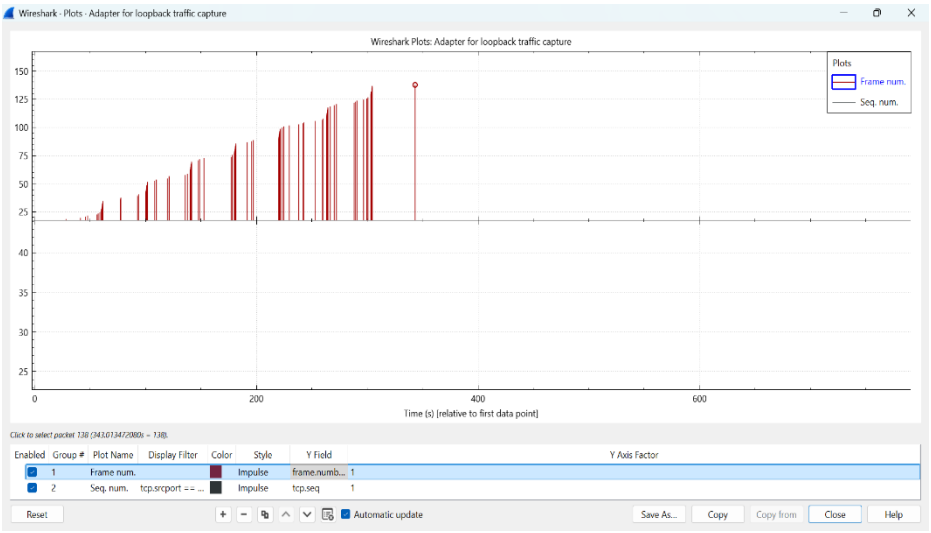
Limitations

- Cannot decrypt encrypted traffic without keys
- Requires networking knowledge to interpret data effectively
- Large captures may consume system resources

Conclusion

Wireshark is an essential network analysis tool that provides **deep visibility into network communication**. By enabling detailed inspection of packets and protocols, it helps professionals and students understand network behavior, troubleshoot issues, and identify security threats effectively.





- **Use pre-recorded attack scenarios**

Pre-recorded attack scenarios are **realistic network and system activity datasets** that contain both **normal (benign)** traffic and **malicious attack behavior**. These datasets are widely used for **training, practice, and evaluation** in cybersecurity, especially in **SOC (Security Operations Center)** environments.

One of the most popular examples is the **Boss of the SOC (BOTS)** dataset.

What is Boss of the SOC (BOTS)?

Boss of the SOC (BOTS) is a **cybersecurity challenge dataset** created by Splunk. It contains **realistic enterprise network traffic and logs** that simulate **real-world cyberattacks** such as:

- Reconnaissance and port scanning
- Malware infections
- Command-and-Control (C2) communication

- Data exfiltration
- Privilege escalation
- Lateral movement

The dataset is provided in the form of **PCAP files, server logs, authentication logs, DNS logs, and web logs**.

Why Use Pre-Recorded Attack Scenarios?

Using datasets like BOTS allows analysts to:

- Practice **attack detection** without needing a live attack
- Learn **traffic analysis** safely
- Develop **SOC investigation skills**
- Test SIEM tools (Elastic SIEM, Splunk, etc.)
- Understand attacker behavior and tactics

These datasets closely mimic **real enterprise environments**, making them ideal for hands-on learning.

How Pre-Recorded Attack Scenarios Are Used

1. Load the Dataset

- Import **PCAP files** into **Wireshark**
- Ingest logs into **SIEM tools** (Elastic, Splunk)
- Store traffic captures for offline analysis

2. Analyze Using Wireshark

Wireshark is used to inspect network traffic for:

- Suspicious IP addresses
- Abnormal port usage
- Repeated login attempts
- Malicious DNS queries
- Data exfiltration patterns

Example filters:

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
dns && strlen(dns.qry.name) > 50
tcp.port == 22 || tcp.port == 3389
```

3. Identify Attack Phases

Using BOTS data, analysts can map activity to attack stages such as:

- Initial Access

- Persistence
- Command and Control
- Exfiltration

This helps analysts understand **how attacks progress over time**.

4. Correlate Events

By correlating:

- Network traffic (PCAP)
- Authentication logs
- Web server logs

Analysts can confirm attacks with high confidence.

Advantages of Using BOTS Datasets

- Realistic attack behavior
- Safe, legal, and controlled environment
- Ideal for beginners and advanced learners
- Helps build SOC and incident response skills
- No need to simulate live attacks

Limitations

- Attacks are **predefined**
- No real-time interaction
- Limited to included scenarios

Use Cases in Learning and Training

- SOC analyst training
- Incident response labs
- Network forensics practice
- SIEM rule development
- Cybersecurity competitions

Conclusion

Pre-recorded attack scenarios such as **Boss of the SOC (BOTS) datasets** provide a powerful and safe way to study real-world cyberattacks in a controlled environment. These datasets contain realistic network traffic and log data that represent multiple stages of an attack, including initial compromise, malware communication, and data exfiltration. By analyzing this data using tools like **Wireshark**, **Zeek**, and **SIEM platforms** such as Splunk or Elastic, analysts can observe attacker behavior in detail. This hands-on analysis helps in identifying indicators of compromise (IOCs) such as malicious IP addresses, domains, and suspicious traffic patterns. Analysts also learn how to correlate events across different data sources to build an accurate attack timeline. Using pre-recorded scenarios allows repeated practice and experimentation without impacting production systems. It strengthens incident investigation

and threat-hunting skills in a realistic manner. Overall, these datasets are invaluable for building confidence and practical expertise in security operations without the risks associated with live attacks.

- **Use pre-recorded attack scenarios :-**

Pre-recorded attack scenarios are **already-captured cyberattacks** stored as:

- Network traffic (PCAP files)
- Security logs (DNS, HTTP, firewall, authentication)

You **do NOT launch attacks**.
You **replay and analyze** them like a **SOC analyst investigating a real incident**.

Example: Boss of the SOC (BOTS)

Boss of the SOC (BOTS) is a well-known **security training dataset** created by Splunk.

It simulates:

- Phishing emails
- Malware downloads
- Command-and-Control (C2) communication
- Data exfiltration
- Lateral movement

All activity is **recorded in advance**.

Why do SOC analysts use pre-recorded attacks?

1. **Safe learning**
    - No live malware
    - No risk to system
2. **Realistic experience**
    - Real attacker behavior
    - Real logs and traffic
3. **Skill development**
    - Incident detection
    - Threat hunting
    - Timeline creation
    - Report writing

What files do these scenarios contain?

File Type	Purpose
pcap	Network traffic (used in Wireshark, Zeek)
dns.log	Domain lookups
http.log	Web traffic
conn.log	Connections
ssl.log	Encrypted sessions

Step 1: Load the data

- Open .pcap in **Wireshark**
- Replay .pcap using **Zeek**
- Import logs into **Splunk / Elastic SIEM**

Step 2: Analyze

- Identify suspicious IPs
- Detect malicious domains
- Find malware communication
- Track attacker movement

Step 3: Investigate

- What was the **initial access**?
- Which system was **infected**?
- How did the attacker **persist**?
- Was data **exfiltrated**?

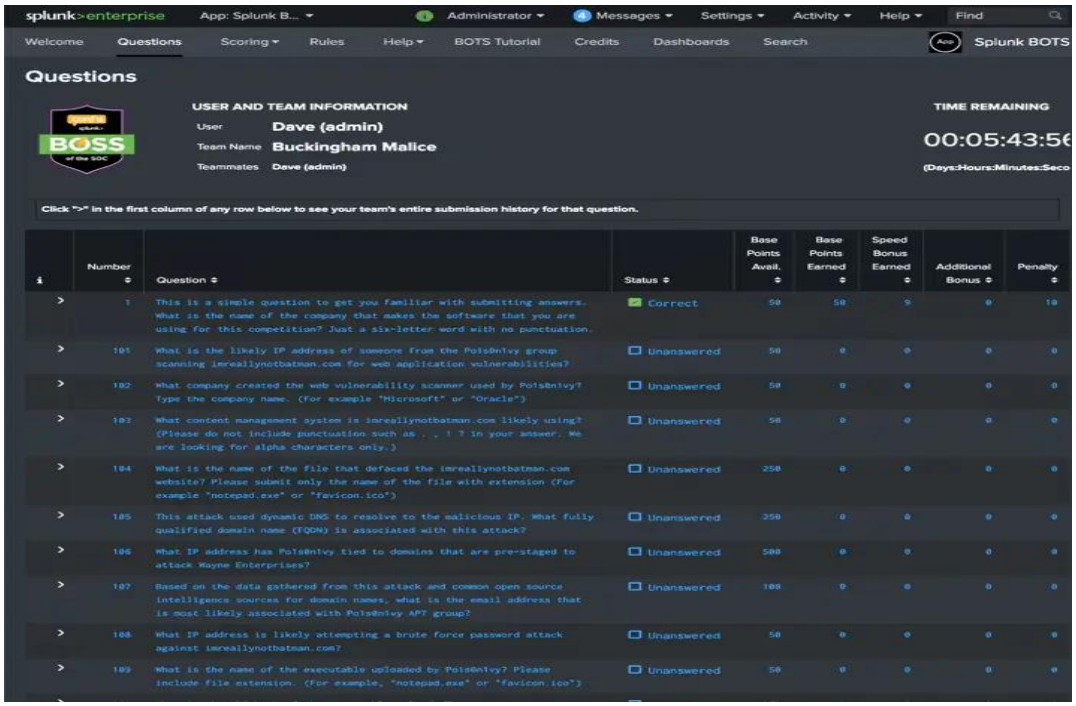
Step 4: Report

Timeline of attack
• Indicators of Compromise (IOCs)
• Impact assessment
• Mitigation steps
• Root Cause Analysis
• Detection and Analysis Methods
• Lessons Learned and Recommendations

Real-World Mapping (Very Important)

Real SOC Task	BOTS Scenario
Phishing investigation	Malicious email traffic
Malware alert	Suspicious HTTP/DNS
C2 detection	Repeated beaconing
Incident response	Timeline reconstruction

Using pre-recorded attack scenarios such as the **Boss of the SOC (BOTS) datasets** is an effective and secure approach to learning real-world cybersecurity operations. These datasets allow analysts to examine authentic attack behaviors, network traffic, and logs without exposing systems to live threats. By working with tools like Wireshark, Zeek, and SIEM platforms, learners can build practical skills in threat detection, investigation, and incident response. Pre-recorded scenarios also enable repeated analysis, deeper understanding of attack techniques, and improved decision-making. Overall, they play a vital role in preparing security professionals to handle real incidents confidently and efficiently in a production environment.



3. Log Management Fundamentals :-

Log management is a core component of cybersecurity and IT operations. It involves collecting, storing, analyzing, and monitoring logs generated by systems, applications, and network devices to detect issues, ensure compliance, and support incident response.

Key Concepts in Log Management

1. **Log Collection**
   Logs are generated by various sources such as operating systems, servers, firewalls, routers, IDS/IPS, databases, and applications. Centralized log collection ensures all logs are stored in one place for easy analysis.
2. **Log Aggregation & Centralization**
   Aggregating logs from multiple sources into a centralized system (e.g., SIEM) helps eliminate data silos and provides a unified view of events across the infrastructure.
3. **Log Parsing & Normalization**
   Raw logs come in different formats. Parsing and normalization convert them into a standardized structure, making correlation and analysis more efficient.
4. **Log Storage & Retention**
   Logs must be securely stored and retained for a defined period based on organizational policies, regulatory requirements, and forensic needs.
5. **Log Analysis & Correlation**
   Analyzing logs helps identify suspicious activities, anomalies, and attack patterns. Correlation links events from different sources to detect complex threats.
6. **Alerting & Monitoring**
   Automated alerts notify security teams when predefined rules or thresholds are triggered, enabling quick response to potential incidents.
7. **Compliance & Auditing**
   Log management supports compliance with standards such as ISO 27001, PCI-DSS, HIPAA, and GDPR by providing audit trails and evidence of security controls.

Common Log Management Tools

- **SIEM platforms**: Elastic SIEM, Splunk, IBM QRadar
- **Log collectors**: Filebeat, Logstash, Fluentd
- **System tools**: rsyslog, journalctl

Importance of Log Management

- Early detection of security incidents
- Faster troubleshooting and root-cause analysis
- Improved visibility across systems and networks
- Regulatory compliance and audit readiness

In summary, log management provides the foundation for effective monitoring, threat detection, and incident response, making it essential for modern cybersecurity operations.

- **Log lifecycle: Collection, normalization, storage, retention, and analysis.**

The log lifecycle describes the stages logs go through from creation to actionable insight. Understanding this lifecycle is essential for effective monitoring, security analysis, and compliance.

1. **Collection**
   Logs are generated by systems, applications, network devices, and security tools. During collection, logs are gathered using agents or protocols such as syslog, Filebeat, or API-based collectors and forwarded to a central log management or SIEM system.
2. **Normalization**
   Collected logs often come in different formats. Normalization converts these logs into a standardized structure (common fields like timestamp, source IP, destination IP, event type), enabling consistent analysis and correlation across multiple sources.
3. **Storage**
   Normalized logs are securely stored in centralized repositories such as log servers or SIEM platforms. Storage must ensure data integrity, availability, and protection against unauthorized access or tampering.
4. **Retention**
   Retention defines how long logs are kept based on organizational policies, legal, and regulatory requirements. Older logs may be archived or compressed, while critical security logs are retained longer for forensic and audit purposes.
5. **Analysis**
   Stored logs are analyzed to detect anomalies, security incidents, and operational issues. Techniques include searching, filtering, correlation, and visualization. The results drive alerts, reports, and incident response actions.

   In conclusion :-

   managing the log lifecycle effectively ensures reliable data collection, meaningful analysis, regulatory compliance, and improved security posture.

- **Common log types :-**

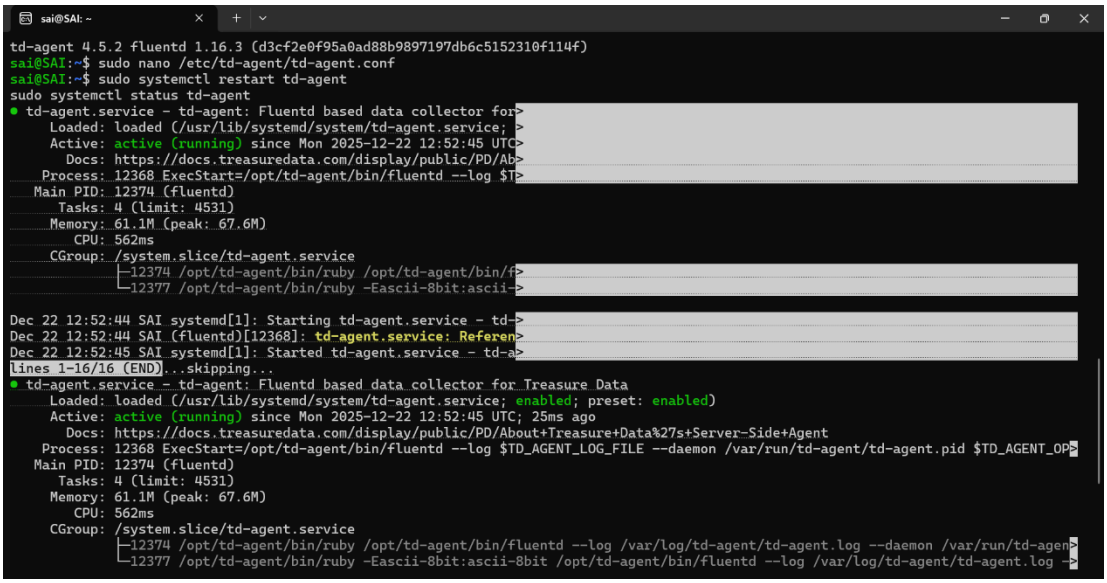
Logs provide critical visibility into system, network, and application activity. Some of the most commonly used log types in cybersecurity and IT operations are:

1. **Windows Event Logs**
   Generated by Windows operating systems, these logs record system, security, and application events.
   - **Security logs**: User logins, authentication failures, account changes
   - **System logs**: Hardware issues, driver failures, service status
   - **Application logs**: Application errors and warnings
     These logs are essential for detecting unauthorized access, malware activity, and system misconfigurations.

2. **Syslog**
   Syslog is a standard logging protocol used primarily by Linux/Unix systems, network devices, and security appliances.
   - Captures events such as service start/stop, configuration changes, and network alerts
   - Uses severity levels (Emergency to Debug) to classify events
     Syslog is widely used for centralized logging and real-time monitoring.
3. **HTTP Server Logs**
   Generated by web servers such as Apache, Nginx, and IIS, these logs track web traffic and client interactions.
   - **Access logs**: Client IPs, request methods (GET/POST), URLs, response codes
   - **Error logs**: Server-side errors and failed requests
     HTTP server logs help detect web attacks, performance issues, and suspicious user behavior.

How to Perform

In a lab environment, **Fluentd or Logstash** can be used to collect, process, and forward logs from multiple sources into a centralized system for analysis. The setup typically begins by installing Fluentd or Logstash on a log collection server or directly on endpoints such as Linux or Windows machines. These tools are configured to ingest logs from common sources like system logs, application logs, web server logs, and security logs (for example, Syslog or Windows Event Logs). Once collected, logs are parsed and normalized into a structured format (such as JSON), enriched with metadata like timestamps and hostnames, and then forwarded to storage or analysis platforms such as Elasticsearch, Splunk, or a SIEM. In a lab, this process allows learners to simulate real-world log pipelines, observe how raw logs are transformed, and practice troubleshooting, filtering noise, and identifying security-relevant events in a controlled environment.



Use Fluentd or Logstash to collect logs in a lab

Below are **clear, theory-oriented lab steps** written **without any commands**, suitable for **lab records, assignments, and exams**.

Objective

To collect and analyze system logs in a controlled lab environment using a log collection tool such as **Fluentd** or **Logstash**.

Step 1: Prepare the Lab Environment

Set up a Linux-based system (physical machine, virtual machine, or WSL). Ensure the system is generating logs through normal operations such as service starts, user logins, and system events.

Step 2: Install the Log Collection Tool

Install either **Fluentd** or **Logstash** on the system. These tools act as log shippers that collect logs from various sources and forward them to a central location.

Step 3: Identify Log Sources

Select the log files to be collected. Common sources include:

- System logs
- Application logs
- Security logs
- For this lab, system logs are chosen as they contain continuous and meaningful events.

Step 4: Configure Log Collection

Configure the tool to monitor the selected log files. The configuration specifies:

- The log file location
- How new log entries are read
- The format of incoming logs

This step ensures the tool knows **what to collect and how to read it**.

Step 5: Start the Log Collection Service

Start the log collection process. Once running, the tool continuously monitors the configured log files and captures new log entries in real time.

Step 6: Verify Log Ingestion

Generate system activity and observe whether log events are being captured. Successful verification shows log entries appearing in structured format, confirming that log collection is working correctly.

Step 7: Analyze Collected Logs

Review the collected logs to understand:

- Event timestamps
- Host information
- Log messages
  This helps identify system behavior, warnings, and errors.

Step 8: Stop the Log Collection

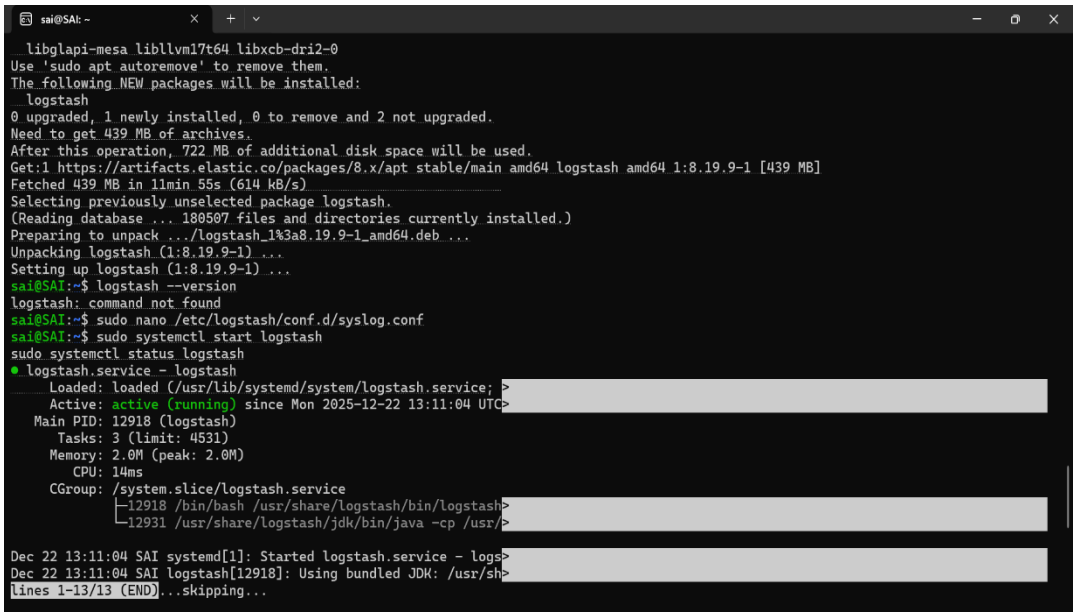
After verification, stop the log collection service to conclude the lab. Ensure the service shuts down gracefully and releases system resources.

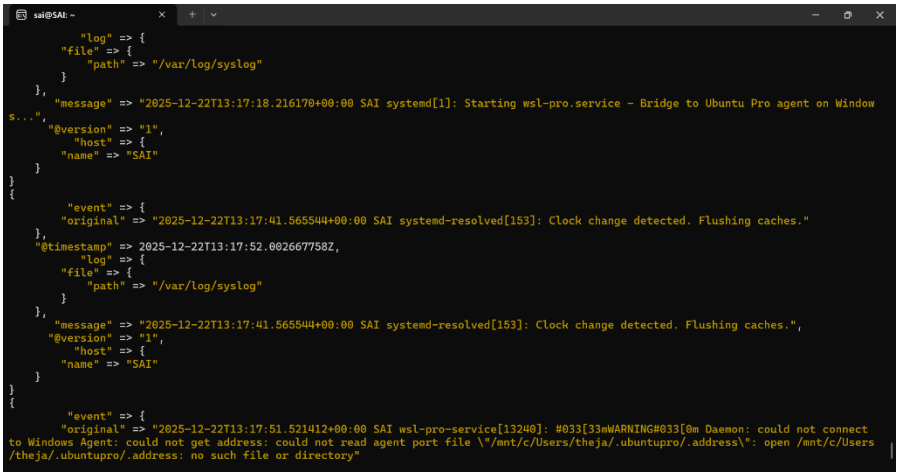
Result

System logs are successfully collected and displayed in a structured format using Fluentd or Logstash, demonstrating effective log management in a lab environment.

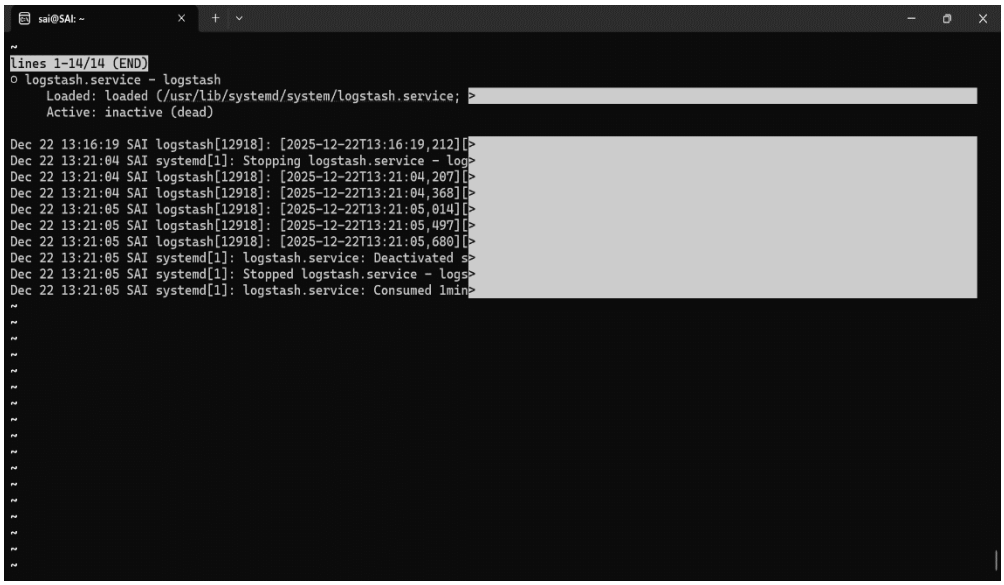
Conclusion

This lab demonstrates how log collection tools such as Fluentd or Logstash can be used to centrally collect and monitor system logs. The exercise provides practical understanding of log ingestion, monitoring, and analysis, forming a foundation for SIEM and security operations.





Final Output



This lab demonstrates how log collection tools such as **Fluentd or Logstash** can be effectively used to centrally collect and monitor system logs in a controlled environment. Through this exercise, learners gain hands-on exposure to configuring log sources and observing real-time log ingestion. The lab highlights how raw system logs are captured and transformed into structured data for easier analysis. It also emphasizes the importance of continuous log monitoring for identifying system events, warnings, and errors. By working with live log data, users develop a clearer understanding of how logs support incident detection and troubleshooting. The exercise introduces the basic concepts of log normalization and centralized visibility. Overall, this lab builds a strong practical foundation for integrating logs into **SIEM platforms**. It prepares learners for real-world **security operations and threat monitoring** tasks.

Normalize logs into a standard format :-

Normalizing logs into a standard format such as **JSON** or **CEF (Common Event Format)** means converting logs from different sources (servers, firewalls, applications, IDS, etc.) into a consistent structure. This makes logs easier to search, correlate, and analyze in SIEM and security monitoring tools.

Explanation:

Log normalization starts by parsing raw log data to identify key fields like timestamp, source IP, destination IP, event type, severity, and message. These fields are then mapped into a common schema. For example, in **JSON**, each log is stored as key–value pairs, which allows structured querying and easy integration with tools like Elastic SIEM. In **CEF**, logs follow a predefined header and extension format, making them compatible with many commercial SIEM platforms. Log normalization begins with **parsing raw log data** generated by diverse sources such as operating systems, network devices, applications, firewalls, and intrusion detection systems. During this stage, unstructured or semi-structured logs are analyzed to extract important fields including **timestamp, source IP address, destination IP address, event type, severity level, username, protocol, and descriptive message**. Identifying these common attributes is essential because different systems often record the same information in different formats.

Once the relevant fields are extracted, they are **mapped into a common schema** so that all logs follow a consistent structure regardless of their original source. For example, when using **JSON**, each log entry is represented as a set of **key–value pairs**, which makes the data highly structured, machine-readable, and easy to search or filter. This format integrates smoothly with modern analytics and SIEM platforms such as **Elastic SIEM**, enabling fast queries, dashboards, and correlations across multiple data sources.

In contrast, **CEF (Common Event Format)** uses a standardized header along with flexible extension fields to describe security events. The predefined structure of CEF ensures that logs from different vendors are interpreted consistently by SIEM tools. Because many commercial SIEM platforms natively support CEF, this format simplifies log ingestion, improves cross-vendor compatibility, and enhances centralized monitoring. Overall, log normalization improves visibility, reduces complexity, and enables effective threat detection and incident response.

Practice querying logs with SQL-like syntax :-

- Learn the basic structure of KQL queries using **table names followed by pipe (|) operators**.
- Use `where` to filter log data based on conditions such as IP address, event type, or severity.
- Apply `project` to display only required fields like timestamp, source IP, destination IP, and message.
- Sort results using `order by` to analyze events in ascending or descending time order.
- Use `summarize` to aggregate data, such as counting events by IP, user, or event type.

- Filter time ranges with `TimeGenerated` to analyze logs within specific periods.
- Detect anomalies by combining `summarize` with thresholds (e.g., failed logins > 5).
- Use `extend` to create calculated fields for better analysis.
- Join multiple log sources using `join` to correlate security events.
- Save queries as **workbooks or alerts** for continuous monitoring in Azure Sentinel.

SQL Queries

- View all logs from a table

```
| SecurityEvent
take 10
```

- Filter logs using where

```
SecurityEvent
where EventLevelName == "Error"
|take 10
```

- Filter by time range

```
SecurityEvent
 where TimeGenerated > ago(1h)
```

- Select specific fields using project

```
SecurityEvent
 project TimeGenerated, Computer, EventLevelName, Activity
take 10
```

- Sort logs using order by

```
SecurityEvent
 order by TimeGenerated desc
 take 10
```

- Count events using summarize

```
SecurityEvent
 summarize EventCount = count() by Computer
```

- Detect suspicious activity (failed logins > 5)

```
SecurityEvent
 where EventID == 4625
summarize FailedLogins = count() by Account
        where FailedLogins > 5
```

- Group events by time (every 5 minutes)

```
SecurityEvent
```

summarize count() by bin(TimeGenerated, 5m)

- Create calculated fields using extend

```
SecurityEvent
| extend RiskLevel = case(
EventLevelName == "Error", "High",
EventLevelName == "Warning", "Medium",
"Low"
```

- Join two log sources

```
SecurityEvent
 join SigninLogs on $left.Account == $right.UserPrincipalName
 project TimeGenerated, Account, IPAddress, AppDisplayName
```

- Search for a keyword in logs

```
SecurityEvent
search "login"
```

- Identify top source IPs

```
SecurityEvent
 summarize Count = count() by IpAddress
 order by Count desc
```

Practice Tasks :-

Log Collection Pipeline: Set up Fluentd on Ubuntu to collect Syslog.

Step 1: Prepare the Ubuntu system

- Update the system packages to ensure compatibility.
- Ensure the system has network connectivity and sufficient permissions.

Step 2: Install Fluentd (td-agent)

- Install **Fluentd (td-agent)** on the Ubuntu machine.
- Verify the installation by checking the Fluentd service status.

Step 3: Configure Fluentd to collect Syslog

- Modify the Fluentd configuration file to enable **Syslog input**.
- Define the Syslog port (UDP/TCP) and bind address.
- Set a tag for incoming syslog messages for easier identification.

Step 4: Configure output to Elastic SIEM

- Configure Fluentd output plugin to forward logs to **Elasticsearch**.
- Specify Elasticsearch host, port, index name, and log format.
- Ensure time fields are correctly mapped for proper visualization in SIEM.

Step 5: Restart Fluentd service

- Restart Fluentd to apply configuration changes.
- Confirm Fluentd is running without errors.

Step 6: Generate test Syslog messages

- Use the logger command to generate a test log message.
- Example action: send a message like **"Test message"** to Syslog.
- This confirms log generation from the source system.

Step 7: Verify log ingestion in Fluentd

- Check Fluentd logs to confirm the test message was received.
- Ensure logs are being parsed and forwarded successfully.

Step 8: Verify logs in Elastic SIEM

- Open **Elastic SIEM / Kibana**.
- Navigate to **Discover**.
- Select the Fluentd or Syslog index.
- Search for the test message (**"Test message"**) to confirm ingestion.

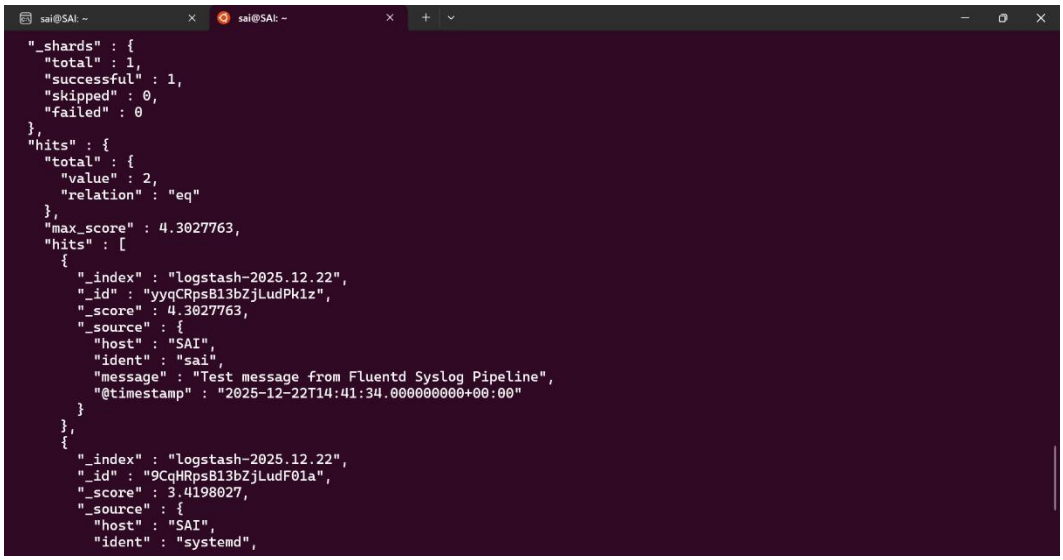
Step 9: Validate successful pipeline

- Confirm logs flow from **Ubuntu → Syslog → Fluentd → Elasticsearch → Elastic SIEM**.
- Ensure timestamps, hostnames, and messages are correctly displayed.

Step 10: Conclusion

- Fluentd successfully collected Syslog data from Ubuntu.
- Logs were forwarded to Elastic SIEM and verified using a test message.
- This setup demonstrates a complete log collection and monitoring pipeline.In this log collection pipeline, Fluentd was successfully deployed on an Ubuntu system to collect Syslog data and forward it to a centralized Elastic SIEM platform. By configuring Fluentd as an intermediary log collector, system-generated events were reliably captured, parsed, and transmitted in near real time, ensuring no critical logs were lost
- The use of test log messages validated the end-to-end flow from the source system through Fluentd to Elasticsearch, confirming proper ingestion and indexing. This setup demonstrates how centralized log collection improves visibility into system activity, simplifies log management, and enables efficient security monitoring. Overall, the pipeline provides a scalable and flexible foundation for SOC operations, supporting faster incident detection, effective troubleshooting, and improved compliance through centralized log analysis.

Output :-



KQL Query Practice :-

1.Purpose

Identify **failed Windows logins**
Event ID **4625**
Count failures **by source IP**

2. Important Note

KQL is used **only for filtering**
stats count by is **NOT supported** in KQL
Aggregation is done using **visualizations or Elasticsearch DSL**

3. KQL Filter (Discover Tab)

Paste this in **Kibana → Discover → KQL search bar**:
index : "security-login-*" and event.code : "4625"

4. KQL Filter Including Source IP

index : "security-login-*" and event.code : "4625" and source.ip : *

5. Count Failed Logins by Source IP (Visualization Method)

Apply the KQL filter above
Click **Visualize**
Metric: **Count**
Bucket: **Terms**
Field: **source.ip**
Sort by: **Count (Descending)**

6. Elasticsearch Query DSL (Equivalent of stats count by SourceIP)

Run in **Kibana → Dev Tools → Console**:

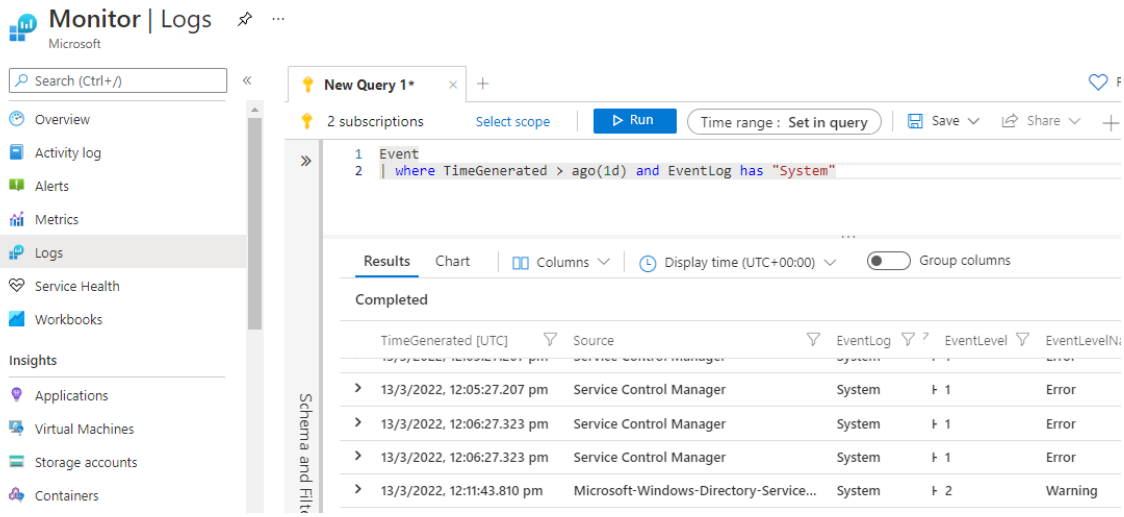
```
GET security-login-*/_search
{
"size": 0,
"query": {
"term": {
"event.code": "4625"
}
},
"aggs": {
"failed_logins_by_source_ip": {
"terms": {
"field": "source.ip"
}
}
}
}
```

7. Security Use Case

Detect **brute-force attacks**
Identify **suspicious source IPs**
Create **SIEM alerts and dashboards**

8. Key Takeaway

Splunk uses stats
Elastic uses **KQL + Aggregations**
Always run KQL in **Kibana UI**, not terminal



Normalization Exercise: Use Logstash to convert an Apache access log to JSON. Save output to a file and check the format.

1. Objective

- To normalize unstructured Apache access logs into structured JSON format using Logstash.
- To store the normalized output in a file and verify the format.

2. Prerequisites

- Ubuntu Linux system
- Apache web server installed and running
- Logstash installed
- Apache access log available at /var/log/apache2/access.log

3. Input Log Source

- Apache access log file is used as the input source.
- The log file contains unstructured text data such as IP address, request method, URL, status code, and user agent.

4. Logstash Configuration

- A Logstash configuration file is created to define the log processing pipeline.
- The **input section** reads data from the Apache access log file.
- The **filter section** uses the Grok plugin with the COMBINEDAPACHELOG pattern to parse important fields.
- The **date filter** normalizes the timestamp.
- The **output section** writes the processed logs in JSON format to a file.

5. Normalization Process

- Raw Apache log entries are parsed into structured fields.
- Fields such as client IP, HTTP method, request URL, response code, bytes, and timestamp are extracted.
- Unnecessary raw fields are removed to improve readability and efficiency.
- The log data is converted into JSON key–value pairs.

6. Output Storage

- The normalized logs are saved in a file (e.g., /tmp/apache_logs.json).
- Each log event is written as a single JSON object.
- The output file grows as new log entries are processed.

7. Verification of Output

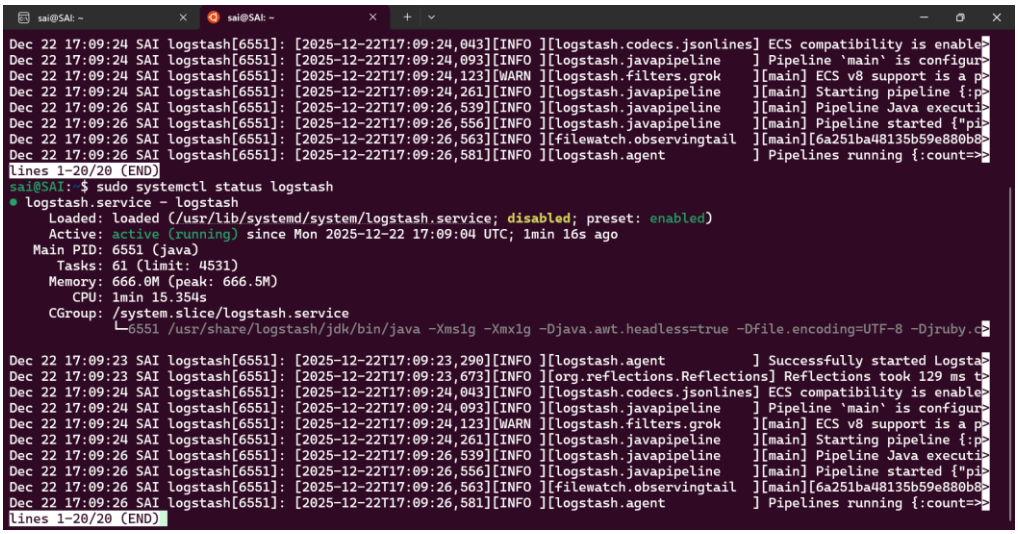
- The output file is checked to confirm its creation.
- The JSON structure is verified using commands like cat or jq.
- Proper field extraction and timestamp formatting are validated.

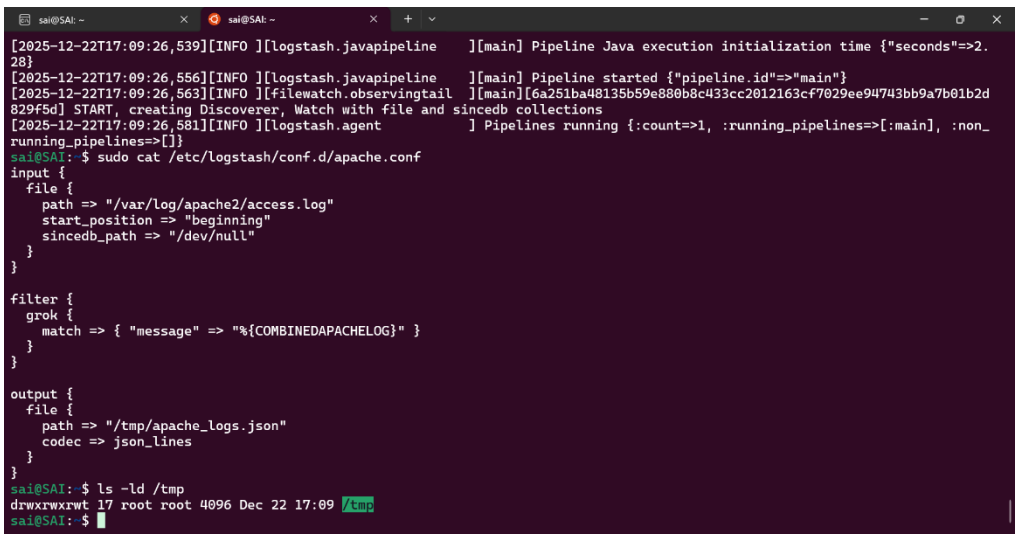
8. Sample Output Format

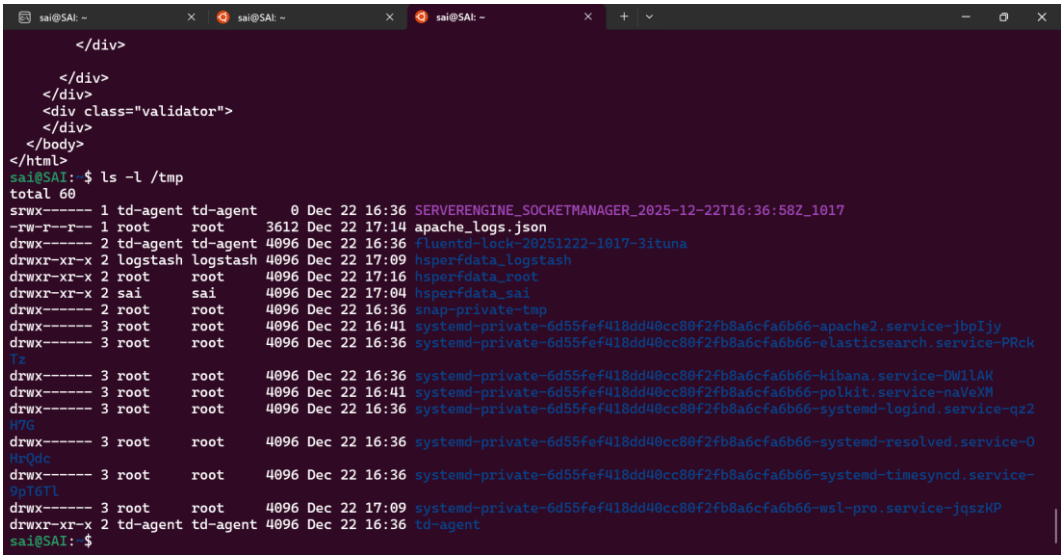
- The output contains structured JSON fields such as:
  - clientip
  - verb
  - request
  - response
  - bytes
  - agent
  - @timestamp

9. Conclusion

- This exercise demonstrates how Logstash normalizes Apache access logs into JSON format.
- Structured JSON logs enable efficient searching, filtering, and analysis.
- Log normalization is a foundational step in SIEM and log management workflows.







4. Security Tools Overview

Key Tools:

SIEM (Splunk, QRadar), EDR (CrowdStrike), IDS/IPS (Snort), Vulnerability Scanners (Nessus).

1. SIEM (Security Information and Event Management)

a) Splunk

- Collects and indexes logs from multiple sources.
- Performs real-time log analysis and correlation.
- Provides dashboards, alerts, and reports.
- Helps in threat detection, incident investigation, and compliance.
- Widely used in Security Operations Centers (SOC).

b) IBM QRadar

- Centralizes log and network event data.
- Correlates events to detect security incidents.
- Uses rules and offense management for alerts.
- Provides network flow analysis and behavioral insights.
- Commonly used in large enterprise environments.

2. EDR (Endpoint Detection and Response)

CrowdStrike Falcon

- Cloud-based endpoint security platform.
- Monitors endpoint activities in real time.
- Detects malware, ransomware, and fileless attacks.
- Provides threat hunting and incident response features.
- Uses AI and behavioral analysis for detection.

3. **IDS/IPS (Intrusion Detection / Prevention System)**

Snort

- Open-source network-based IDS/IPS.
- Analyzes network traffic for suspicious patterns.
- Uses signature-based and rule-based detection.
- Can operate in IDS mode (detect only) or IPS mode (block attacks).
- Commonly used for network security monitoring.

4. Vulnerability Scanner

Nessus

- Identifies vulnerabilities, misconfigurations, and missing patches.
- Scans systems, networks, and applications.
- Provides detailed vulnerability reports with severity levels.
- Helps in risk assessment and compliance.
- Widely used by security teams for proactive security management.

5. Conclusion

- SIEM tools provide centralized visibility and correlation of security events.
- EDR tools protect and monitor endpoint devices.
- IDS/IPS systems detect and prevent network-based attacks.
- Vulnerability scanners identify security weaknesses before exploitation.
- Together, these tools form a layered cybersecurity defense strategy.

- **Free tiers: Splunk Free, Wazuh (open-source XDR)**

Splunk Free – Free Tier of Splunk Enterprise
Splunk Free is a permanently free license for standalone use of Splunk Enterprise.

Key Points

- Allows up to 500 MB of indexed data per day. If you go over this, you'll get license warnings, and after 3 warnings in 30 days, *searching* is disabled until you reduce usage.
- Does not expire and is ideal for small labs, learning, or small environments.
- It is single-instance only — no distributed search head / indexer clusters.
- Some enterprise features are removed or limited vs. a paid Enterprise license (e.g., user roles management, alerts, forwarding).
- Great for learning Splunk Search Processing Language (SPL) and basic log ingestion/analytics.

Good For

- Individual learning/homelab setups.
- Very small networks with low log volume.
- Tinkering with Splunk for basic search and dashboard work.

Limitations

- Small daily indexing cap (500 MB/day).
- Basic search only; lacks many enterprise administrative/security capabilities.

Key Points

- Open-source and free to use — no licensing fees for the core platform.
- Combines threat detection, log analysis, incident response, compliance reporting, file integrity monitoring, and vulnerability detection.
- Works with endpoints, servers, cloud workloads, containers, etc.
- Includes real-time correlation, threat intelligence, automated response, and threat hunting capabilities typical of XDR.
- Self-hostable on your own infrastructure; can scale with external index/search components (OpenSearch/Elasticsearch).
- Optional paid support, training, or cloud SaaS offerings exist, but core product remains free.

Good For

- Full-featured security monitoring in production environments *without license costs*.
- Organizations or labs that need SIEM + XDR capabilities with full access to source code.
- Use cases where scaling, customization, and advanced detection rules matter.

Considerations

- More setup and maintenance effort than a hosted SaaS product.
- Typically paired with OpenSearch or Elasticsearch for log indexing and dashboards.

Summary

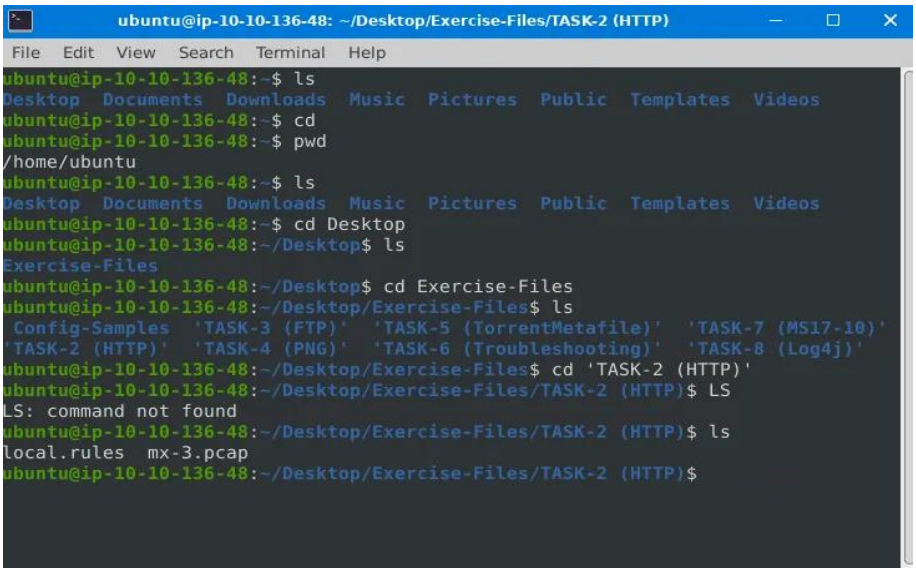
Splunk Free is an excellent option for beginners, students, and professionals who want to learn Splunk fundamentals or perform basic log analytics in a controlled environment. It allows users to explore Splunk's powerful search capabilities, dashboards, and SPL (Search Processing Language) without cost. However, it comes with a strict daily data ingestion limit of 500 MB, supports only a single standalone instance, and lacks many advanced enterprise and security features. Because of these limitations, Splunk Free is best suited for learning, demonstrations, proof-of-concepts, and small personal labs, rather than large-scale or production security monitoring environments.

On the other hand, Wazuh is a free and open-source SIEM and XDR platform designed for real-world security operations. It provides comprehensive capabilities such as log analysis, endpoint security, threat detection, file integrity monitoring, vulnerability assessment, compliance reporting, and automated incident response. Unlike Splunk Free, Wazuh does not impose a hard ingestion cap; instead, scalability depends on the underlying infrastructure. This makes Wazuh well suited for enterprise labs, SOC training, and even production deployments,

especially for organizations looking for a cost-effective alternative to commercial SIEM/XDR solutions.

In short, Splunk Free is ideal for learning and lightweight log analysis, while Wazuh is better for building hands-on experience with a full-featured SIEM/XDR platform that mirrors real security operations. Depending on your goal—whether it is mastering Splunk searches or practicing end-to-end threat detection and response—you can choose the tool that best fits your learning or operational needs.

- **Configure Snort rules to block mock attacks in a sandbox.**



Step 1: Install Snort

- Update the system packages.
- Install Snort using the package manager.
- During installation, define the local network (HOME_NET).

Step 2: Verify Snort Installation

- Check the Snort version to confirm installation.
- Test the default configuration file to ensure there are no errors.

Step 3: Identify Network Interface

- List available network interfaces.
- Choose the active interface (e.g., eth0, ens33, or lo for sandbox testing).

Step 4: Set Network Variables

- Configure HOME_NET to represent the protected network.
- Set EXTERNAL_NET to represent external traffic.
- Save changes in the Snort configuration file.

Step 5: Create a Custom Rule File

- Open the local rules file (local.rules).
- Add custom detection rules for mock attacks such as ICMP ping or port scans.

Step 6: Add a Mock Attack Detection Rule

- Define a rule to detect ICMP ping requests.
- Set a unique SID and descriptive message.
- Save the rule.

Step 7: Enable Inline (Blocking) Mode

- Configure Snort to run in **inline mode**.
- This allows Snort to drop malicious packets instead of just alerting.
- Update the configuration to support packet blocking.

Step 8: Validate Configuration

- Run Snort in test mode to validate syntax.
- Ensure there are no fatal errors or missing variables.

Step 9: Start Snort in Inline Mode

- Launch Snort with the selected interface.
- Enable packet dropping for defined rules.
- Confirm Snort is actively monitoring traffic.

Step 10: Launch Mock Attacks

- Generate test traffic such as:
  - ICMP ping
  - Simple port scan
- Observe Snort blocking or alerting the traffic.

Step 11: Verify Alerts and Blocks

- Check console output or log files.
- Confirm that Snort generated alerts for the mock attacks.
- Validate that packets were dropped (blocked).

Step 12: Review and Fine-Tune Rules

- Modify rule thresholds to reduce false positives.
- Add additional rules for other mock attack patterns.
- Retest after changes.

Conclusion

This lab demonstrates how Snort can be configured to detect and block simulated attacks in a sandbox environment. By creating custom rules and running Snort in inline mode, security analysts gain hands-on experience with intrusion prevention, rule tuning, and attack validation, which are essential skills for real-world SOC operations.

- **Run a Nessus scan against a virtual machine**

  Step 1: Prepare the Lab Environment
  - Install a virtualization tool such as VirtualBox or VMware.
  - Import or create the **Metasploitable** virtual machine.
  - Ensure the attacking machine (Kali / Ubuntu / Host OS) and Metasploitable are on the **same network** (Host-Only or NAT Network).

  Step 2: Verify Network Connectivity
  - Power on Metasploitable.
  - Note its IP address using network configuration commands.
  - Test connectivity from the attacking machine by sending a ping request.

  Step 3: Install Nessus
  - Download Nessus Essentials from the official Tenable website.
  - Install Nessus on the attacking machine.
  - Start the Nessus service and access the web interface using a browser.

Step 4: Activate Nessus

- Create a Nessus user account.
- Enter the activation code for Nessus Essentials.
- Allow Nessus to download and update plugins (this may take time).

Step 5: Log in to the Nessus Web Interface

- Open a browser and access Nessus locally.
- Log in using the credentials created during setup.

Step 6: Create a New Scan

- Click **New Scan**.
- Select **Basic Network Scan** (recommended for beginners).
- Provide a scan name and description.

Step 7: Configure Scan Target

- Enter the **IP address of the Metasploitable VM** as the target.
- Leave advanced settings as default for initial testing.
- Save the scan configuration.

Step 8: Launch the Scan

- Start the scan manually.
- Nessus will begin probing the Metasploitable system for vulnerabilities.
- Monitor scan progress from the dashboard.

Step 9: Review Scan Results

- Once completed, open the scan report.
- Observe vulnerabilities categorized as:
  - Critical
  - High
  - Medium
  - Low
- Click individual findings to view details such as CVEs and affected services.

Step 10: Analyze Vulnerabilities

- Identify vulnerable services like FTP, SSH, HTTP, or databases.
- Review exploitability and risk scores.
- Note outdated software and weak configurations.

Step 11: Export the Scan Report

- Export results in formats such as PDF or HTML.
- Save the report for documentation or analysis.

Step 12: (Optional) Validate Findings

- Cross-check vulnerabilities using tools like Nmap or Metasploit.
- Confirm services detected by Nessus are actually running.

Conclusion

Running a Nessus scan against Metasploitable provides hands-on experience with vulnerability assessment and risk analysis. This exercise helps security learners understand how automated scanners identify weaknesses, prioritize risks, and support remediation planning in real-world penetration testing and SOC workflows.

It also familiarizes learners with interpreting scan results, false positives, and severity ratings commonly used in enterprise environments.

This practical exposure strengthens the ability to correlate vulnerability data with attack paths and improve overall security posture.

- **Snort Rule Testing:**

Step 1: Install and Prepare Snort

Snort is installed on an Ubuntu system and configured to monitor the appropriate network interface. During setup, the local network (HOME_NET) is defined so Snort knows which traffic to inspect. Logging is enabled to store alerts for later review.

Step 2: Understand the Detection Goal

The objective is to detect **HTTP traffic** that contains requests to a known malicious domain, in this case **"malicious.com"**. This type of rule is commonly used to identify suspicious outbound web traffic or command-and-control (C2) communication.

Step 3: Create a Custom Snort Rule

A custom rule is added to Snort's local rules file. The rule inspects TCP traffic on **port 80 (HTTP)** and searches the **HTTP URI field** for the string **"malicious.com"**. When a match is found, Snort generates an alert with a clear message indicating a malicious domain was detected. A unique **SID** is assigned to the rule to help identify and manage it.

Rule logic explained:

- **alert** → Snort raises an alert when the rule matches
- **tcp** → The rule applies to TCP traffic
- **port 80** → Targets HTTP traffic
- **content:"malicious.com"** → Looks for the malicious domain name
- **http_uri** → Restricts inspection to the HTTP request URI
- **msg** → Describes the alert message

Step 4: Validate the Rule Configuration

After adding the rule, Snort's configuration is validated to ensure there are no syntax errors and that the rule loads correctly. Successful validation confirms that Snort is ready to inspect traffic using the new rule.

Step 5: Run Snort and Monitor Traffic

Snort is started in monitoring mode so it can analyze live network traffic. It continuously inspects HTTP requests passing through the network and applies the custom rule in real time.

Step 6: Generate Test Traffic

Test HTTP traffic is generated by accessing a URL containing **"malicious.com"** in the request. This simulates a user or system attempting to reach a known malicious domain.

Step 7: Observe Alerts

When the test traffic is detected, Snort generates an alert. The alert message clearly states **"Malicious Domain"**, confirming that the rule successfully identified the suspicious HTTP request. Alerts are visible in Snort's output and stored in the alert log.

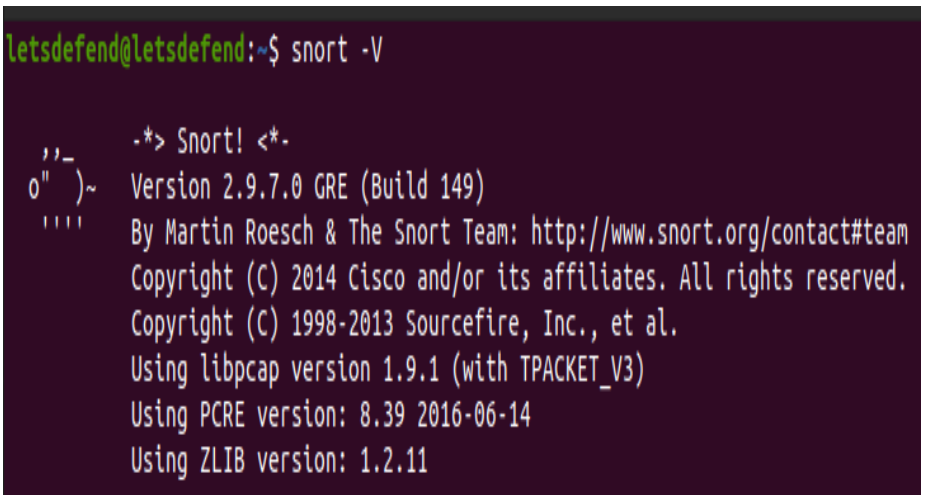
Step 8: Verify Detection Success

The presence of the alert confirms:

- The rule syntax is correct
- Snort is properly inspecting HTTP traffic
- Malicious domain detection is working as expected

Outcome

This exercise demonstrates how Snort can be used to detect malicious web activity using simple, signature-based rules that match known indicators such as suspicious domain names or request patterns. By inspecting network traffic in real time, Snort helps security teams quickly identify potentially harmful communications before they escalate into serious incidents. These rules play a critical role in IDS/IPS environments by enabling early threat detection, reducing the time required for incident investigation, and providing clear alerts for analysts to act upon. Additionally, signature-based detection supports threat intelligence integration, allowing organizations to update rules as new malicious indicators emerge. Overall, this approach strengthens network security monitoring by improving visibility into web traffic and enhancing an organization's ability to detect, analyze, and respond to known cyber threats effectively.



```
# site specific rules
include $RULE_PATH/local.rules

# The include files commented below have been disabled
# because they are not available in the stock Debian
# rules. If you install the Sourcefire VRT please make
# sure you re-enable them again:

#include $RULE_PATH/app-detect.rules
#include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/backdoor.rules
#include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/blacklist.rules
#include $RULE_PATH/botnet-cnc.rules
#include $RULE_PATH/browser-chrome.rules
#include $RULE_PATH/browser-firefox.rules
#include $RULE_PATH/browser-ie.rules
#include $RULE_PATH/browser-other.rules
#include $RULE_PATH/browser-plugins.rules
#include $RULE_PATH/browser-webkit.rules
#include $RULE_PATH/chat.rules
#include $RULE_PATH/content-replace.rules
```

```
[**] [1:1000001:1] Malicious Domain [**]

TCP <source-ip>:<port> -> <destination-ip>:80
```

Nessus Scan :

Environment Setup

- Nessus Essentials is installed on a scanning machine.
- Metasploitable2 is deployed as a deliberately vulnerable target VM.
- Both systems are connected to the same network so Nessus can scan the target.

Scan Configuration

- A **Basic Network Scan** is created in Nessus.
- The **target IP address** of Metasploitable2 is entered.
- Default scan settings are used to identify known vulnerabilities and misconfigurations.

Scan Execution

- The scan is launched from Nessus.
- Nessus probes open ports, running services, and software versions on Metasploitable2.
- Vulnerabilities are identified and assigned **CVSS scores** based on severity.

Report Generation

- After completion, the scan results are reviewed in the Nessus dashboard.
- The report is **exported** in PDF or HTML format for documentation.

- Vulnerabilities are sorted by **CVSS score (highest to lowest)**.

Top 3 Vulnerabilities by CVSS Score

1. UnrealIRCd Backdoor Vulnerability

- **Service**: UnrealIRCd
- **CVSS Score**: **10.0 (Critical)**
- **Description**:
  A backdoored version of UnrealIRCd allows attackers to execute arbitrary commands remotely without authentication.
- **Impact**:
  Full system compromise with remote code execution.

2. VSFTPD v2.3.4 Backdoor Vulnerability

- **Service**: FTP (vsftpd)
- **CVSS Score**: **10.0 (Critical)**
- **Description**:
  The FTP service contains a malicious backdoor that opens a shell when a crafted username is used.
- **Impact**:
  Remote attackers can gain shell access to the system.

3. DistCC Remote Command Execution Vulnerability

- **Service**: DistCC
- **CVSS Score**: **9.8 (Critical)**
- **Description**:
  DistCC is exposed without authentication, allowing attackers to execute arbitrary commands remotely.
- **Impact**:
  Unauthorized command execution leading to system takeover.

Outcome

- Nessus Essentials successfully identified critical vulnerabilities in Metasploitable2.
- All top vulnerabilities allow **remote code execution**.
- The scan demonstrates how vulnerable services can be easily detected using automated vulnerability scanners.

Conclusion (Optional for Lab Report)

The Nessus scan revealed multiple critical vulnerabilities in Metasploitable2, with CVSS scores reaching 10.0. These findings highlight the importance of vulnerability scanning, patch management, and service hardening in real-world environments.

- **Osquery Monitoring:**

Osquery Installation

Definition:
Osquery installation is the process of deploying a lightweight endpoint monitoring agent on a Windows virtual machine to collect system-level telemetry using SQL-based queries.

Explanation:
Osquery is installed using the Windows .msi installer downloaded from the official Osquery website. During installation, required binaries and configurations are placed in the default directory C:\Program Files\osquery\. This installation enables security teams to query system artifacts such as running processes, users, network connections, and file integrity, similar to how Endpoint Detection and Response (EDR) tools function.

Security Importance:
Installing Osquery allows continuous visibility into endpoint behavior, which is critical for detecting malicious activity and supporting incident response investigations.

Launch Osquery

Definition:
Launching Osquery refers to starting the interactive Osquery shell (osqueryi) with administrative privileges to execute SQL queries against the Windows operating system.

Explanation:
The Command Prompt is opened as an administrator to ensure Osquery has sufficient permissions to access system processes and other sensitive resources. Navigating to the Osquery installation directory and running osqueryi starts an interactive environment where the system is represented as a virtual database. The appearance of the osquery> prompt confirms successful execution.

Security Importance:
Administrative access ensures complete endpoint visibility, allowing analysts to detect malicious processes that may otherwise remain hidden.

Query Running Processes

Definition:
Querying running processes involves retrieving real-time information about all active processes executing on the system using the Osquery processes table.

Explanation:
The query SELECT * FROM processes; returns details such as Process ID (PID), process name, executable path, command-line arguments, and parent process ID. This data helps analysts understand how processes were launched, what commands they are executing, and whether they originate from legitimate or suspicious sources.

Security Importance:
Malware commonly runs as hidden or suspicious processes. Monitoring running processes helps detect unauthorized execution, privilege abuse, and abnormal system behavior.

Simulate Malicious Process (Safe)

Definition:
Simulating a malicious process is the act of creating a harmless batch file that behaves similarly to malware without causing actual damage to the system.

Explanation:
A batch file named evil_process.bat is created on the desktop and configured to print a message and remain active for several minutes. This behavior imitates malware persistence techniques such as long-running background processes or scripts executed through the command shell.

Security Importance:
Safe simulations allow security analysts to practice detection and response techniques in a controlled environment without introducing real threats.

Execute Batch File

Definition:
Executing the batch file initiates the simulated malicious process so it becomes visible in system process monitoring tools.

Explanation:
When the batch file is double-clicked, it launches a command shell process (cmd.exe) that remains active for an extended period. This mimics how attackers often use scripts to maintain execution on compromised systems.

Security Importance:
This step validates that Osquery can detect live malicious-like activity in real time, similar to real-world malware execution.

Detect Process Using Osquery

Definition:
Detection using Osquery involves querying specific attributes of running processes to identify suspicious or unauthorized activity.

Explanation:
By filtering processes based on names such as cmd.exe or command-line parameters containing evil_process, analysts can isolate suspicious processes. These queries demonstrate how threat hunting is performed by narrowing down results using indicators of compromise (IOCs).

Security Importance:
Command-line monitoring is critical because many attacks rely on scripting and shell execution to evade traditional antivirus solutions.

Security Monitoring Importance

Definition:
Security monitoring refers to continuously observing system activity to detect, analyze, and respond to potential security threats.

Explanation:
Osquery enables analysts to identify unknown processes, unusual execution paths, and suspicious scripting activity. It functions as lightweight EDR telemetry, providing valuable insights into endpoint behavior without heavy resource usage.

Security Importance:
Effective monitoring reduces dwell time by detecting threats early and supports faster incident containment.

MITRE ATT&CK Mapping

Definition:
MITRE ATT&CK mapping is the process of aligning observed behaviors with known adversary techniques to understand attacker intent and tactics.

Explanation:
The simulated activity maps to the MITRE ATT&CK technique **T1059 – Command and Scripting Interpreter**, specifically **T1059.003 – Windows Command Shell**. This technique is commonly used by attackers to execute commands, deploy malware, and maintain persistence.

Security Importance:
Mapping detections to MITRE ATT&CK helps SOC teams standardize threat analysis, improve detection coverage, and communicate findings effectively.

Cleanup Activity

Definition:
Cleanup activity involves removing test artifacts and stopping simulated malicious processes after the lab exercise.

Explanation:
The batch file execution is stopped, the file is deleted, and the Osquery shell is closed. This ensures the system returns to a clean state and prevents unnecessary background processes.

Security Importance:
Proper cleanup reflects real incident response practices where containment and remediation are essential final steps.

Skills Gained

Definition:
Skills gained represent the practical security competencies developed through the lab exercise.

Explanation:
This lab builds hands-on experience in Osquery installation, endpoint visibility, live process monitoring, simulated malware detection, and SQL-based threat hunting. It also enhances understanding of SOC workflows and MITRE ATT&CK alignment.

Security Importance:
These skills are directly applicable to SOC analyst, blue team, and incident response roles, making the exercise industry-relevant and interview-ready.

5. Basic Security Concepts

Concepts:-

- **CIA triad (Confidentiality, Integrity, Availability).**

The CIA Triad is a foundational model in cybersecurity that defines the three core objectives of information security: Confidentiality, Integrity, and Availability. Every security control, policy, and mechanism in an organization is designed to support one or more of these principles.

1. Confidentiality
Definition
Confidentiality ensures that information is accessible only to authorized individuals, systems, or processes and is protected from unauthorized disclosure.
Why it is important
- Prevents sensitive data (passwords, personal information, financial records, intellectual property) from being exposed.
- Protects user privacy and helps organizations comply with laws and regulations (GDPR, HIPAA, etc.).

Threats to Confidentiality
- Unauthorized access
- Data breaches
- Eavesdropping and sniffing attacks
- Insider threats
- Phishing and social engineering
- Weak authentication

Controls and Mechanisms
- Authentication: Passwords, biometrics, MFA
- Authorization: Role-Based Access Control (RBAC), least privilege
- Encryption: Data at rest (disk encryption), data in transit (TLS/SSL)
- Network Security: Firewalls, VPNs
- Data Classification: Public, Internal, Confidential, Restricted

Example
- Only HR staff can access employee salary data.
- HTTPS encrypts data between a browser and a web server.

2. Integrity
Definition
Integrity ensures that data remains accurate, complete, and unaltered during storage, processing, and transmission.
Why it is important
- Prevents unauthorized or accidental modification of data.
- Ensures trust in systems and decision-making.
- Critical for financial systems, medical records, and logs.

Threats to Integrity

- Unauthorized data modification
- Malware and ransomware
- Man-in-the-middle (MITM) attacks
- SQL injection
- Human errors
- Software bugs

Controls and Mechanisms

- Hashing: MD5, SHA-256 (to detect changes)
- Digital Signatures: Verify authenticity and integrity
- Checksums
- Access Controls: Prevent unauthorized modification
- Version Control
- Audit Logs and Monitoring

Example

- A file hash changes if the file is tampered with.
- Database transaction logs ensure records are not altered improperly.

3. Availability

Definition

Availability ensures that systems, data, and services are accessible to authorized users when needed.

Why it is important

- Downtime affects business operations, productivity, and customer trust.
- Critical for services like banking, healthcare, and e-commerce.

Threats to Availability

- Denial of Service (DoS / DDoS) attacks
- Hardware failures
- Power outages
- Natural disasters
- Ransomware
- Network failures

Controls and Mechanisms

- Redundancy: Multiple servers, failover systems
- Backups: Regular, tested backups
- Disaster Recovery Plans (DRP)
- High Availability (HA) architectures
- Load Balancers
- Patch Management
- Monitoring and Incident Response

Example

- A website hosted on multiple servers remains online if one server fails.

- Cloud services with auto-scaling maintain uptime during traffic spikes.

Principle	Goal	Protects Against	Common Controls
Confidentiality	Prevent unauthorized access	Data breaches, spying	Encryption, RBAC, MFA
Integrity	Prevent unauthorized modification	Tampering, malware	Hashing, digital signatures
Availability	Ensure access when needed	DoS, outages	Backups, redundancy

Real-World Example (Banking System)

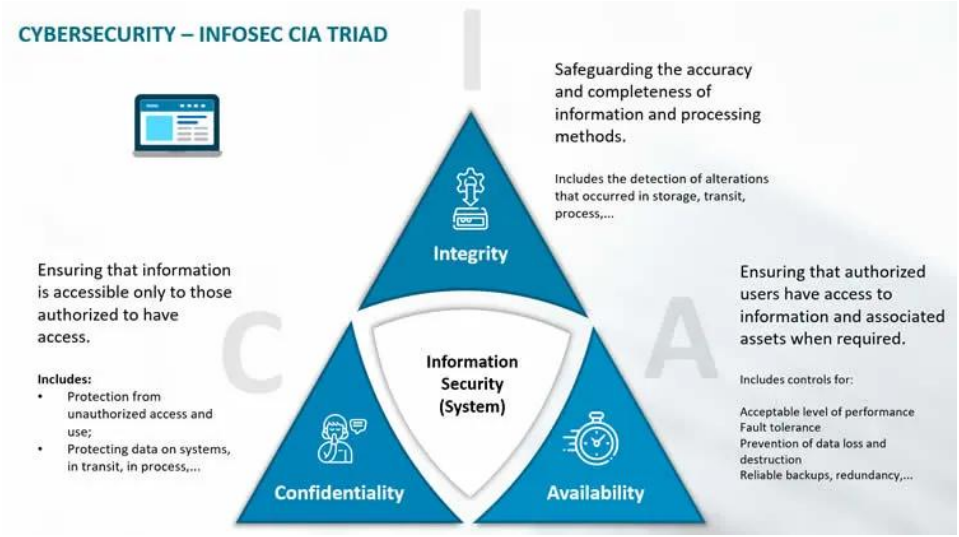
- **Confidentiality**: Customer data encrypted and accessible only to bank employees.
- **Integrity**: Transaction records protected with hashing and logs.
- **Availability**: ATM services run 24/7 with redundant servers.

Key Takeaway

The CIA Triad provides a balanced approach to cybersecurity. Focusing on only one element can weaken the system—for example:

- Strong confidentiality without availability can block legitimate users.
- High availability without integrity can spread corrupted data.

A secure system must maintain all three simultaneously.



- **Threat vs. vulnerability vs. risk.**

Threat

- A threat is anything that has the potential to cause harm to a system, network, or organization.
- It can be human, natural, or technical.
- Examples:
    - Hackers
    - Malware (ransomware, trojans)
    - Phishing attacks
    - Insider misuse
    - Natural disasters (fire, flood)

Vulnerability

- A vulnerability is a weakness or flaw in a system that can be exploited by a threat.
- Vulnerabilities exist due to:
    - Unpatched software
    - Weak passwords
    - Misconfigurations
    - Open ports or insecure services
- **Examples:**
    - Outdated Apache server
    - Default credentials
    - Missing security patches
    - SQL injection flaw

Risk

- Risk is the likelihood and impact of a threat exploiting a vulnerability.
- It answers the question:
  "What could go wrong, and how bad would it be?"
- Risk depends on:
    - Threat likelihood
    - Vulnerability severity
    - Impact on business or systems
- **Examples:**
    - High risk: Internet-facing server with critical vulnerability
    - Low risk: Vulnerability on an isolated test machine

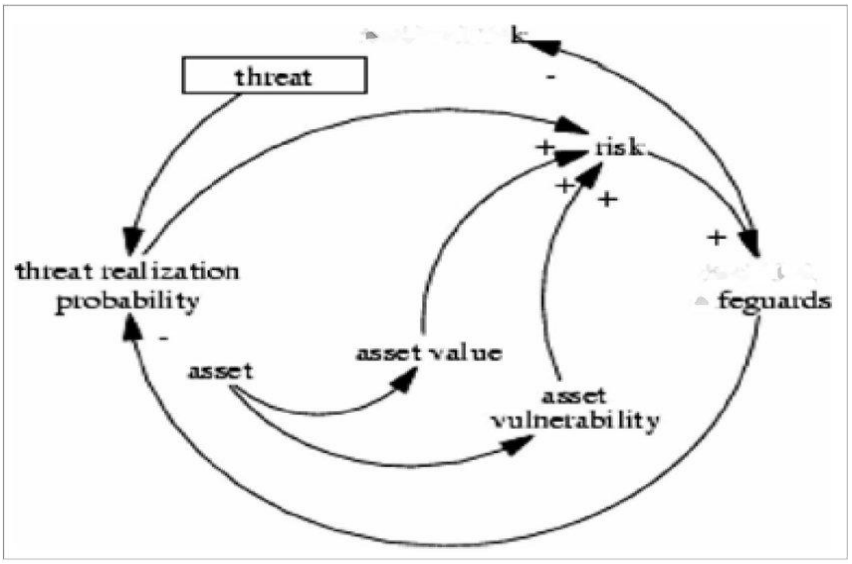
Risk = Threat × Vulnerability × Impact

Simple Real-World Example

- Threat: Hacker on the internet
- Vulnerability: Weak admin password
- Risk: Unauthorized access and data breach

Term	Meaning	Example
Threat	Potential cause of harm	Malware

Vulnerability	Weakness in system	Unpatched software
Risk	Chance of damage occurring	Data breach



- **Defense-in-depth, zero trust.**

Defense-in-Depth

Definition

Defense-in-Depth is a security strategy that uses **multiple layers of controls** so that if one layer fails, others still protect the system.

Key Layers

1. **Physical Security** – locks, CCTV, biometric access
2. **Network Security** – firewalls, IDS/IPS, VPN
3. **Perimeter Security** – web gateways, email filters
4. **Host/Endpoint Security** – antivirus, EDR
5. **Application Security** – secure coding, WAF
6. **Data Security** – encryption, DLP
7. **Identity & Access** – MFA, least privilege
8. **Monitoring & Response** – SIEM, SOC

Example

- Firewall blocks malicious IP
- IDS detects suspicious traffic
- EDR stops malware execution
- SIEM alerts SOC analyst

Benefits

- Reduces single points of failure
- Improves attack detection
- Limits attacker movement

Zero Trust

Definition

Zero Trust is a security model based on the principle:

"Never trust, always verify."

No user, device, or application is trusted by default — even inside the network.

Core Principles

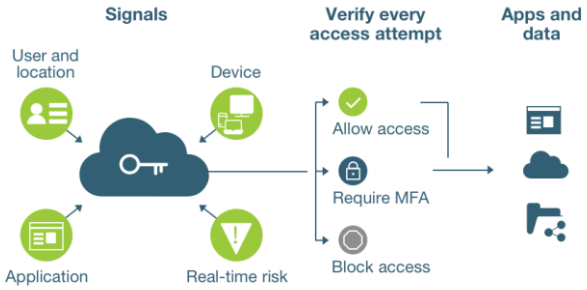
1. **Verify explicitly** – authenticate and authorize every request
2. **Least privilege access** – give minimum required access
3. **Assume breach** – design systems as if attackers are already inside
4. **Continuous monitoring** – verify access continuously

Key Components

- Multi-Factor Authentication (MFA)
- Identity-based access control
- Device posture checks
- Network micro-segmentation
- Continuous logging & monitoring

Example

- Employee logs in from new device
- MFA required
- Device health checked
- Access granted only to specific application
- Activity continuously monitored



Aspect	Defense-in-Depth	Zero Trust
Focus	Layered security controls	Identity-centric security

Trust Model	Partial trust inside network	No implicit trust
Network	Perimeter-based	Micro-segmented
Access	Role-based	Context-aware
SOC Use	Detect & contain attacks	Prevent lateral movement

Defense-in-Depth and Zero Trust are complementary security approaches that together provide strong protection against modern cyber threats. Defense-in-Depth ensures security through multiple layered controls, so that the failure of one control does not compromise the entire system. Zero Trust strengthens this model by removing implicit trust and continuously verifying every user, device, and access request, even within the internal network. When implemented together, these strategies help organizations reduce attack surfaces, prevent lateral movement, detect threats early, and respond effectively to breaches. This combined approach is essential for building a resilient and proactive cybersecurity posture in today's complex threat landscape.

- **Flashcards for terminology**

Flashcards for Cybersecurity Terminology (Anki-Style)
Flashcard 1
Q: What is a Threat?
A: A potential cause of harm, such as an attacker, malware, or natural disaster.
Flashcard 2
Q: What is a Vulnerability?
A: A weakness in a system that can be exploited by a threat.
Flashcard 3
Q: What is Risk?
A: The likelihood and impact of a threat exploiting a vulnerability.
Flashcard 4
Q: What is Defense-in-Depth?
A: A layered security approach using multiple controls to protect systems.
Flashcard 5
Q: What is Zero Trust?
A: A security model that trusts no user or device by default and verifies every access request.
Flashcard 6
Q: What is SIEM?
A: Security Information and Event Management used for log collection, correlation, and alerting.
Flashcard 7
Q: What is IDS vs IPS?
A: IDS detects attacks; IPS detects and actively blocks attacks.

Flashcard 8
Q: What is CVSS?
A: Common Vulnerability Scoring System used to rate vulnerability severity.
Flashcard 9
Q: What is Patch Management?
A: The process of applying updates to fix vulnerabilities.
Flashcard 10
Q: What is Least Privilege?
A: Granting users only the access necessary to perform their tasks.

- **Case studies**

  - Organization: Equifax
  - Impact: ~147 million people affected
  - Data Exposed: SSNs, birth dates, addresses, credit data

  Root Cause
  - A known vulnerability in Apache Struts (CVE-2017-5638)
  - Patch was available but not applied
  - Lack of effective vulnerability and patch management

  What Failed
  - Failure to apply critical security patches
  - Inadequate asset inventory
  - Weak monitoring and detection
  - Expired SSL certificate prevented alert visibility

  Threat–Vulnerability–Risk Mapping
  - Threat: External attackers
  - Vulnerability: Unpatched Apache Struts flaw
  - Risk: Massive data breach and identity theft

  Lessons Learned
  - Patch critical vulnerabilities immediately
  - Maintain accurate asset inventories
  - Implement continuous monitoring (SIEM)
  - Use Defense-in-Depth and Zero Trust principles
  - Regular vulnerability scanning (e.g., Nessus)

6. **Security Operations Workflow**
   Stages:
   1. Detection: Alerts from SIEM/EDR.

2. Triage: Prioritize based on severity.
3. Investigation: Correlate logs, IOC hunting.
4. Response: Containment, eradication.

- **Detection: Alerts from SIEM/EDR :-**
  Detection is the **first and most critical stage** of incident handling. It focuses on identifying suspicious or malicious activities as early as possible.
  Security events are detected using **SIEM (Security Information and Event Management)** and **EDR (Endpoint Detection and Response)** tools.
  - **SIEM** collects logs from multiple sources such as firewalls, intrusion detection systems, servers, applications, databases, and endpoints. It correlates these logs in real time to identify patterns that may indicate malicious behavior, such as repeated failed logins, unusual network connections, or policy violations. By applying correlation rules, threat intelligence, and behavioral analytics, SIEM generates alerts when abnormal or suspicious activity is detected.
  - **EDR** focuses specifically on endpoints such as desktops, laptops, and servers. It continuously monitors processes, file activity, registry changes, and memory usage to detect suspicious behaviors like malware execution, privilege escalation, lateral movement, or ransomware activity.
    Alerts generated by SIEM and EDR indicate **potential security incidents** and serve as triggers for SOC analysts to begin further analysis.

- **Triage: Prioritize based on severity:-**
  Triage is the process of quickly assessing and prioritizing alerts to determine which ones require immediate action.
  SOC analysts review alerts and prioritize them based on:
  - Severity (critical, high, medium, low)
  - Impact on business operations and data
  - Confidence level of detection accuracy

  During triage, analysts classify alerts as:
  - True Positive – confirmed malicious activity
  - False Positive – legitimate activity mistakenly flagged
  - Benign Activity – expected behavior that does not pose a risk

  High-severity alerts affecting critical assets such as production servers, databases, or privileged accounts are handled first. Low-risk or informational alerts may be deprioritized, monitored, or closed to reduce alert fatigue and focus resources on real threats.

- **Investigation :-**
  Investigation involves deep analysis to understand the nature, scope, and impact of the incident.

SOC analysts correlate logs from multiple sources using the SIEM to build a complete picture of the event. This includes:

- Reviewing authentication logs, system logs, network traffic, and application logs
- Performing IOC (Indicator of Compromise) hunting, such as checking suspicious IP addresses, file hashes, domain names, URLs, and registry keys
- Analyzing endpoint telemetry from EDR to understand process execution, file modifications, and user actions

During this phase, analysts determine:

- How the attack started (initial access)
- What systems and accounts are affected
- Whether lateral movement or data exfiltration occurred
- The attack timeline, from entry point to detection

- **Response**

  Response focuses on stopping the attack, removing the threat, and restoring normal operations.

  - Containment: Immediate actions are taken to limit damage, such as isolating infected systems, blocking malicious IP addresses or domains, disabling compromised user accounts, and stopping malicious processes.
  - Eradication: The root cause of the incident is addressed by removing malware, deleting malicious files, patching exploited vulnerabilities, closing exposed services, and resetting credentials.
  - Recovery: Systems are restored to normal operation using clean backups, monitoring is increased to ensure no reinfection occurs, and services are gradually brought back online.
  - Lessons Learned: After recovery, the incident is reviewed to identify gaps in detection and response. Detection rules are updated, security controls are improved, and procedures are refined to prevent similar incidents in the future.

- **Conclusion :-**

  This structured SOC incident handling process—Detection, Triage, Investigation, and Response—ensures that security incidents are identified early, prioritized effectively, investigated thoroughly, and resolved efficiently. By following this approach, SOC teams can minimize business impact, reduce attacker dwell time, prevent recurrence, and continuously strengthen the organization's overall security posture. analysis to understand the nature, scope, and impact of the incident.

Simulate workflows using incident response platforms :-

Purpose of Simulation

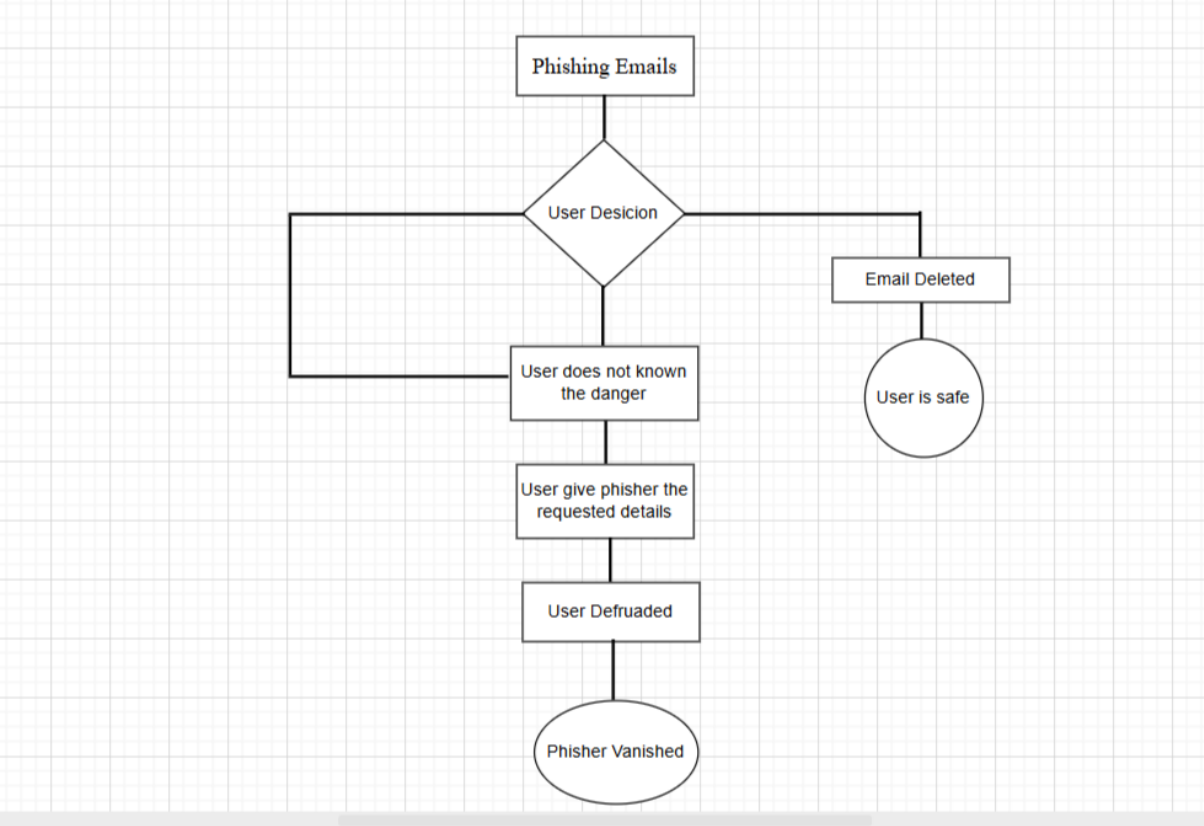
Incident response platforms like TheHive are used to:

- Standardize incident handling
- Track alerts, tasks, evidence, and actions
- Improve coordination between SOC analysts
- Practice real-world incident workflows safely

How TheHive Is Used in a Workflow

- An alert (e.g., phishing email) is ingested from SIEM or email security tools.
- The alert is converted into a case in TheHive.
- The case is assigned to an analyst.
- Investigation tasks (IOC analysis, email header review, user validation) are tracked.
- Response actions (blocking sender, resetting passwords) are documented.
- The case is closed with lessons learned.

Create a flowchart for a phishing email incident :-



7. Incident Response Basics

   IR Lifecycle

1. Preparation
Goal: Be ready before an incident occurs.
Key Activities:

- Develop Incident Response policies and playbooks
- Form an IR team and define roles & responsibilities
- Deploy security tools (SIEM, EDR, IDS/IPS, SOAR)
- Conduct employee security awareness training
- Maintain asset inventory and network diagrams

Output: IR plan, trained team, monitoring tools

2. Identification (Detection & Analysis)
Goal: Identify and confirm a security incident.
Key Activities:

- Monitor alerts from SIEM, EDR, IDS, email gateways
- Analyze logs, network traffic, and alerts
- Validate whether the event is a true incident
- Determine incident type (phishing, malware, DDoS, insider threat)
- Assess severity and scope

Output: Confirmed incident, incident classification

3. Containment
Goal: Limit the spread and impact of the incident.
Types of Containment:

- **Short-term:** Isolate affected systems, block IPs/domains
- **Long-term:** Apply patches, temporary fixes

Key Activities:

- Disconnect infected machines from the network
- Disable compromised accounts
- Block malicious indicators (IOCs)
- Preserve evidence for forensics

Output: Incident impact controlled

4. Eradication
Goal: Remove the root cause of the incident.
Key Activities:

- Remove malware or malicious files
- Close exploited vulnerabilities
- Patch systems and update configurations
- Remove unauthorized user accounts
- Improve detection rules

Output: Threat fully removed

5. Recovery
Goal: Restore systems to normal operations.
Key Activities:

- Restore systems from clean backups
- Monitor systems for re-infection
- Validate system integrity
- Bring systems back online gradually

Output: Normal business operations restored

6. Lessons Learned (Post-Incident Activity)
Goal: Improve future incident handling.
Key Activities:

- Conduct post-incident review meeting
- Document timeline, actions taken, and impact
- Identify gaps in tools, processes, or training
- Update IR playbooks and controls

Output: Incident report, improved security posture

IR Lifecycle Flow

Preparation
↓
Identification
↓
Containment
↓
Eradication
↓
Recovery
↓
Lessons Learned

Phishing Incident

- **Identification:** SIEM alerts on suspicious email
- **Containment:** Block sender domain, isolate affected user
- **Eradication:** Remove malicious payload, reset credentials
- **Recovery:** Restore user access
- **Lessons Learned:** Improve email filtering rules

- **Preparation → Identification → Containment → Eradication → Recovery → Lessons Learned.**

1. Preparation
Definition:
Preparation is the phase where an organization establishes policies, tools, trained personnel, and procedures to respond effectively to security incidents before they occur.
Purpose:
Get ready before an incident happens.
Activities:

- Create Incident Response (IR) policies and playbooks
- Define IR team roles and escalation paths
- Deploy security tools (SIEM, EDR, IDS/IPS, SOAR)
- Conduct security awareness training
- Maintain asset inventory and regular backups

Outcome:
The organization is ready to respond quickly and effectively to incidents.

2. Identification
Definition:
Identification is the process of detecting, analyzing, and confirming that a security event is a real incident and determining its scope and severity.
Purpose:
Detect and confirm a security incident.
Activities:

- Monitor alerts from SIEM, EDR, firewall, and email security tools
- Analyze logs and suspicious activities
- Distinguish real incidents from false positives
- Identify the incident type and severity

Outcome:
A confirmed incident with known scope and impact.

3. Containment

Definition:
Containment involves actions taken to limit the spread and impact of an incident while preventing further damage.
Purpose:
Stop the incident from spreading.
Activities:

- Isolate infected or affected systems
- Disable compromised user accounts
- Block malicious IP addresses, domains, and file hashes
- Preserve evidence for forensic analysis

Outcome:
The damage caused by the incident is limited and controlled.

4. Eradication
Definition:
Eradication is the process of completely removing the root cause of the incident from the environment.
Purpose:
Remove the root cause of the incident.
Activities:

- Remove malware and malicious files
- Patch exploited vulnerabilities
- Remove backdoors and unauthorized access
- Update security rules and detection controls

Outcome:
The threat is completely eliminated from the system.

5. Recovery
Definition:
Recovery focuses on restoring affected systems and services to normal operations in a secure and monitored manner.
Purpose:
Restore normal operations safely.
Activities:

- Restore systems from clean and verified backups
- Monitor systems for abnormal or suspicious behavior
- Validate system integrity and functionality
- Return systems to production gradually

Outcome:
Business operations are fully restored.

6. Lessons Learned

Definition:

Lessons Learned is the post-incident phase where the organization reviews the incident to improve future detection, response, and prevention.

Purpose:

Improve future incident response.

Activities:

- Conduct post-incident review meetings
- Document the incident timeline, actions taken, and impact
- Identify gaps in tools, processes, and procedures
- Update IR plans, playbooks, and training programs

Outcome:

Improved security posture and stronger incident response readiness.

- **NIST SP 800-61 framework**

1. Preparation

Definition:

Preparation involves establishing the policies, procedures, tools, and trained personnel required to effectively respond to security incidents.

Key Activities:

- Develop Incident Response policies and plans
- Define roles, responsibilities, and escalation paths
- Deploy security tools (SIEM, EDR, IDS/IPS, SOAR)
- Conduct training and awareness programs
- Maintain asset inventory and backups

Goal:

Ensure the organization is ready to handle incidents efficiently.

2. Detection and Analysis

Definition:

This phase focuses on identifying potential security incidents, analyzing alerts, and determining whether an event is a real incident.

Key Activities:

- Monitor alerts from SIEM, IDS/IPS, EDR, firewalls
- Analyze logs, network traffic, and endpoints
- Validate true incidents vs false positives
- Classify incident type and assess severity
- Determine scope and impact

Goal:

Quickly identify and understand security incidents.

3. Containment, Eradication, and Recovery

Definition:

This combined phase focuses on limiting the damage, removing the threat, and restoring systems to normal operations.

Containment

- Isolate affected systems
- Block malicious IPs, domains, and accounts
- Preserve evidence

Eradication

- Remove malware and backdoors
- Patch vulnerabilities
- Eliminate unauthorized access

Recovery

- Restore systems from clean backups
- Monitor systems for reinfection
- Return systems to production

Goal:

Stop the attack, remove the threat, and safely resume operations.

4. Post-Incident Activity (Lessons Learned)

Definition:

This phase focuses on reviewing the incident to improve future incident response capabilities.

Key Activities:

- Conduct post-incident review meetings
- Document incident timeline and actions taken
- Identify gaps in tools, processes, and training
- Update policies, playbooks, and controls

Goal:

Strengthen security posture and prevent recurrence.

NIST SP 800-61 Lifecycle Flow

Preparation
↓
Detection & Analysis
↓
Containment → Eradication → Recovery
↓
Post-Incident Activity

Tabletop exercises

Tabletop Exercises (TTX)

Definition
Tabletop exercises are discussion-based simulation exercises where incident response team members walk through a hypothetical cyber incident scenario to test preparedness, decision-making, communication, and response procedures without affecting live systems.

Purpose of Tabletop Exercises

- Validate the Incident Response (IR) plan
- Test roles, responsibilities, and escalation paths
- Improve coordination and communication
- Identify gaps in policies, tools, and training
- Build confidence before a real incident occurs

How Tabletop Exercises Work

- A scenario (e.g., phishing, ransomware, data breach) is presented
- Participants discuss actions they would take at each IR phase
- No real systems are touched (discussion only)
- A facilitator guides the exercise and injects new events

Key Participants

- SOC analysts
- Incident Response team
- IT and system administrators
- Management and legal teams
- Communications / PR team

Typical Tabletop Exercise Flow

1. Scenario introduction
2. Incident detection discussion
3. Decision-making and escalation
4. Containment and remediation discussion
5. Communication and reporting decisions
6. Post-incident review

Example Scenario
Ransomware Attack

- User reports encrypted files
- SOC detects abnormal activity
- Discuss isolation of systems
- Decide on backup restoration
- Evaluate legal and notification requirements

Benefits of Tabletop Exercises

- Low-risk and cost-effective
- Improves response time
- Enhances teamwork
- Strengthens compliance (NIST, ISO 27001)
- Reveals weaknesses before attackers do

Frequency

- Recommended at least once or twice per year
- After major changes (new tools, infrastructure, or staff)

Mapping to NIST SP 800-61

- Falls under Preparation and Post-Incident Activity
- Helps refine playbooks and response strategies

8. Documentation Standards :-

1.Incident reports

Definition:

A formal record that describes a security incident, including what happened, impact, response actions, and lessons learned.

Used for:

- Management reporting
- Legal/compliance evidence
- Post-incident review

2. Runbook

Definition:

A step-by-step technical guide that explains how to respond to a specific type of incident.

Used for:

- DDoS attacks
- Phishing emails
- Malware infections

3. SOP (Standard Operating Procedure)

Definition:

A documented set of instructions describing how routine security tasks are performed.

Used for:

- Log monitoring
- Alert triage

- User access reviews

4. Post-Mortem (Lessons Learned)
Definition:
A review document created after an incident is resolved, focusing on what went wrong, what went well, and how to improve.
Used for:

- Process improvement
- Preventing recurrence

5**. Using Templates** (SANS Incident Handler's Handbook)
The SANS Incident Handler's Handbook provides widely accepted templates that include:

- Incident summary
- Timeline
- Indicators of compromise (IOCs)
- Response actions
- Impact analysis
- Lessons learned

Using templates ensures no critical information is missed.

- **Mock Incident Report – DDoS Attack**
  1.Incident Overview

  Incident Type: Distributed Denial-of-Service (DDoS)
  Date Detected: 20 December 2025
  Time Detected: 14:35 IST
  Reported By: SOC Monitoring System (SIEM)
  Severity: High

  2. Incident Definition
  A Distributed Denial-of-Service (DDoS) attack is a cyberattack in which multiple compromised systems send excessive traffic to a target server or network, causing service disruption or complete outage.

  3. Incident Description
  At 14:35 IST, the SOC detected an abnormal spike in inbound traffic targeting the company's public web server.
  The traffic originated from multiple IP addresses across different geographic regions, indicating a DDoS attack.

4. Detection Method

- SIEM alert triggered due to:
  - Sudden traffic surge (10x normal baseline)
  - Repeated requests to HTTP port 80
- Firewall logs confirmed high-volume traffic from multiple sources

5. Impact Assessment

- Public website became unavailable for 18 minutes
- No data breach or data loss detected
- Internal systems remained unaffected
- Customer access temporarily disrupted

6. Response Actions Taken

- SOC analyst validated the alert
- Incident escalated to the Incident Response Team
- Rate limiting enabled on the firewall
- Malicious IPs blocked temporarily
- Traffic rerouted via DDoS protection service
- Continuous monitoring performed until traffic normalized

7. Indicators of Compromise (IOCs)

- Multiple source IP addresses
- High-frequency HTTP GET requests
- Abnormal traffic patterns

8. Resolution

The attack traffic subsided after mitigation controls were applied.
Normal service was restored at 14:53 IST.

9. Lessons Learned

- Early detection minimized downtime
- Automated DDoS mitigation should be enabled permanently
- Improve alert thresholds for traffic anomalies

10. Recommendations

- Implement always-on DDoS protection
- Update incident response runbook
- Conduct DDoS tabletop exercises quarterly

1. Log Analysis Practice

1. Event ID 4625 – Failed Logon
Definition

- Event ID 4625 is generated when a user login attempt fails in Windows.

How to Filter Event ID 4625

- Open Event Viewer
- Navigate to:
  - Windows Logs → Security
- Click Filter Current Log
- Set:
  - Logged: Any time
  - Event IDs: 4625
  - Event sources: Leave empty
- Click OK

Key Fields to Analyze (4625)

- Account Name – Username that failed to log in
- Logon Type
  - 2 → Local login
  - 3 → Network login
  - 10 → Remote Desktop (RDP)
- Failure Reason – Bad password, unknown user, locked account
- Source Network Address – IP address of the source system

Brute-Force Indicators (4625)

- Multiple 4625 events in a short time
- Same source IP attempting many logins
- Repeated attempts on one or multiple accounts
- Logon Type 3 or 10 frequently observed

2. Event ID 7045 – New Service Creation
Definition

- Event ID 7045 is generated when a new Windows service is installed on the system.

How to Filter Event ID 7045

- Open Event Viewer
- Navigate to:
  - Windows Logs → System
- Click Filter Current Log
- Set:
  - Event IDs: 7045
- Click OK

Key Fields to Analyze (7045)

- Service Name – Name of the newly created service
- Service File Name – Executable path
- Service Account – Account running the service
- Start Type – Automatic or Manual

Suspicious Indicators (7045)

- Random or unfamiliar service names
- Executables located in:
  - Temp folders
  - Users\Public
- Services created by non-admin users
- Service creation shortly after failed logins

3. Identifying Brute-Force Attacks (Correlation)

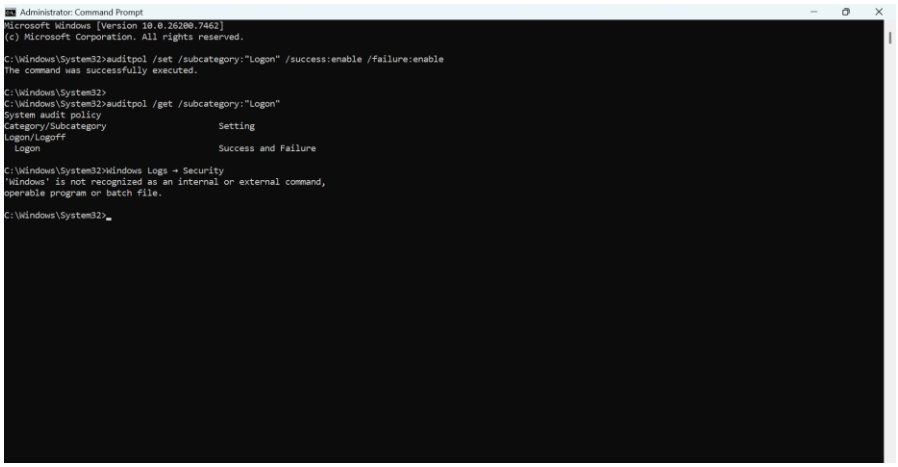
- Multiple 4625 events detected
- Followed by successful login (4624)
- Followed by 7045 service creation
- Indicates possible:
  - Brute-force attack
  - Successful compromise
  - Persistence mechanism

4. SOC Analyst Response (High-Level)

- Identify attacking IP address
- Block IP at firewall
- Disable or reset compromised account
- Investigate newly created services
- Remove malicious services
- Document incident

5. Quick Exam / Interview Points

- 4625 → Failed login attempt
- 7045 → New service creation
- Multiple 4625 events → Brute-force attempt
- 4625 + 7045 → Possible system compromise
- Windows Event Viewer is key for host-based detection



- **Browser History Analysis**

1. Definition

Browser History Analysis is the process of examining web browser artifacts to identify:

- Malicious URLs
- Phishing sites
- Command-and-control (C2) communication
- User activity related to security incidents

Chrome stores browsing data in a **SQLite database**, which can be analyzed using **Eric Zimmerman's forensic tools**.

2. Why Use Eric Zimmerman's Tools

- Free and widely used in **digital forensics & incident response (DFIR)**
- Parses browser artifacts automatically
- Supports Chrome, Edge, Firefox
- Provides readable timelines and reports

3. Chrome History File Location

- **Windows Path:**

C:\Users\<Username>\AppData\Local\Google\Chrome\User Data\Default\History

4. Tools Used (Eric Zimmerman)

- BrowsingHistoryView or Hindsight
- SQLite tools (optional)
- Timeline Explorer

5. Steps to Parse Chrome History

Step 1: Close Chrome Browser

- Ensure Chrome is not running
- Otherwise, history file will be locked

Step 2: Copy History File

- Navigate to Chrome history path
- Copy the **History** file to another folder (e.g., Desktop)

Step 3: Run Eric Zimmerman's Tool

- Launch **Hindsight** (recommended)
- Select **Chrome** as browser type
- Point the tool to the copied **History** file

Step 4: Generate Output

- Tool parses:
  - URLs
  - Visit counts
  - Timestamps
  - Referrers
- Output produced in:
  - CSV
  - JSON
  - SQLite

Step 5: Analyze Results

- Open output in:
  - Excel
  - Timeline Explorer
- Sort by:
  - Visit count
  - Last visit time
  - URL domain

6. Identifying Malicious URLs

Indicators of Malicious Activity

- URLs with:
  - Random strings
  - Shortened links (bit.ly, tinyurl)
  - Suspicious TLDs (.ru, .xyz, .top)
- Domains newly registered
- URLs accessed shortly before malware infection
- Repeated access to same suspicious domain

7. Correlation with Threat Intelligence

- Compare URLs against:
  - VirusTotal
  - AbuseIPDB
  - Threat intel feeds
- Identify:
  - Phishing
  - Malware download sites
  - C2 communication

8. SOC / DFIR Use Case

- Investigate phishing incidents
- Trace initial access vector
- Identify user-click behavior
- Support incident timelines

9. Output Documentation

- Record:
  - Malicious URL
  - Timestamp
  - Browser profile
  - User account
- Include findings in:
  - Incident report
  - Post-mortem analysis

Brute-Force Detection

1. Objective

- Detect multiple **failed login attempts**
- Identify **brute-force attack behavior**
- Collect **forensic evidence**

2. Generate Failed Logins

- Start **Windows VM**
- Lock screen using **Win + L**
- Enter **wrong password** multiple times (5–15 attempts)
- This creates **failed authentication events**

3. Open Event Viewer

- Press **Win + R**
- Type eventvwr.msc
- Press **Enter**

4. Navigate to Security Logs

- Go to:
- Windows Logs → Security
- Security log stores **authentication events**

5. Filter Event ID 4625

- Click **Filter Current Log**
- In the box showing **<All Event IDs>**, type:
- 4625
- Leave **Event sources** empty
- Click **OK**

6. Verify Failed Login Events

- Confirm:
  - **Event ID:** 4625
  - **Message:** An account failed to log on
- Check important fields:
  - Account Name
  - Failure Reason (bad password)
  - Logon Type (2, 3, or 10)
  - Source Network Address
  - Time Created

7. Identify Brute-Force Pattern

- Multiple failures for the **same user**
- Repeated attempts from the **same IP**
- Attempts within a **short time period**
- No successful login in between

8. Export Logs to CSV

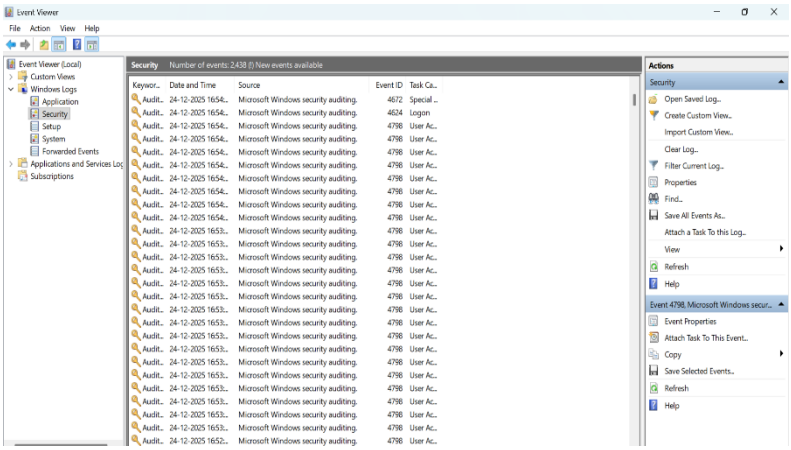
- Right-click **Security**
- Select **Save Filtered Log File As**
- Choose format: **CSV**
- Save as failed_logins_4625.csv

9. (Optional) CLI Export

- Open **Command Prompt (Admin)**
- Run:
- wevtutil qe Security "/q:*[System[(EventID=4625)]]" /f:csv > failed_logins.csv

10. Outcome

- Failed logins detected
- Brute-force activity identified
- Evidence exported for investigation
- Logs ready for SIEM or report



Conclusion

Brute-force attacks can be effectively detected by monitoring repeated failed login attempts in Windows Security logs.
Event ID 4625 provides critical details such as username, source IP, and failure reason for each failed authentication.
Filtering and exporting these events helps identify attack patterns and preserve forensic evidence.
Early detection enables faster response and strengthens overall system security.

- **Zimmerman Tools Practice**

Purpose

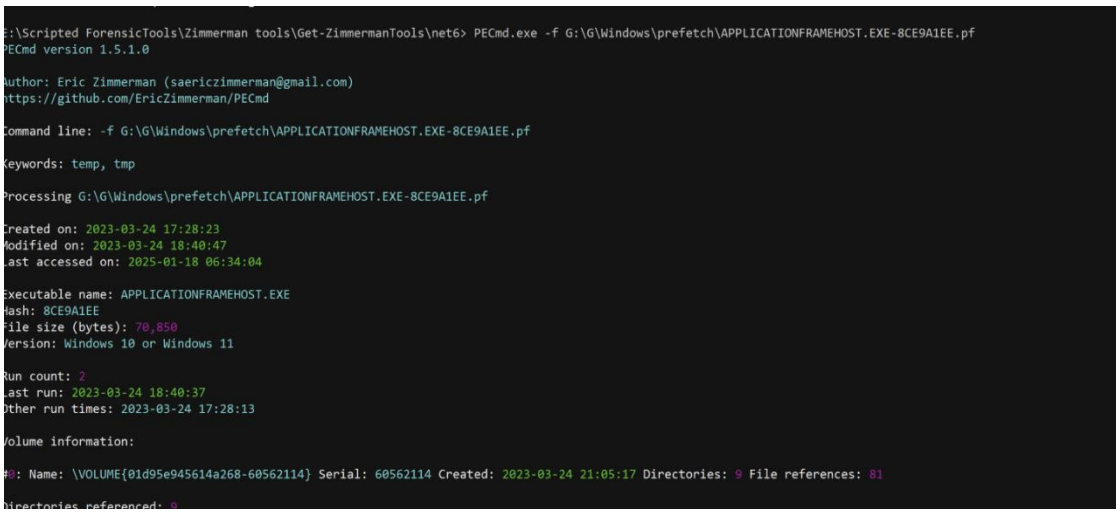
- Zimmerman Tools are used in **digital forensics** to analyze system and browser artifacts.
- **LECmd** helps analyze browser-related evidence such as **Chrome history**.
- It is commonly used in **incident response** and **user activity investigations**.

Practice Steps (Conceptual)

- Chrome browser history is collected from the user profile directory.
- LECmd parses the history database into readable forensic data.
- The output is reviewed to identify visited websites.
- A known test URL (e.g., http://test.com) is searched to confirm browsing activity.
- Extracted data supports **timeline analysis and evidence validation**.

- **Minimal Commands Used**
  sudo apt install mono-runtime
  mono LECmd.exe -h
  mono LECmd.exe -f History

- **Conclusion :-**
  Zimmerman Tools simplify the analysis of forensic artifacts by converting complex system data into a readable and structured format.
  LECmd plays an important role in examining browser-related artifacts, especially Chrome browsing history.
  It helps investigators identify visited websites, timestamps, and user activity patterns efficiently.
  By analyzing browser data, investigators can reconstruct user behavior during an incident.
  This practice improves understanding of digital evidence collection and interpretation.
  Overall, it builds strong hands-on skills in digital forensics and incident response.



- **Document Security Events**

1. What Is a Security Event?

A **security event** is any observable action on a system or network that is relevant to security. Examples include:

- Failed login attempts
- Successful logins outside business hours
- Malware detections
- Unauthorized access attempts

Not all security events are incidents, but **every incident starts as a security event**.

2. Purpose of Documenting Security Events

Documenting security events helps to:

- Maintain **accountability and traceability**
- Support **incident response and forensic investigations**
- Meet **compliance and audit requirements**
- Improve **future detection and prevention**

3. Security Event Documentation Template (With Definitions)

Template Fields

Date/Time | Source IP | Event ID | Description | Action Taken

Field Definitions

Date/Time

- **Definition:** The exact date and time when the event was detected or logged by the system.
- **Why it matters:** Helps build timelines and correlate events.
- **Format:** YYYY-MM-DD HH:MM (Local time or UTC)

Source IP

- **Definition:** The IP address from which the activity originated.
- **Why it matters:** Identifies the attacker or system generating the event.
- **Example:** 192.168.1.10

Event ID

- **Definition:** A unique numeric identifier assigned by the operating system or application to represent a specific event type.
- **Why it matters:** Allows precise filtering and identification of events.
- **Example:**
  - Windows Failed Login → **Event ID 4625**

Description

- **Definition:** A detailed explanation of what happened, how it was detected, and why it is relevant.

- **Why it matters:** Provides context for analysts and auditors.
- **Best practice:** Describe facts, not assumptions.

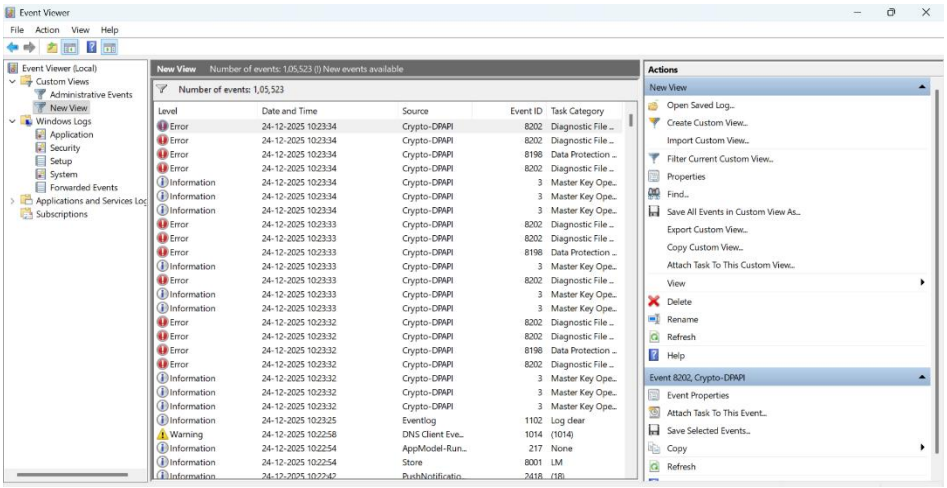
Action Taken

- **Definition:** Steps taken after identifying the event.
- **Why it matters:** Shows response effort and accountability.
- **Examples:** Monitoring, blocking IP, escalating to SOC.

4. Performing the Task Using Commands (Windows)

Scenario

Multiple failed login attempts from the same IP address.



After reviewing the command output from the Security Event logs, it was observed that multiple failed login attempts were recorded on the system. The events correspond to **Event ID 4625**, which indicates unsuccessful authentication attempts in Windows. All the failed login entries originated from the same **source IP address (192.168.1.10)** within a short time interval. This repeated failure pattern suggests suspicious behavior and may indicate a **brute-force login attempt**. The timestamps in the logs confirm that the attempts occurred consecutively, strengthening the likelihood of unauthorized access attempts rather than a user error.

Based on this observation, the event was documented using the security event template, and appropriate actions such as monitoring the IP address and escalating the alert to the security team were initiated.

4. Set Up Monitoring Dashboards

What Is a Monitoring Dashboard

A **monitoring dashboard** is a visual interface that displays security data in the form of charts, tables, and graphs. It helps security analysts quickly understand:

- Who is generating alerts
- What type of events are occurring
- How frequently critical events happen

Dashboards are commonly created in tools like **Kibana (Elastic Stack)** or **Grafana**.

Purpose of Security Monitoring Dashboards

Monitoring dashboards are used to:

- Detect threats early
- Reduce alert fatigue
- Identify attack patterns
- Support SOC decision-making
- Provide real-time security visibility

Tool Overview

Kibana

- Used with Elasticsearch
- Best for log analysis and SIEM use cases
- Supports Sigma rules and prebuilt detections

Grafana

- Used for metrics and logs
- Can connect to Elasticsearch, Prometheus, Loki
- Strong visualization capabilities

Step 1: Prepare the Data Source

Before creating dashboards:

- Logs must be ingested into Elasticsearch or another backend
- Logs may include:
  - Windows Security Events
  - Firewall logs
  - IDS/IPS alerts
  - SIEM alerts

Example fields used:

- source.ip
- event.code (Event ID)
- event.severity
- @timestamp

Step 2: Create Visualization – Top 10 Source IPs Generating Alerts

Definition
This visualization shows the **most active IP addresses** triggering security alerts. It helps identify:

- Brute-force attacks
- Scanning activity
- Malicious hosts

In Kibana (Conceptual Steps)

1. Open **Kibana → Visualize / Lens**
2. Select your **security index** (e.g., logs-*)
3. Choose a **Bar Chart or Table**
4. Set:
    - **Metric:** Count of events
    - **Bucket:** Source IP (source.ip)
5. Sort by **highest count**
6. Limit results to **Top 10**

Description (After Visualization Output)
The visualization displays the top 10 source IP addresses responsible for generating the highest number of security alerts. IP address **192.168.1.10** appears frequently, indicating repeated suspicious activity. This helps analysts quickly identify potentially malicious hosts and prioritize investigation efforts.

Step 3: Create Visualization – Frequency of Critical Event IDs
Definition
This visualization shows how often **critical or high-risk Event IDs** occur over time.
Examples:

- Event ID 4625 – Failed login
- Event ID 4688 – New process created
- Event ID 4719 – Audit policy change

In Kibana (Conceptual Steps)

1. Create a **Line Chart or Bar Chart**
2. Set:
    - **X-axis:** Time (@timestamp)
    - **Y-axis:** Count of events
3. Apply filter:
    - event.code: 4625
    - OR severity = critical
4. Group by **Event ID**

Description (After Visualization Output)

The chart shows the frequency of critical Event IDs over time. A spike in **Event ID 4625** indicates multiple failed login attempts within a short duration, suggesting possible brute-force activity. Monitoring this trend allows early detection of authentication-related attacks.

Step 4: Use Pre-Built Dashboards (Sigma Detection Rules)
What Are Sigma Rules?
Sigma is a generic detection rule format used to identify suspicious behavior across different SIEM platforms.

Why Use Pre-Built Dashboards?

- Saves time
- Based on community-tested detections
- Covers common attack techniques (MITRE ATT&CK)

How It Works (Conceptually)

1. Import Sigma rules into your SIEM
2. Rules generate alerts when conditions match
3. Pre-built dashboards automatically visualize:
    - Alert counts
    - Severity levels
    - Tactics and techniques

Description (After Using Pre-Built Dashboards)
Pre-built dashboards based on Sigma detection rules provide immediate visibility into common attack patterns such as credential abuse and lateral movement. These dashboards reduce manual effort and ensure standardized threat detection across the environment.

Step 5: Final Dashboard Layout (SOC Best Practice)
A complete SOC dashboard typically includes:

- Top alert-generating IPs
- Event ID frequency charts
- Severity distribution
- Timeline of alerts
- Detection rule hit counts

Outcome of This Activity
After setting up monitoring dashboards:

- Analysts can quickly identify threats
- Security events are visually correlated
- Response time is reduced
- Security posture visibility improves

4. **Configure Alert Rules**

Objective of the Custom Alert Rule
Definition
A custom alert rule in Wazuh is a user-defined detection logic that extends default security rules.
The objective of this rule is to identify SSH brute-force attacks, where an attacker repeatedly tries different passwords to gain unauthorized access.
Why this is important

- SSH is a common remote access service
- Repeated failed login attempts indicate credential-guessing attacks
- Early detection helps prevent system compromise

2. Understanding the Detection Logic
Condition to detect

- Event type: Failed SSH login
- Number of attempts: 3 or more
- Time window: 2 minutes (120 seconds)

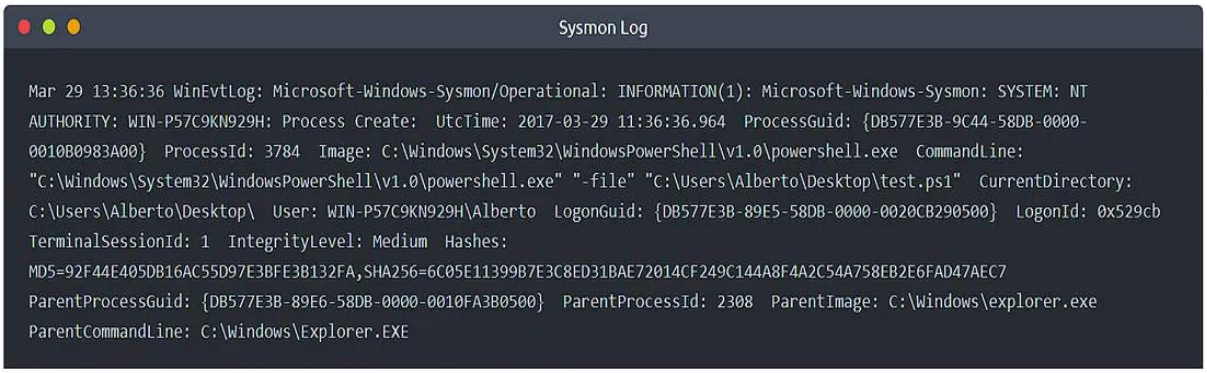
How Wazuh detects this

- Wazuh already detects individual SSH failed logins
- The custom rule correlates multiple failures
- When the frequency threshold is reached, an alert is generated

3. Create the Custom Wazuh Rule
File used

/var/ossec/etc/rules/local_rules.xml
This file is used for user-defined rules and is not overwritten during updates.
Rule Configuration Explanation

```
<rule id="100100" level="10">
  <if_matched_sid>5716</if_matched_sid>
  <frequency>3</frequency>
  <timeframe>120</timeframe>
  <description>SSH Brute Force: 3 failed logins in 2 minutes</description>
</rule>
```



Detailed Explanation of Each Field

Field	Description
rule id="100100"	Unique identifier for the custom rule
level="10"	Alert severity level (High risk)
if_matched_sid	Refers to the existing Wazuh rule that detects a single SSH failed login
frequency	Number of times the base rule must trigger
timeframe	Time window in seconds (120 seconds = 2 minutes)
description	Human-readable alert message

4. Apply the Rule (Wazuh Restart)
Purpose
After adding a rule, Wazuh must reload its configuration so the rule becomes active.
What happens internally

- Wazuh parses the rule file
- Loads correlation logic
- Starts monitoring SSH events in real time

5. Testing the Custom Alert Rule
Test Method
Manually generate failed SSH login attempts.

Test Command
ssh wronguser@192.168.1.x
Test Procedure

1. Attempt to log in using a non-existent or wrong username
2. Enter an incorrect password
3. Repeat this 3 times within 2 minutes

Why this works

- Each failed attempt is logged in /var/log/auth.log
- Wazuh detects each failure
- The correlation rule counts the attempts
- On the 3rd failure, the alert is triggered

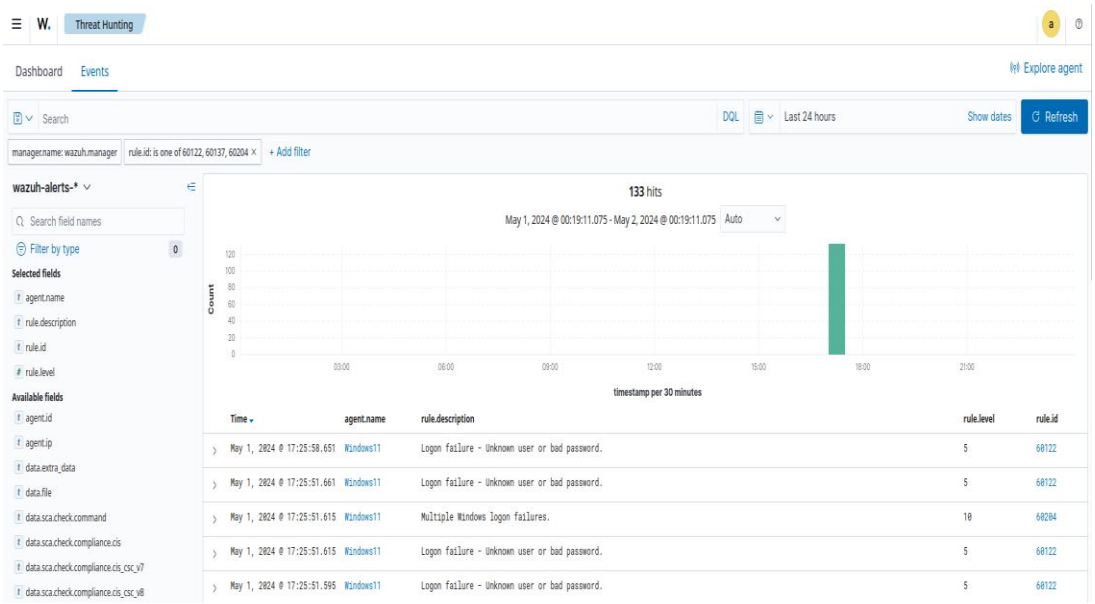
6. Alert Generation and Detection

What Wazuh Detects

- Multiple authentication failures
- Same source IP
- Short time interval

Alert Details

- Rule ID: 100100
- Severity Level: 10
- Attack Type: SSH Brute Force
- Technique: Credential Access (MITRE T1110)



7. Alert Validation in Wazuh Dashboard

Validation Process

1. Open Wazuh Dashboard
2. Navigate to Security Events
3. Filter by:
    - Rule ID: 100100
    - Group: authentication failures
4. Verify:
    - Alert timestamp
    - Source IP address
    - Description message

Expected Result
An alert stating:
“SSH Brute Force: 3 failed logins in 2 minutes”

8. Effectiveness of the Rule
Security Benefits

- Detects brute-force attacks early
- Reduces noise from single failed attempts
- Improves visibility of authentication abuse

Accuracy

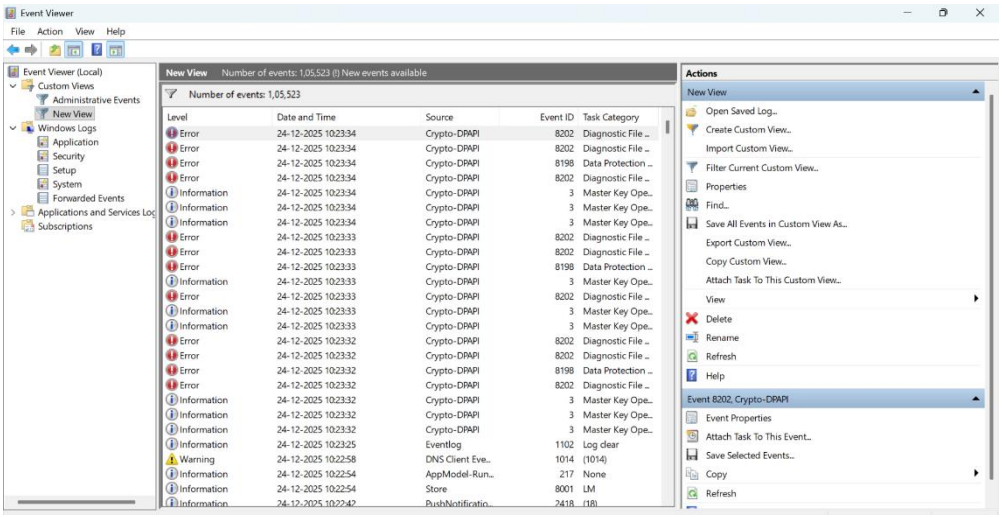
- Frequency-based detection minimizes false positives
- Time-bound logic ensures realistic attack detection

9. Documentation Example (Lab / Exam Ready)

Field	Value
Event Type	SSH Brute Force
Detection Method	Custom Wazuh Correlation Rule
Threshold	3 failed attempts
Time Window	2 minutes
Rule ID	100100
Status	Successfully Detected
Response	Alert generated and reviewed

10. Conclusion

This custom Wazuh rule successfully detects repeated failed SSH login attempts, which are a strong indicator of a brute-force attack. By correlating events within a defined time window, Wazuh provides accurate and actionable security alerts, strengthening system defense.



Alert Validation
Alert Validation in Wazuh
Objective:
To confirm that the custom Wazuh rule detecting **3 or more failed SSH logins within 2 minutes** is working correctly and to document how effective the rule is in identifying brute-force attacks.

1. Purpose of Alert Validation
Definition
Alert validation is the process of verifying that a security rule:

- Triggers under the correct conditions
- Produces accurate alerts
- Provides meaningful and actionable information

Why it is important

- Ensures the rule is not misconfigured
- Confirms the rule detects real attack behavior
- Prevents false positives or missed attacks
- Builds confidence in the SIEM detection capability

2. Accessing the Wazuh Dashboard
Steps

1. Open a web browser
2. Log in to the Wazuh Dashboard
3. **Navigate to:**
    - Security Events or
    - Threat Hunting

Definition
The Wazuh Dashboard is the centralized interface used to view, analyze, and investigate security alerts collected from monitored systems.

3. Locating the Custom Alert
Filtering the Alerts
Apply the following filters:

- Rule ID: 100100
- Rule description: *SSH Brute Force: 3 failed logins in 2 minutes*
- Alert level: High (Level 10)
- Event category: Authentication failures

What this does
Filtering isolates the alert generated by the custom brute-force detection rule, making validation easier and more accurate.

4. Verifying Alert Details
Key Fields to Validate

Field	What to Check	Why It Matters
Timestamp	Matches test time	Confirms real-time detection
Source IP	Attacker's IP	Identifies attack origin
Agent Name	Target system	Confirms affected host
Rule ID	100100	Confirms custom rule triggered
Severity Level	10	Confirms risk classification
Description	SSH brute force	Confirms correct detection

Validation Outcome
If all values match the test scenario, the alert is valid and accurate.

5. Correlation Confirmation
What Correlation Means
Correlation in Wazuh means:

- Multiple failed login events
- Combined into a single high-severity alert

Validation Check

- Ensure the alert appears after the 3rd failed login
- Confirm failures occurred within 2 minutes

Importance
This proves the rule is frequency-based, not triggered by a single failed attempt.
6. MITRE ATT&CK Mapping Validation
Observed Technique

- T1110 – Brute Force

Why this is useful

- Aligns detection with industry frameworks
- Improves incident response and reporting
- Helps SOC teams understand attacker behavior

7. Documenting the Alert Event

Alert Documentation Template

Field	Description
Date & Time	When alert was generated
Source IP	Attacking IP address
Target System	Host under attack
Rule ID	100100
Alert Description	SSH Brute Force detected
Attempts Count	3 failed logins
Time Window	2 minutes
Detection Status	Successful
Analyst Remarks	Rule functioned as expected

8. Evaluating Rule Effectiveness

Effectiveness Criteria

Accuracy

- Alert triggered only after threshold reached
- No alert for single failed login

Timeliness

- Alert generated immediately after 3rd attempt

Relevance

- Detects real brute-force behavior
- Focused on authentication abuse

Noise Reduction

- Prevents false positives
- Avoids alert fatigue

9. Strengths of the Rule

- Detects SSH brute-force attacks early
- Uses correlation instead of single events
- Provides high-confidence alerts
- Easily extendable with active response

10. Limitations and Improvements

Limitations

- Does not block attacker automatically
- May miss very slow brute-force attempts

Possible Enhancements

- Enable Active Response to block IP
- Add email or Slack notifications
- Increase correlation for distributed attacks

11. Final Validation Conclusion

Summary

The alert validation confirms that the Wazuh custom rule:

- Successfully detects multiple failed SSH login attempts
- Accurately correlates events within a defined time window
- Generates meaningful, high-severity alerts
- Effectively identifies brute-force attack behavior.

Learning Approach

1. Objective of the Learning Approach

Definition

A mini-SOC (Security Operations Center) lab is a small, controlled environment designed to practice threat detection, investigation, and response using real security tools.

Goal

- Learn how SIEM and endpoint tools work together
- Gain hands-on experience with alert detection and investigation
- Understand how alerts map to MITRE ATT&CK adversary behavior

2. Tools Used in the Mini-SOC Lab

2.1 Wazuh (SIEM)

Definition

Wazuh is an open-source Security Information and Event Management (SIEM) and XDR platform that:

- Collects logs from endpoints
- Correlates events using rules
- Generates security alerts
- Provides a centralized dashboard for analysis

Role in the Lab

- Detect security events (failed logins, malware, file changes)
- Correlate events over time
- Generate alerts mapped to attack techniques

2.2 Osquery (Endpoint Visibility)

Definition

Osquery is an endpoint monitoring tool that allows you to:

- Query operating system data using SQL-like queries
- Collect information about processes, users, files, registry, and network activity

Role in the Lab

- Provide deep endpoint visibility
- Validate Wazuh alerts with live system data
- Investigate suspicious activity (processes, PowerShell usage, persistence)

3. Mini-SOC Architecture (Conceptual)

Endpoint (Windows/Linux)
Wazuh Agent
Osquery
|
Logs + Endpoint Data
|
Wazuh Manager (SIEM)
|
Wazuh Dashboard
|
SOC Analyst Investigation

Explanation

- Endpoints generate logs and system activity
- Wazuh agent forwards logs to the manager
- Osquery provides on-demand endpoint data
- Wazuh Dashboard centralizes alerts and analysis

4. Detection Process in the Mini-SOC Lab

Step 1: Event Generation

An attacker-like action occurs, such as:

- Failed SSH logins
- PowerShell command execution
- Suspicious process creation

Step 2: Log Collection

- Wazuh agent collects system and security logs
- Osquery collects endpoint state data

Step 3: Alert Generation

- Wazuh correlates events using detection rules
- Alerts are generated when suspicious behavior is detected

5. MITRE ATT&CK Mapping Process
Definition
MITRE ATT&CK is a globally recognized framework that categorizes attacker behavior into:

- Tactics (Why the attacker acts)
- Techniques (How the attacker acts)

6. Example: Mapping Alerts to MITRE ATT&CK
6.1 PowerShell Attack Detection
Scenario
An attacker executes malicious PowerShell commands on a Windows endpoint.
Detection in Wazuh

- Wazuh detects PowerShell execution from event logs
- Suspicious command lines are flagged

MITRE Mapping

Component	Mapping
Tactic	Execution
Technique	T1059 – Command and Scripting Interpreter
Sub-Technique	T1059.001 – PowerShell

Why This Mapping Matters

- Identifies attacker intent (code execution)
- Helps SOC analysts prioritize alerts
- Enables structured threat hunting

7. Osquery's Role in Alert Investigation
Validation Using Osquery
After Wazuh raises an alert:

- Osquery queries confirm:
  - Running PowerShell processes
  - Command-line arguments
  - Parent process relationships
  - Persistence mechanisms

Example Investigation Questions

- Is PowerShell running unexpectedly?
- What command was executed?
- Which user launched the process?
- Is this behavior normal?

8. SOC Analyst Workflow in the Lab
Step-by-Step Workflow

1. Alert appears in Wazuh dashboard
2. Analyst reviews rule description and severity
3. Alert is mapped to MITRE ATT&CK
4. Osquery is used to validate endpoint behavior
5. Analyst decides whether it is:
    - True positive
    - False positive
6. Incident is documented

9. Learning Outcomes from This Approach

Skills Gained

- SIEM alert analysis
- Endpoint investigation
- MITRE ATT&CK mapping
- Threat detection logic
- SOC analyst thinking

Knowledge Developed

- How attackers operate
- How detections align with tactics
- How tools complement each other

10. Documentation Example (Lab-Ready)

Field	Value
Tool Used	Wazuh + Osquery
Alert Type	PowerShell Execution
MITRE Tactic	Execution
MITRE Technique	T1059
Detection Source	Windows Event Logs
Validation Method	Osquery process query
Outcome	Confirmed suspicious activity

11. Conclusion

This learning approach combines SIEM detection (Wazuh) with endpoint visibility (Osquery) and MITRE ATT&CK mapping to simulate real-world SOC operations. By building a mini-SOC lab, learners develop practical skills in alert detection, investigation, and adversary behavior analysis, preparing them for real SOC analyst roles.