Page 1: Introduction

SIEM Internship Program – Phase 1 Learning

Welcome to Phase 1 of the SIEM (Security Information and Event Management) Internship Program. This foundational stage is designed to immerse cybersecurity learners in the practical aspects of building, configuring, and utilizing a SIEM platform for real-world threat detection and analysis.

As cyber threats grow in scale and complexity, the role of a well-configured SIEM becomes critical in enabling early detection, response, and investigation. Whether in a corporate SOC (Security Operations Center), a government environment, or a managed security service provider (MSSP), SIEM systems act as the **central nervous system** of cybersecurity operations — aggregating logs, correlating activities, and generating actionable alerts.

Program Focus:

This phase emphasizes **hands-on experience** with the following components:

- **Building a Detection Lab:** Set up virtual machines and essential security tools to simulate an enterprise-like environment.
- **Log Source Configuration:** Enable and forward key log types such as Windows Event Logs, Sysmon, and Linux audit logs.
- **Simulating Realistic Attacks:** Recreate tactics and techniques from the MITRE ATT&CK framework such as brute-force attacks, lateral movement, and log tampering.
- **Detection Engineering:** Write SPL (Search Processing Language) rules in Splunk to identify and alert on suspicious activities.
- **Structured Documentation:** Record detection logic, observations, and screenshots in Markdown format suitable for GitHub and professional reporting.

Expected Outcomes:

By the end of this phase, I will be able to:

- Understand how log sources contribute to threat detection.
- Simulate basic attacker behaviors and recognize their footprints in event logs.
- Build and validate detection rules in a SIEM environment.

- Develop a personal GitHub-based portfolio documenting my learning journey and technical competencies.

This is the first step in preparing for real-world SOC analyst roles, Blue Team operations, and cybersecurity monitoring.

Page 2: What is a SIEM and Why It Matters?

Understanding SIEM (Security Information and Event Management)

A **SIEM** — Security Information and Event Management — is a security solution that enables organizations to collect, normalize, analyze, and respond to security-relevant data from across their infrastructure. It acts as a centralized platform where logs and telemetry data from various sources such as firewalls, servers, endpoints, and network devices are aggregated, enriched, and analyzed in near real-time.

SIEMs play a **critical role** in modern cybersecurity operations by helping detect threats early, investigate incidents, and meet compliance requirements. Prominent SIEM tools include **Splunk**, **Elastic Stack (ELK)**, **Wazuh**, **IBM QRadar**, **ArcSight**, and **Microsoft Sentinel**, among others.

Core Functions of a SIEM:

1. **Log Collection and Centralization:**
    - Collects logs from multiple heterogeneous systems (e.g., Windows, Linux, firewalls, cloud apps).
    - Uses agents or forwarders (like Sysmon, Winlogbeat, Filebeat) to send logs to the SIEM platform.
2. **Parsing and Normalization:**
    - Standardizes logs into a consistent format to enable easier searching and analysis.
3. **Correlation and Detection:**
    - Uses rules or machine learning to detect suspicious behavior based on log patterns (e.g., multiple failed logins followed by success).
4. **Alerting and Notification:**
    - Sends real-time alerts to security teams when specific detection rules or thresholds are met.

5. **Threat Hunting and Forensics:**
    - Enables analysts to perform historical searches, timelines, and case building using collected logs.
6. **Dashboards and Visualizations:**
    - Offers visual representations (graphs, tables, KPIs) to understand the security posture at a glance.
7. **Compliance Reporting:**
    - Helps organizations meet regulatory standards like GDPR, HIPAA, PCI-DSS by tracking and reporting on security events.

Why SIEM Matters in the Real World

Modern enterprises deal with **thousands to millions of security events per day**. Without a SIEM, analyzing this volume of data manually would be nearly impossible. SIEMs allow:

- **Rapid Detection of Attacks:** E.g., brute-force, lateral movement, malware execution.
- **Faster Incident Response:** Real-time alerting reduces time to detect (TTD) and time to respond (TTR).
- **Root Cause Analysis:** With historical logs and event correlation, analysts can trace an attacker's path through the network.
- **Compliance and Auditing:** SIEMs provide documentation and proof of control implementation.

Real-World Use Cases of SIEMs:

- Detecting **unauthorized access attempts** (e.g., brute force, password spraying).
- Identifying **suspicious logins** during non-business hours.
- Correlating **lateral movement** using RDP, SMB, or PSExec.
- Flagging **log deletion attempts** (e.g., wevtutil cl or Clear-EventLog commands).
- Tracking **new user creation** or privilege escalation.
- Alerting when **firewall or antivirus logs** show signs of malware activity.

Conclusion:

In essence, a SIEM turns raw machine data into actionable intelligence. It serves as the **eyes and ears** of my security infrastructure. Mastering SIEM tools and understanding how to detect attacks using logs is one of the most **valuable skills in cybersecurity today**.

Page 3: Lab Setup

Overview of the SOC Lab Environment

Setting up a functional Security Operations Center (SOC) lab is the foundational step in learning how real-world security teams detect and respond to cyber threats. The goal of Phase 1 was to create a miniature, but realistic, environment that mimics an enterprise network — with logging, monitoring, and attack simulation components integrated to allow practical hands-on experience.

This section outlines the tools, systems, and network design used to build the lab.

Virtualization Platform

To simulate a networked environment locally, virtualization was used:

- **VirtualBox** or **VMware Workstation** for creating and running multiple virtual machines.
- NAT or Host-Only Adapter configuration used to isolate the lab and allow internal network communication.

Core Lab Components

Role	System	Purpose
SIEM Server	Windows or Linux VM running **Splunk Enterprise** (Free Edition)	Centralized log collection and analysis
Windows Victim Machine	Windows 10 VM	Target of simulated attacks, logs forwarded to SIEM

Role	**System**	**Purpose**
Attacker Machine	Kali Linux VM	Used for brute force, lateral movement, and privilege escalation simulations
Log Forwarders	Sysmon, Winlogbeat, NXLog	Used to collect system logs and send them to Splunk
Linux Target (Optional)	Ubuntu with Auditd	For collecting and forwarding Linux security events

---

SIEM Platform: Splunk Free Edition

- **Version Used:** Splunk Enterprise (free license, 500MB/day limit)
- **Configuration:** Installed on a dedicated VM, listening on TCP/UDP ports (e.g., 9997)
- **Features Used:** Search, alerts, dashboards, saved reports

---

Log Forwarding and Sources

- **Windows Logs:**
  - Enabled **Security Auditing** via secpol.msc and Group Policy.
  - Installed **Sysmon** with a custom configuration for deep process/thread monitoring.
  - Used **Winlogbeat** or **NXLog** to forward Event Logs to Splunk over TCP.
- **Linux Logs (Optional):**
  - **Auditd** configured to capture file access, logins, and sudo activity.
  - Logs forwarded via Filebeat or direct syslog integration.

Network Diagram

Below is a high-level description of the lab network:

```
+-------------------+     TCP 9997     +--------------------+
|  Windows Victim  | ------------------> |  SIEM Server     |
|  (Sysmon + Logs) |                  |  (Splunk)       |
+-------------------+                 +--------------------+
       ↑
       |
 Simulated Attacks
       |
+-------------------+
|  Kali Linux VM   |
| (Hydra, CrackMap) |
+-------------------+
```

This topology ensures logs generated during attacks are captured and sent to the SIEM, enabling full end-to-end detection testing.

Optional Lab Enhancements

- **TryHackMe Labs:** Used for guided practice on attacks like RDP brute-force, lateral movement.
- **Security Onion or Wazuh:** For deeper network visibility (e.g., packet capture, NIDS).
- **pfSense Firewall:** To simulate perimeter controls and log firewall activity.

Conclusion

The SOC lab was designed to balance realism and simplicity. The tools chosen reflect those used in enterprise SOCs, and the virtualized environment allows for complete attack and detection cycles. The foundation built here is essential for the detection use cases in the upcoming pages.

Page 4: Log Sources Configuration

A Security Information and Event Management (SIEM) system is only as powerful as the data it receives. Configuring diverse log sources is essential for visibility across endpoints, network devices, and servers. In Phase 1, the focus was on enabling and forwarding logs from both Windows and Linux systems to build a baseline for threat detection.

🔐 Windows Log Configuration

Windows machines generate rich security logs that can be used to detect brute-force attempts, lateral movement, privilege escalation, and more. The following steps were taken to enhance visibility:

1. Enable Security Auditing

- Accessed via secpol.msc → Local Policies → Audit Policy.
- Enabled the following key audit policies:
  - Audit Logon Events
  - Audit Account Logon Events
  - Audit Account Management
  - Audit System Events
- Resulted in Event IDs like 4624, 4625, 4720, 4728, and 1102.

2. Install Sysmon (System Monitor)

- Sysmon is a Windows system service that logs detailed event data.
- Installed using:
- sysmon.exe -accepteula -i sysmonconfig.xml
- Sample logs include:
  - **Process creation (Event ID 1)**
  - **Network connections (Event ID 3)**
  - **Registry modifications (Event ID 13)**

3. Configure Log Forwarding

- **NXLog** or **Winlogbeat** used to forward logs from Windows to Splunk:
  - Configured to monitor:
    - Security

        - Sysmon
        - System
        - Application logs
    - Sent via TCP to Splunk Universal Forwarder or directly to Splunk Enterprise.

🐧 Linux Log Configuration (Optional but Recommended)

Linux systems were optionally included to simulate real-world heterogeneous environments.

1. Auditd Configuration

- Auditd (Linux Auditing Daemon) was used to track system activity.
- Monitored for:
    - User login/logout
    - sudo command executions
    - File permission changes
- Configuration file: /etc/audit/audit.rules

2. rsyslog or Filebeat Forwarding

- Logs forwarded via:
    - **rsyslog** (UDP or TCP)
    - **Filebeat** with a Linux module for structured log parsing.
- Destination: Splunk, listening on appropriate port (e.g., 514 or 9997)

🧪 Testing Log Flow

To verify that the log sources were correctly configured:

- Performed a simulated login and observed **Event ID 4624** in Splunk.
- Cleared the event logs with wevtutil cl Security and confirmed **Event ID 1102** was captured.
- Added a new user via net user and ensured **Event ID 4720** was indexed.

💡 Tips for Real-World Readiness

- Always **time-sync** my machines (use NTP) to avoid correlation issues.
- Use **hostnames and tags** in my log forwarders for better identification.
- Tune log sources to balance between verbosity and performance.

Conclusion

With proper log source configuration, a SIEM becomes a powerful lens into system activity. Whether detecting a brute-force attack or privilege escalation, the reliability and richness of logs are what enable accurate and timely detection.

Page 5: Detection Use Case 1 - Brute Force Login

Overview

The brute force login attack is a common technique used by attackers to gain unauthorized access to systems by systematically trying many passwords or passphrases until the correct one is found. This use case focuses on detecting such an attack using Windows Security Event Logs collected through Splunk.

Attack Simulation

In our lab setup, we manually simulated a brute force attack from a Kali Linux VM targeting a Windows virtual machine. Using tools such as **Hydra**, we repeatedly attempted login with incorrect passwords. After several failed attempts, a successful login was achieved using legitimate administrator credentials.

Tools Used:

- **Attacker:** Kali Linux VM
- **Target:** Windows 10 VM
- **Attack tool:** Hydra (or similar password cracking utility)
- **SIEM:** Splunk Enterprise
- **Log Source:** Windows Security Event Logs (Event IDs 4625 and 4624)
- **Log Forwarder:** Splunk Universal Forwarder installed on Windows VM

Event IDs Monitored

- **4625**: Failed login attempt

- **4624**: Successful login

These events are critical in detecting the brute force pattern.

Detection Logic

The detection rule monitors multiple failed login attempts (threshold: 10 or more) within a short time window (5 minutes). It then correlates these failed attempts with a successful login from the same source account or IP address occurring shortly after.

This logic helps identify brute force attempts followed by successful privilege escalation.

SPL Query

```
index=* sourcetype=WinEventLog:Security (EventCode=4625 OR EventCode=4624)

| eval Account = Account_Name

| stats count(eval(EventCode=4625)) as failed_attempts,

      count(eval(EventCode=4624)) as successful_logins,

      earliest(_time) as first_attempt, latest(_time) as last_attempt

  by Account, Computer

| where failed_attempts >= 10 AND successful_logins >= 1 AND (last_attempt - first_attempt) <= 300
```

- This query filters events 4625 and 4624.
- It groups events by account and host.
- Alerts when 10+ failed attempts happen, followed by a successful login within 5 minutes (300 seconds).

Screenshots to Capture

- Splunk search query and results showing failed and successful login events.
- Windows Event Viewer displaying the corresponding Event IDs during the brute force simulation.
- Kali Linux terminal with Hydra commands and output.
- Splunk Enterprise dashboard showing alerts or detection results.

Logs to Save

- Raw Security event logs from the Windows VM that correspond to Event IDs 4625 and 4624 during the attack timeframe.

- Splunk saved search logs and alert logs triggered by the detection query.

Analyst Recommendations

- Investigate accounts with multiple failed logins followed by a success.
- Check source IP addresses for suspicious behavior.
- Consider implementing account lockout policies to prevent brute force attempts.
- Monitor for similar patterns in future logs to proactively identify attacks.

Page 6: Detection Use Case 2 – After-Hours Login

Overview

Organizations typically define standard business operating hours—commonly between 9:00 AM and 7:00 PM. Logins occurring outside of these hours, especially from privileged or administrator accounts, may indicate unauthorized access or insider threats. This use case focuses on detecting off-hours login attempts by monitoring Windows Event Logs using Splunk.

Attack Simulation

To simulate suspicious login behavior, we manually logged into the Windows machine using the redadmin account at an off-hour—specifically **08:05 AM on 25th May 2025**. This falls slightly outside our defined business hours and is intended to test the detection mechanism for early-morning or late-night access attempts.

Lab Details

- **User Account Used:** redadmin
- **Login Time:** 05/25/2025 08:05:34 AM
- **Host Name:** hacker
- **Log Source:** Windows Security Event Log (Event ID 4624)
- **Log Collection Tool:** Splunk Universal Forwarder
- **SIEM:** Splunk Enterprise

Event ID Monitored

- **4624**: Successful Logon

This event captures detailed information about user logins, including account name, source machine, and logon time.

Detection Logic

The detection focuses on identifying logins by privileged users that occur outside standard working hours. This is particularly important in environments where administrators are not expected to log in during non-working times unless for scheduled maintenance.

SPL Query

```
index=* sourcetype=WinEventLog:Security EventCode=4624

| eval hour=strftime(_time,"%H")

| where tonumber(hour) < 9 OR tonumber(hour) >= 19

| stats count by Account_Name, host, _time
```

Explanation:

- The query filters only Event ID 4624 (successful logons).
- It extracts the hour from the timestamp and flags any login that occurs before 9 AM or after 7 PM.
- It then groups the data by account, host, and timestamp for analysis.

Screenshots to Capture

- Splunk query window with the above SPL.
- Resulting event data showing the redadmin login at 08:05 AM.
- Event Viewer on Windows VM showing Event ID 4624 with the same timestamp.

Logs to Save

- Exported logs of Event ID 4624 around the simulated login.
- Screenshot or raw output of Splunk search showing detection of the off-hours login.

Analyst Recommendations

- Review all off-hour logins from privileged or sensitive accounts.
- Investigate the purpose of these logins—check change logs, scheduled jobs, or user justification.
- Enforce stricter access controls and multi-factor authentication (MFA) for privileged accounts.

- Set up recurring alerts for after-hours activity from administrator or domain-level users.

Page 7: Detection Use Case 3 – RDP Lateral Movement

Overview

Lateral movement is a key phase in post-compromise attack strategies, where an attacker moves between systems within a network to escalate privileges or access sensitive information. One common technique is using **Remote Desktop Protocol (RDP)**. This use case focuses on detecting such movements via **Logon Type 10**, which signifies an RDP session.

Attack Simulation

We manually simulated lateral movement from one internal system to another using RDP:

- **Attacker Machine:** Kali Linux VM
- **Victim Machine:** Windows VM
- **Action Performed:** RDP login attempt followed by successful RDP access
- **Tool Used:** Kali's RDP client (xfreerdp) or Windows' built-in RDP client
- **Target Logon Type:** 10 (RemoteInteractive – RDP)

Event IDs Monitored

- **4625**: Failed login attempt
- **4624**: Successful login
    - Look for LogonType=10 indicating RDP session

Objective

To detect RDP login (Event ID 4624 with LogonType=10) from unusual internal IP addresses, especially after prior failed RDP attempts. Such a pattern is indicative of a brute-force or credential stuffing attack followed by successful lateral movement.

Detection Logic

We correlate:

1. Failed login attempts from an internal IP (Event ID 4625, LogonType=10)
2. Followed by a successful RDP login (Event ID 4624, LogonType=10) from the **same IP within 10 minutes**

This correlation provides visibility into successful lateral movement after repeated attempts.

SPL Query

```
index=* sourcetype=WinEventLog:Security (EventCode=4625 OR EventCode=4624)

| eval LogonType=coalesce(Logon_Type, LogonType)

| eval SourceIP=coalesce(Source_Network_Address, Workstation_Name)

| where LogonType="10"

| eval Account=Account_Name

| stats count(eval(EventCode=4625)) as failed_attempts,

      count(eval(EventCode=4624)) as successful_logins,

      earliest(_time) as first_attempt, latest(_time) as last_attempt

  by Account, SourceIP, host

| where failed_attempts >= 3 AND successful_logins >= 1 AND (last_attempt - first_attempt) <= 600
```

Explanation:

- Filters for RDP logon attempts (LogonType 10).
- Groups by source IP and account.
- Flags multiple failed attempts followed by a successful RDP login within 10 minutes.

Screenshots to Capture

- Splunk search showing the SPL output with RDP detection.
- Event Viewer logs showing Event ID 4625 and 4624 on the target Windows machine.
- Proof of IP address and logon type correlation.

Logs to Save

- Logs from the Windows Event Viewer for both Event IDs (4625, 4624).
- Splunk output with fields like Account, SourceIP, host, first_attempt, last_attempt.

Analyst Recommendations

- Flag and investigate all RDP logins with a failed attempt pattern.
- Consider RDP login from unknown IPs a high priority unless previously whitelisted.
- Disable RDP where not necessary.
- Implement network segmentation and IP allowlisting.
- Enable multi-factor authentication for RDP access.

Page 8: Detection Use Case 4 – Log Tampering

Overview

Log tampering is a dangerous post-exploitation tactic used by attackers to erase their traces or disrupt incident response. Adversaries often attempt to clear Windows Security logs, disable auditing, or tamper with logging services altogether. Detecting such behavior is crucial for maintaining the integrity of a forensic investigation and ensuring that attack footprints remain traceable.

Attack Simulation Performed

In our lab setup, we simulated a log-clearing attempt from the command line using the following method:

Command Used:

```
wevtutil cl Security
```

This command clears the Windows Security Event Log – an extremely suspicious action when performed by any user, especially outside a maintenance window or without explicit approval.

- **Tool used:** Built-in Windows wevtutil.exe
- **Executed from:** Windows command prompt (cmd)
- **User:** Administrator account
- **Observed via:** Sysmon Event ID 1 (Process Creation)

Log Source & Monitoring Configuration

- **Sysmon** was configured and installed on the Windows machine.
- Logs collected by Splunk via Universal Forwarder.
- Event source: XmlWinEventLog:Microsoft-Windows-Sysmon/Operational

- Key EventCode: **1** (Process Creation)

Objective

Detect whenever the wevtutil cl Security command is executed, especially by admin-level accounts, as this indicates an attempt to clear logs and evade detection.

MITRE ATT&CK Mapping

- **Tactic:** Defense Evasion
- **Technique:** T1562.002 – Impair Defenses: Disable Windows Event Logging

Detection Logic

We use Sysmon Event ID 1 to catch suspicious command-line activities. The command string is matched using pattern detection against known tampering tools or actions.

SPL Query

index=* sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational EventCode=1

| eval cmd=lower(CommandLine)

| search cmd="wevtutil cl security" OR cmd="auditpol /clear" OR cmd="clear-eventlog"

| table _time, Computer, User, CommandLine, ParentImage

Explanation:

- Filters for command-line executions (EventCode=1) with known tampering patterns.
- Searches for specific tampering commands (e.g., wevtutil cl Security).
- Outputs command metadata, user, and machine for triage.

Screenshots to Capture

- Windows CMD or PowerShell where wevtutil cl security is executed.
- Event Viewer (Sysmon) showing Event ID 1 for the tampering command.
- SPL query in Splunk returning detection result.

Logs to Save

- Sysmon logs showing the exact command and timestamp.
- Screenshot of the detection in Splunk.
- Any associated Event Logs that indicate suspicious privilege use.

Analyst Recommendations

- Log clearing is almost never a routine action; it must be flagged immediately.
- Alert on all invocations of wevtutil cl, especially when done by unexpected accounts or at off-hours.
- Retain logs in a remote, centralized system to avoid full deletion from endpoints.
- Monitor command-line activity as part of my baseline detection strategy.

Page 9: Documentation & GitHub Reporting

Importance of Documentation

In cybersecurity, proper documentation is just as critical as detection and response. It ensures repeatability, transparency, and accountability. In a professional SOC (Security Operations Center), analysts document everything—from detection logic to incident timelines—to ensure clear communication and effective threat hunting.

For this internship, documenting each detection scenario ensures I not only understand what I am doing, but can also explain and replicate it later. Employers highly value candidates who can demonstrate not only technical ability but also clear communication and documentation skills.

Recommended Folder Structure for GitHub Repo

To make my Phase 1 project easy to understand and professional, my GitHub repository should follow a clean and consistent structure.

✅ **Suggested Structure:**

```
SIEM-Internship-Phase1/
│
├── scenario-1-detection-brute/
│   ├── detection.md
│   ├── screenshots/
│   ├── logs/
│
├── scenario-2-off-hours-login/
│   ├── detection.md
│   ├── screenshots/
│   ├── logs/
│
├── scenario-3-lateral-movement-rdp/
│   ├── detection.md
│   ├── screenshots/
│   ├── logs/
│
├── scenario-4-log-tampering/
│   ├── detection.md
│   ├── screenshots/
│   ├── logs/
│
├── scenario-5-user-created/
│   ├── detection.md
│   ├── screenshots/
│   ├── logs/
│
├── documentation-phase-1.pdf
├── README.md
```

What Each File Should Contain

🔍 detection.md per Scenario

Each detection should be documented with:

- Scenario description
- Objective of detection
- Tools/logs used
- Detection logic and SPL query

- Sample log/event
- Detection status
- Analyst notes and false positives

🖼 screenshots/ folder

Include images of:

- My SIEM dashboard
- Terminal/command prompt with attack simulation
- Splunk query outputs
- Event Viewer logs

📁 logs/ folder

Include exported:

- JSON, CSV, or text-based logs from Windows, Sysmon, or Splunk
- Sample event data used for validation

Tools for Enhancing Documentation

📐 Diagrams & Visuals

Use tools like:

- **Draw.io (diagrams.net)** – for network diagrams, attack chains
- **Mermaid.js** – markdown-supported flowcharts for GitHub README
- **Lucidchart** (optional)

📄 Markdown Style Tips

- Use ## for subheadings.
- Use ```spl for code blocks and SPL queries.
- Include emojis (✅, ⚠️, 🔍) to highlight detection status or action steps.
- Add internal links if my GitHub repo is large.

Submission Guidelines

- Push everything to GitHub before submission deadline.
- Confirm detection.md for each scenario is complete and formatted.

- Include my completed documentation-phase-1.pdf in the root directory.
- README.md should introduce my project, tools used, and detection highlights.

Page 10: Reflection Questions & Evaluation

Final Thoughts

As I complete Phase 1 of my SIEM Internship Program, it's important to reflect on what I've learned, the challenges I faced, and how this foundational experience will support my future roles in cybersecurity. The following questions are designed to help me evaluate my progress and prepare me to discuss my work confidently in interviews or technical assessments.

These responses must be documented in my GitHub repository, either in a separate reflection.md file or directly appended to my README.md under a "Reflection & Learnings" section.

🧠 Reflection & Evaluation Questions

1. What is the role of a SIEM in modern cybersecurity?

A SIEM aggregates logs from multiple sources, performs real-time analysis, generates alerts, and supports incident investigations. It acts as the central nervous system of a Security Operations Center, helping detect, respond to, and prevent attacks through correlation and visualization.

---

2. What challenges did I face while setting up my lab?

- Misconfigurations in log forwarding agents (e.g., Winlogbeat, Sysmon)
- Network connectivity between virtual machines
- Lack of logs due to auditing policies not being properly enabled
- Time synchronization issues affecting detection accuracy

---

3. What are the differences between Sysmon logs and Windows Security logs?

- **Windows Security Logs**: Native logs that include login events, account changes, policy updates (Event IDs: 4624, 4625, 4720, etc.)

- **Sysmon Logs**: Provide deep visibility into system activity such as process creation, network connections, and registry modifications (Event ID: 1, 3, 11, etc.)

Sysmon enhances detection capabilities beyond what Windows Security Logs offer by default.

---

4. How does a brute force attack appear in logs? Mention specific Event IDs.

A brute force attack is visible through a high frequency of Event ID **4625** (failed logon attempts), usually followed by an Event ID **4624** (successful login) if the attack succeeds. The attacks are often performed over short time frames from the same source IP or user account.

---

5. How would I detect a login outside normal business hours?

Create a detection rule that filters Event ID 4624 (successful login) by logon type (e.g., Type 2 for local, Type 10 for RDP) and uses strptime and date_hour functions to isolate events occurring outside defined working hours (e.g., 8 AM – 6 PM). Off-hours logins by privileged accounts should trigger alerts.

---

6. Describe how RDP lateral movement is tracked in event logs.

RDP lateral movement appears as **Event ID 4624** with **LogonType=10** (Remote Interactive). When detected from an internal, non-standard IP, and especially following failed attempts, it suggests unauthorized internal movement. These logs can be correlated with **4625** (failed RDP attempts) for context.

---

7. What is the risk of log tampering, and how can we detect it?

Log tampering hides attacker actions, making investigation and response difficult. Commands like wevtutil cl Security clear logs. Detection can be done using Sysmon Event ID **1** (Process Creation), filtering for suspicious commands. Centralized log storage mitigates local tampering.

---

8. What improvements would I make in my lab setup if given more time?

- Automate log parsing and enrichment
- Implement alerting and email notifications

- Integrate more advanced data sources (e.g., firewall or antivirus logs)
- Use real attack emulation frameworks like Atomic Red Team or Caldera

---

9. How will this phase help me in real-world interviews or jobs?

This phase provides hands-on experience in setting up a SOC lab, simulating realistic attacks, and building detection logic—skills directly applicable in SOC analyst, threat hunter, and DFIR roles. It shows initiative, technical depth, and documentation skills to future employers.

---

10. What was my biggest takeaway from Phase 1?

The value of structured logging, correlation, and deep inspection of event logs. Even simple attacks leave footprints. With the right tools and queries, defenders can surface critical events and take timely action. This phase has strengthened both my technical skills and analytical thinking.
