Ministry of Communications and Information Technology (MCIT)

Digital Egypt Pioneers Initiative – DEPI

Round 3 (June – December 2025)

Information Security Analyst Track

(MNF3_ISS6_S1)

Graduation Project:

Blue Ants:

Mini Security Operations Center (SOC) Environment

Supervised by:

Eng: Muhammad Gamal

Prepared by:

1. *Ahmed Ehab Ibrahim Shalaby (Team leader)*
2. *Ebrahim Ezzat Ahmed Briqaa*
3. *Yousef Nabil Yousef Hagir*
4. *Shahd Mostafa Ebrahim El Saidy*

Acknowledgment

I would like to express my deepest gratitude to our instructor, **Eng: Muhammad Gamal** for their continuous guidance, support, and constructive feedback throughout the development of this project. Their expertise and dedication played a significant role in helping us design, build, and refine our Mini Security Operations Center.

As the team leader of this project, I am proud to acknowledge the effort, commitment, and collaboration shown by my teammates:
Ebrahim Ezzat, Yousef Hagir, Shahd El Saidy.
Each member contributed with professionalism and enthusiasm, making the teamwork experience both productive and inspiring.

I thank my team for trusting my leadership and for working together to overcome challenges, perform testing, analyzing security events, and delivering a complete, high-quality SOC environment.

Finally, I extend my appreciation to the DEPI program and the creators of the open-source technologies we used—pfSense, Wazuh, Snort, DVWA, and Kali Linux—for providing the tools and knowledge that made this hands-on learning experience possible.

Abstract

This project presents the design and implementation of a fully functional Mini Security Operations Center (SOC) built within a segmented virtual environment. The system consists of three isolated network zones—SIEM, Victims, and Attacker—interconnected through a pfSense firewall configured with Snort IDS for intrusion detection and centralized log forwarding. The Wazuh SIEM platform, deployed on Ubuntu, serves as the core monitoring and correlation engine, receiving logs from pfSense, Windows, Linux, and DVWA servers through integrated Wazuh agents.

A series of controlled attack scenarios were executed from a dedicated Kali Linux machine, including RDP brute-force attempts, SQL injection and XSS. The project demonstrates the complete SOC lifecycle: log ingestion, detection, correlation, triage, and reporting. All events were mapped to the MITRE ATT&CK framework to ensure structured threat analysis. The resulting SOC environment provides hands-on experience in enterprise-grade monitoring, use case development, and incident investigation, enabling a practical understanding of real-world security operations.

Introduction

Cybersecurity has become one of the most critical pillars in protecting modern organizations from evolving cyber threats. Security Operations Centers (SOCs) play a central role in monitoring environments, detecting attacks, and responding effectively. This project focuses on the design and implementation of a **Mini Security Operations Center (SOC)** that mirrors real-world enterprise SOC operations within a controlled virtualized lab environment.

The environment consists of a **segmented network architecture** divided into three main zones:

- **SIEM Network:** hosting the **Wazuh Manager** for centralized log collection, correlation, and alerting.
- **Victims Network:** containing a **Windows 10 endpoint** and a **DVWA** web application server.
- **Attacker Network:** a Kali Linux machine used to simulate offensive security scenarios and generate detectable events.

A pfSense firewall sits at the core of the environment, providing routing, DHCP, firewall enforcement, NAT, and network intrusion detection capabilities via **Snort IDS**. All machines forward their logs to the Wazuh SIEM, enabling centralized visibility across the entire infrastructure.

To validate the SOC's effectiveness, the team executed a set of controlled attack scenarios including **RDP brute force attempts**, **SQL injection**, **Cross-Site Scripting (XSS)**,. Each scenario generated logs that were analyzed, correlated, and mapped to the **MITRE ATT&CK framework**, demonstrating the full lifecycle of threat detection and incident analysis.

This project provides practical, hands-on experience in SOC engineering and operations—covering log ingestion, threat detection, use case development, alert triage, and incident reporting. It serves as a foundation for understanding real-world SOC workflows and equips the team with essential defensive security skills needed in modern cybersecurity roles.

Table of Contents

CHAPTER 1 — SOC SETUP & LOG INGESTION

(Full Expanded Version — Enterprise Grade, with Screenshot Placement Markers)

Summary Overview

This represents **Week 1** of the **Digital Egypt Pioneers Initiative (DEPI)** Mini SOC build. The objective is to design and deploy a **small-scale Security Operations Center (SOC)** that collects, analyzes, and correlates security logs from multiple systems within a segmented virtual network.

The SOC is built on **VMware Workstation Pro** and consists of **five virtual machines**, connected through a **pfSense firewall** that provides routing, firewall, and IDS/IPS (Snort) capabilities.
Each network segment — **SIEM**, **Victims**, and **Attacker** — is logically separated to simulate real-world enterprise environments.

All systems receive IP addresses dynamically via **pfSense DHCP**, and logs are centrally collected and analyzed by **Wazuh Manager**, which serves as the **SIEM** (Security Information and Event Management) platform.

Machine Roles:

Machine	Role	Description
pfSense Firewall	Network Gateway, Firewall & IDS	Routes traffic between networks, filters connections, and runs Snort for intrusion detection. Forwards logs to Wazuh.
Wazuh Manager (Ubuntu)	SIEM Server	Receives logs from pfSense, Windows, and Linux agents. Correlates events and generates alerts.

Machine	Role	Description
Windows 10	Victim Endpoint	Represents a user workstation. Runs the Wazuh agent to forward system and security logs.
DVWA Server	Victim Web Application Server	Hosts the “Damn Vulnerable Web Application” to simulate real-world web attack scenarios.
Kali Linux	Attacker System	Used to perform ethical hacking tests (e.g., scans, brute-force, SQLi, XSS) to trigger alerts and validate SOC functionality.

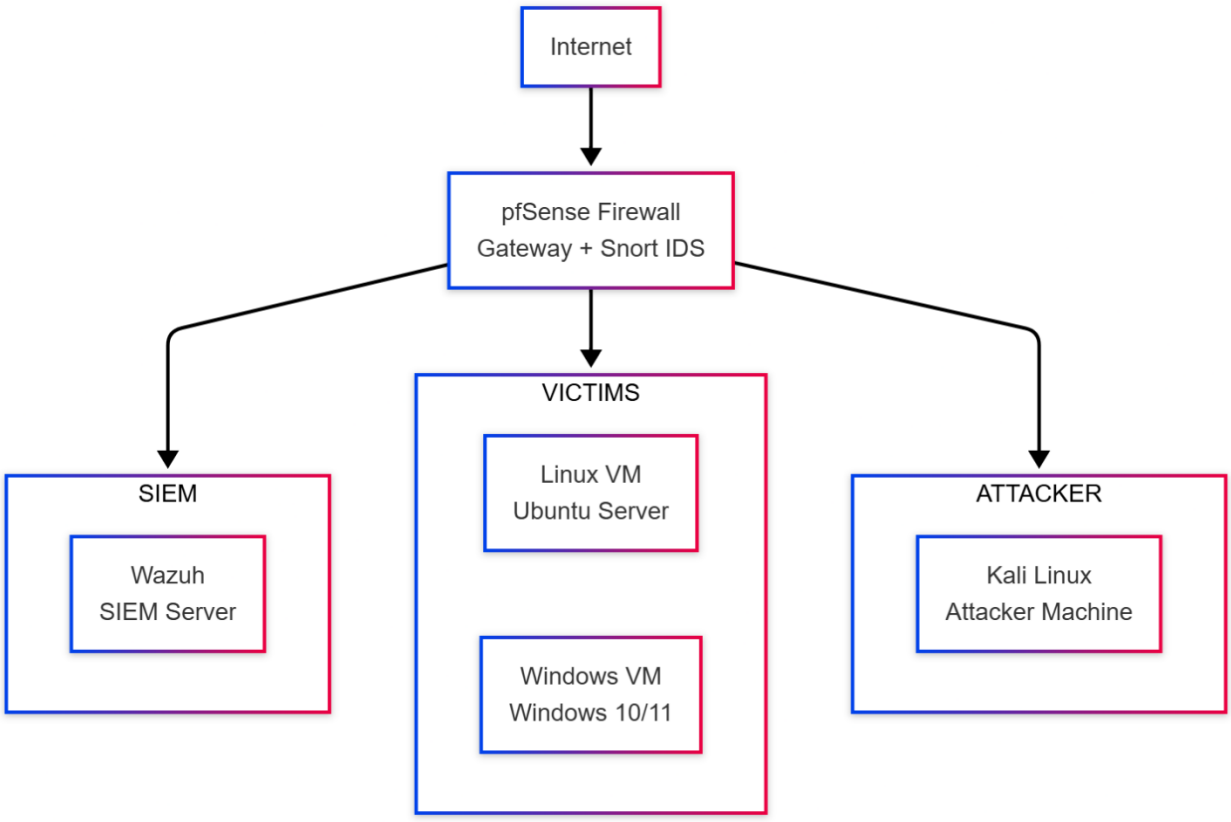
VMs Installation Resources

Virtual Machine	Base OS	Download Link
pfSense	pfSense	https://www.pfsense.org/download/
Ubuntu	Linux (for Wazuh)	https://ubuntu.com/download/desktop
Windows 10 Pro	Windows Endpoint	https://www.microsoft.com/en-us/software-download/windows10
DVWA	Vulnerable Web Application	https://github.com/digininja/DVWA
Kali Linux	Penetration Testing OS	https://www.kali.org/get-kali/#kali-virtual-machines

Virtual Machine	Base OS	Download Link
Wazuh Installation Script	SIEM Platform	https://documentation.wazuh.com/current/quickstart.html

---

Lab Architecture Overview



Network Segmentation Overview:

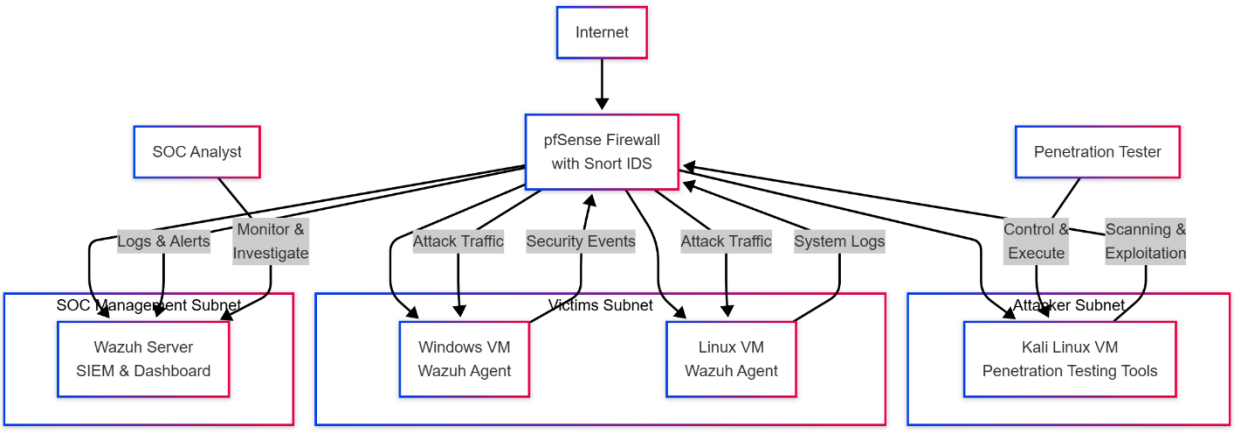
Network	Purpose	Subnet	pfSense IP	Connected VMs
WAN	Internet / updates	DHCP	Dynamic	—

Network	Purpose	Subnet	pfSense IP	Connected VMs
SIEM (LAN1)	Security monitoring	40.40.40.0/24	40.40.40.1	Wazuh Manager
Victims (LAN2)	Production targets	20.20.20.0/24	20.20.20.1	Windows 10, DVWA
Attacker (LAN3)	Red-team zone	30.30.30.0/24	30.30.30.1	Kali

Segmentation policy:

- Victims → SIEM (allowed)
- Victims → Attacker (allowed)
- SIEM → Attacker (allowed)
- SEIM → Victims (allowed)
- Attacker → Victims (allowed)
- Attacker → SIEM (blocked)

Traffic Flow Overview:

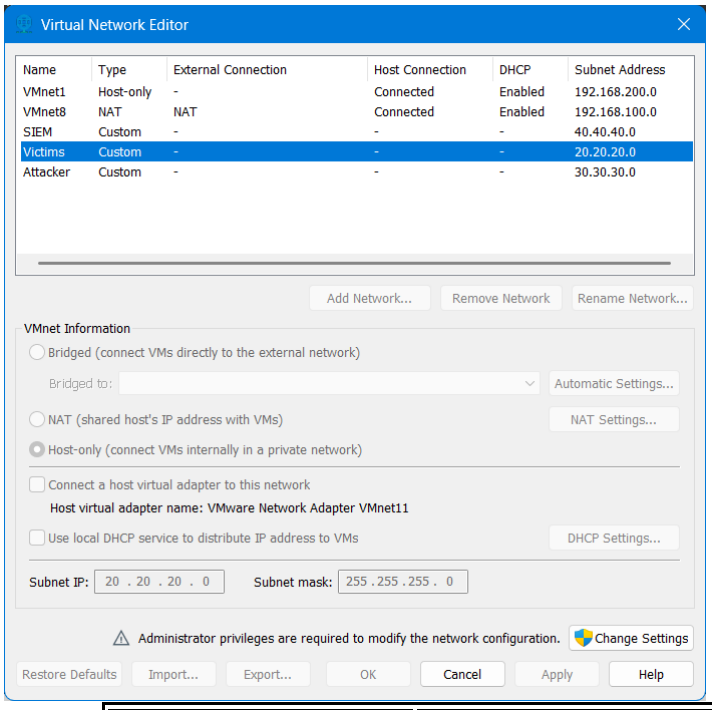


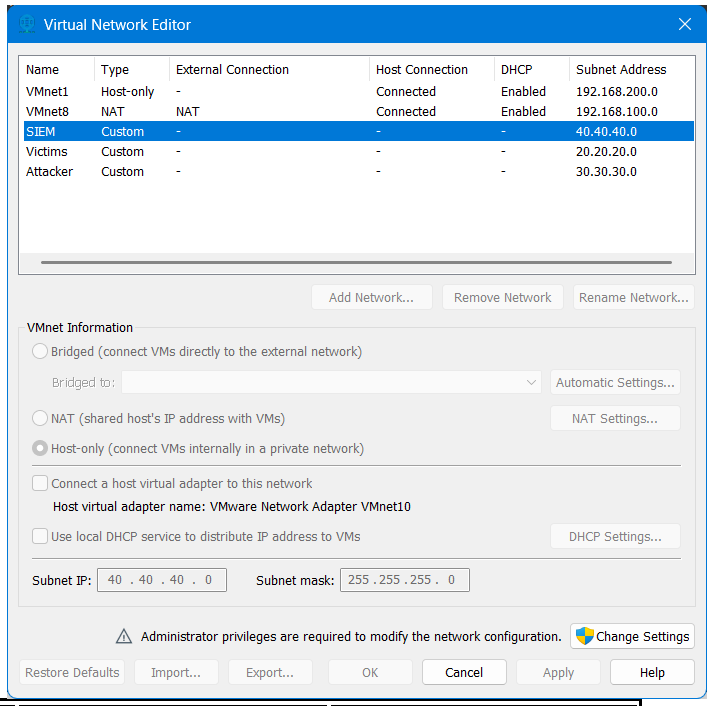
Host & VMware Configuration

Host: Windows 11
Hypervisor: VMware Workstation Pro
Installation: https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

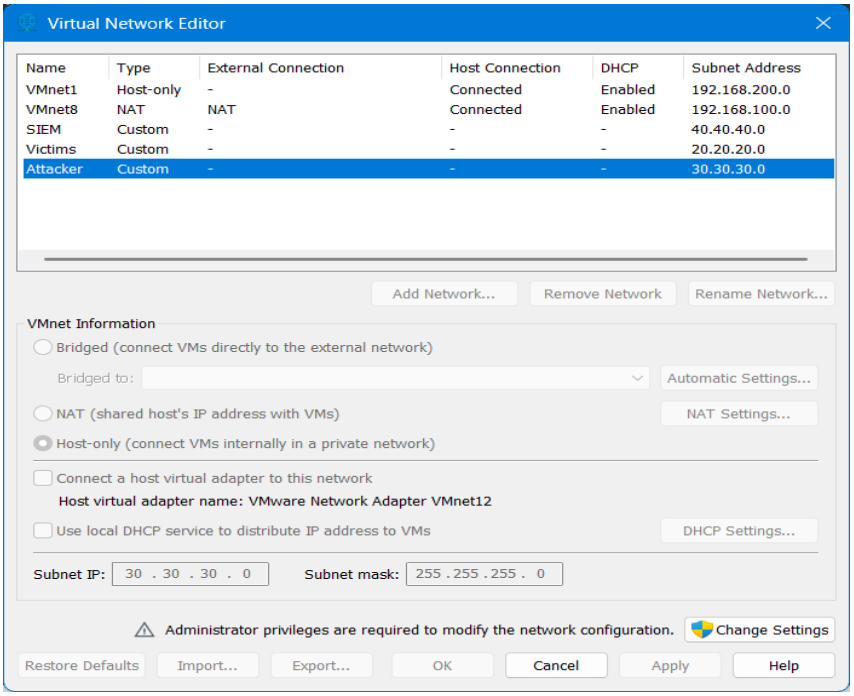
To Create Custom Networks:

- In **Virtual Network Editor** → **Edit**:
- Add 3 virtual network cards with the same settings in screenshots





VMnet	Type	Purpose	Subnet
vmnet0	Bridged	pfSense WAN	Host Network
vmnet2	Custom	SIEM (LAN1)	40.40.40.0/24
vmnet3	Custom	Victims (LAN2)	20.20.20.0/24
vmnet4	Custom	Attacker (LAN3)	30.30.30.0/24



Virtual Machines Deployment

1. pfSense Firewall

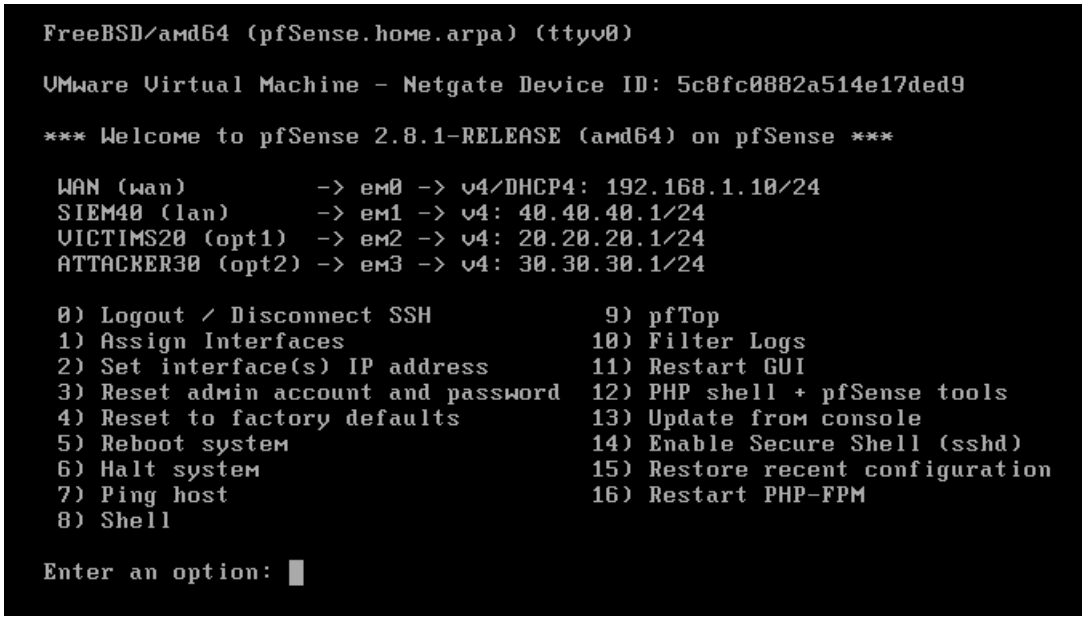
ISO: pfSense: https://www.pfsense.org/download/

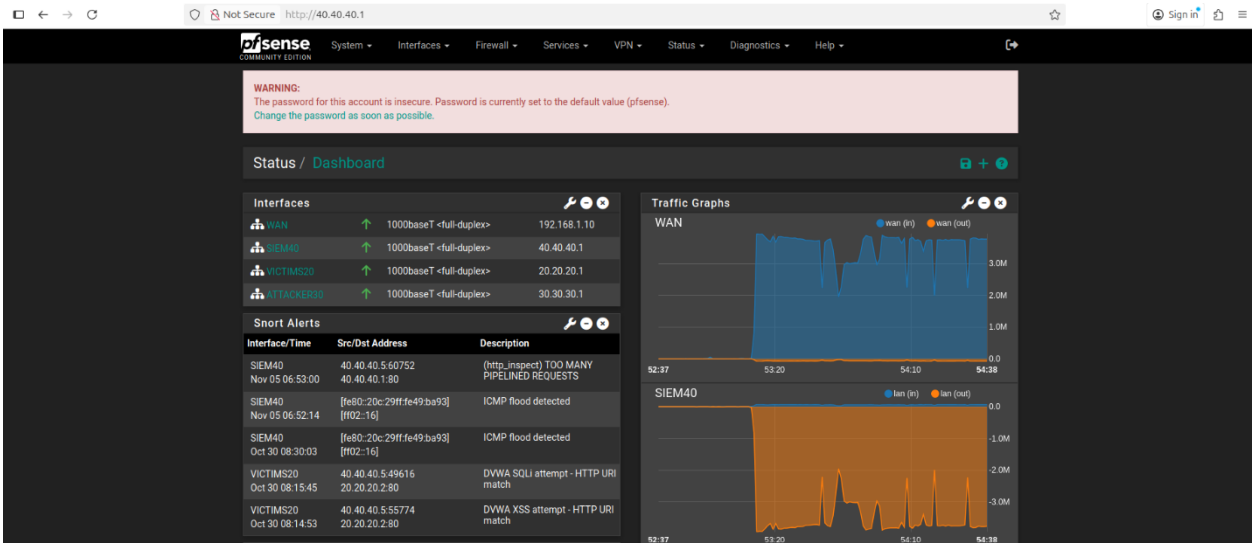
Steps:

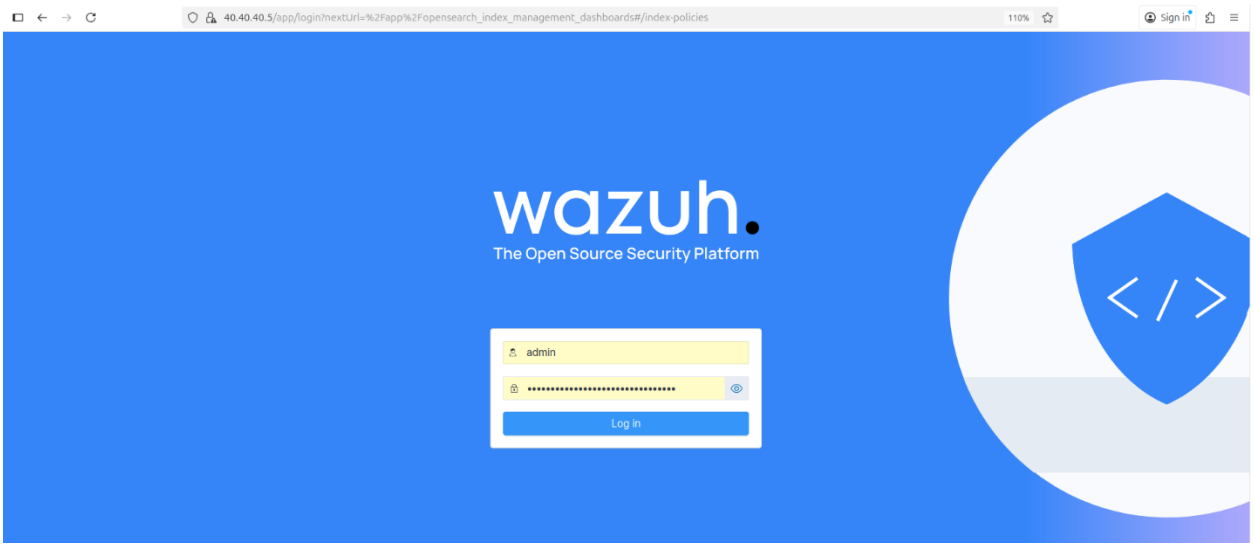
1. Create new VM → *FreeBSD 64-bit* → attach pfSense ISO.
2. Add 4 NICs (WAN + 3 LANs).
3. Boot installer → accept defaults → auto (UFS) partition → reboot.
4. Assign interfaces:
    - WAN: DHCP
    - LAN1: 40.40.40.1/24
    - LAN2: 20.20.20.1/24
    - LAN3: 30.30.30.1/24
5. Enable **DHCP server** for each LAN:
    - LAN1: 40.40.40.50–100
    - LAN2: 20.20.20.50–100

- LAN3: 30.30.30.50–100

6. Access the GUI using the Ip and credentials appeared at the end of installation







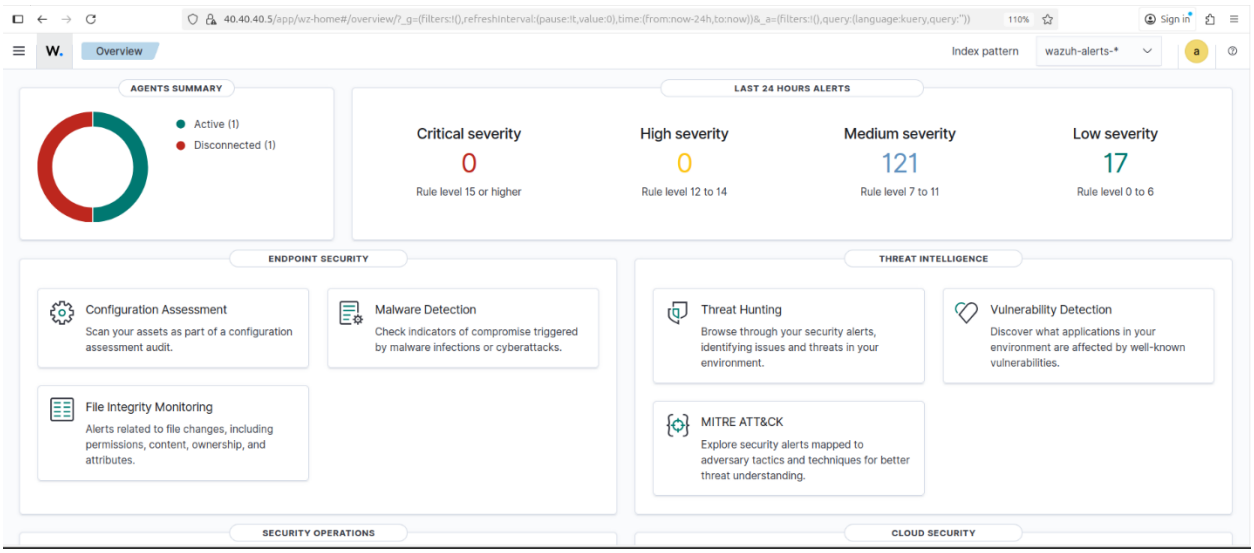
---

2. Ubuntu — Wazuh Manager

Network: LAN1 (SIEM) DHCP

Steps:

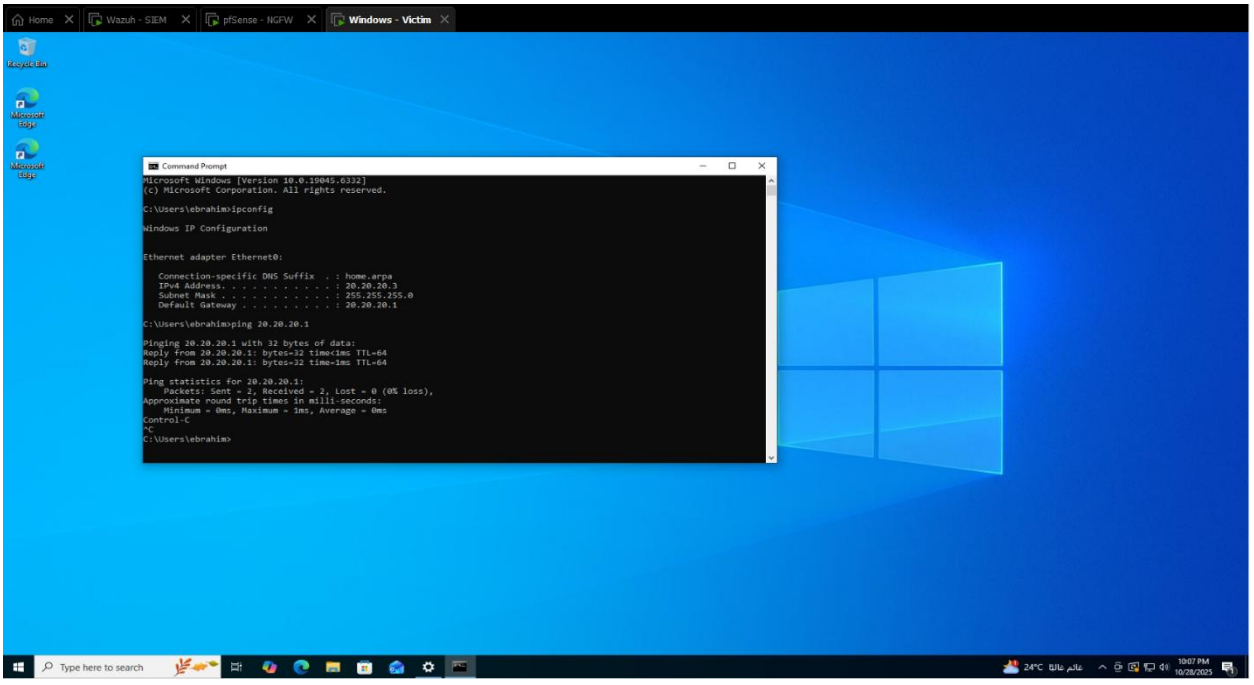
1. Install Ubuntu: https://ubuntu.com/download/desktop
2. Choose automatic (DHCP) network setup.
3. After rebooting, run in terminal: sudo apt update && sudo apt upgrade -y
4. Verify assigned IP, run: ip a
5. Install Wazuh: https://documentation.wazuh.com/current/quickstart.html
6. Access the GUI using the Ip and credentials appeared at the end of installation



3. Windows 10 (Victim)

Network: LAN2 (Victims) — DHCP

- Install Windows normally: https://www.microsoft.com/en-us/software-download/windows10
- Connect to pfSense LAN2 network, verify DHCP-assigned IP (ipconfig).
- Test connectivity to pfSense and Wazuh.

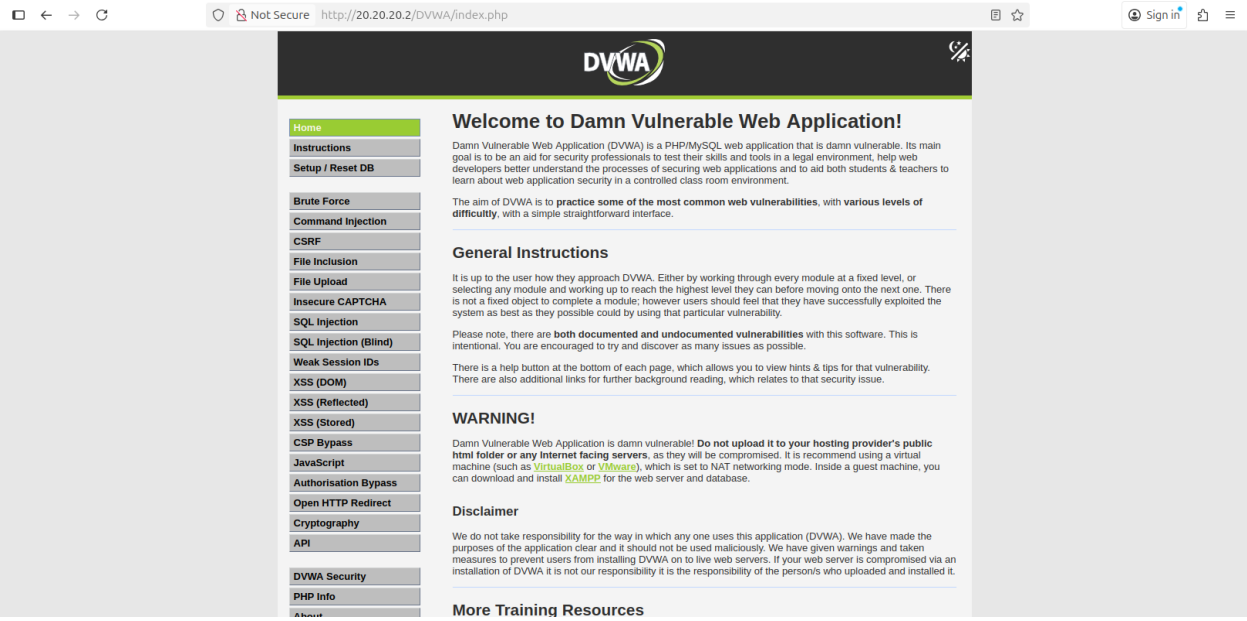


4. DVWA Server (Ubuntu)

Network: LAN2 (Victims) — DHCP

Install LAMP stack + DVWA:

- sudo apt update
- sudo apt install apache2 mysql-server php php-mysqli git -y
- cd /var/www/html
- sudo git clone https://github.com/digininja/DVWA.git
- sudo chown -R www-data:www-data DVWA
- sudo systemctl enable --now apache2 mysql
- Open http://<DVWA-IP>/DVWA

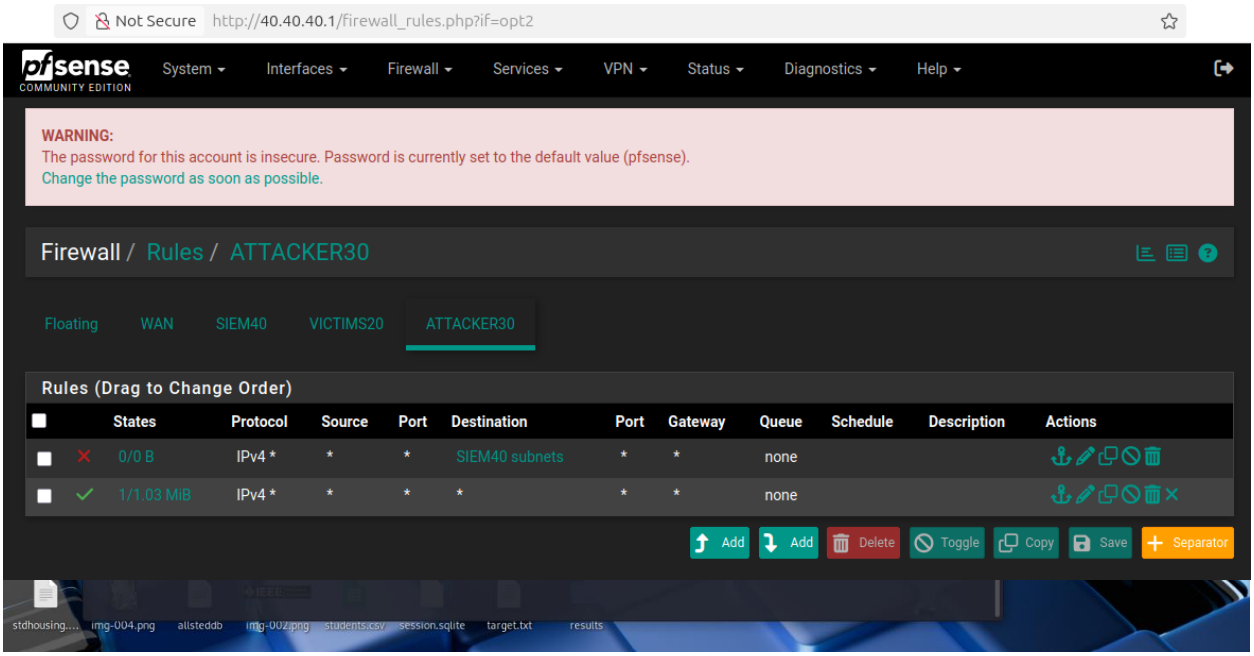


---

5. Kali Linux (Attacker)

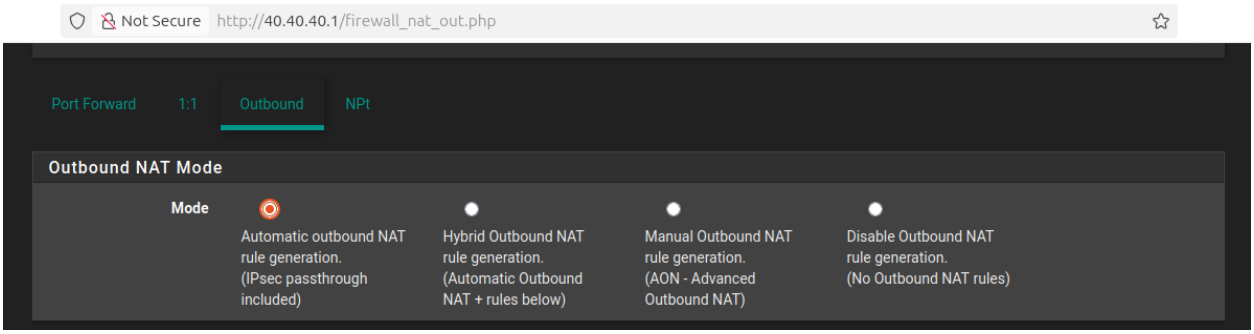
Network: LAN3 (Attacker) — DHCP

- Install from ISO: https://www.kali.org/get-kali/#kali-virtual-machines
- Confirm IP assignment using ip a.
- Test ping to pfSense and Victims network.



6: pfSense Firewall & NAT Configuration

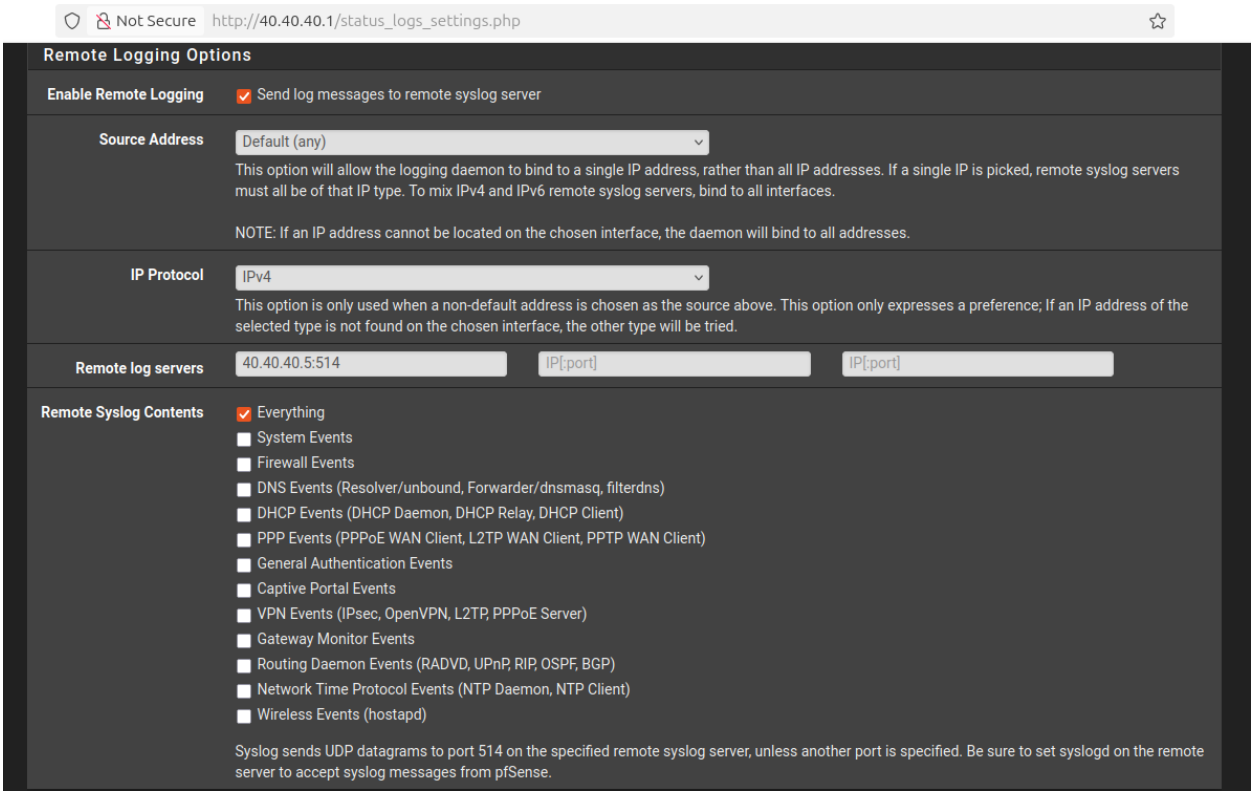
- **NAT:** Automatic mode



- **Firewall Rules:**
  - LAN1 (SIEM): Allow outbound
  - LAN2 (Victims): Allow SIEM + WAN
  - LAN3 (Attacker): Allow Victims, Block to SIEM, Allow to WAN

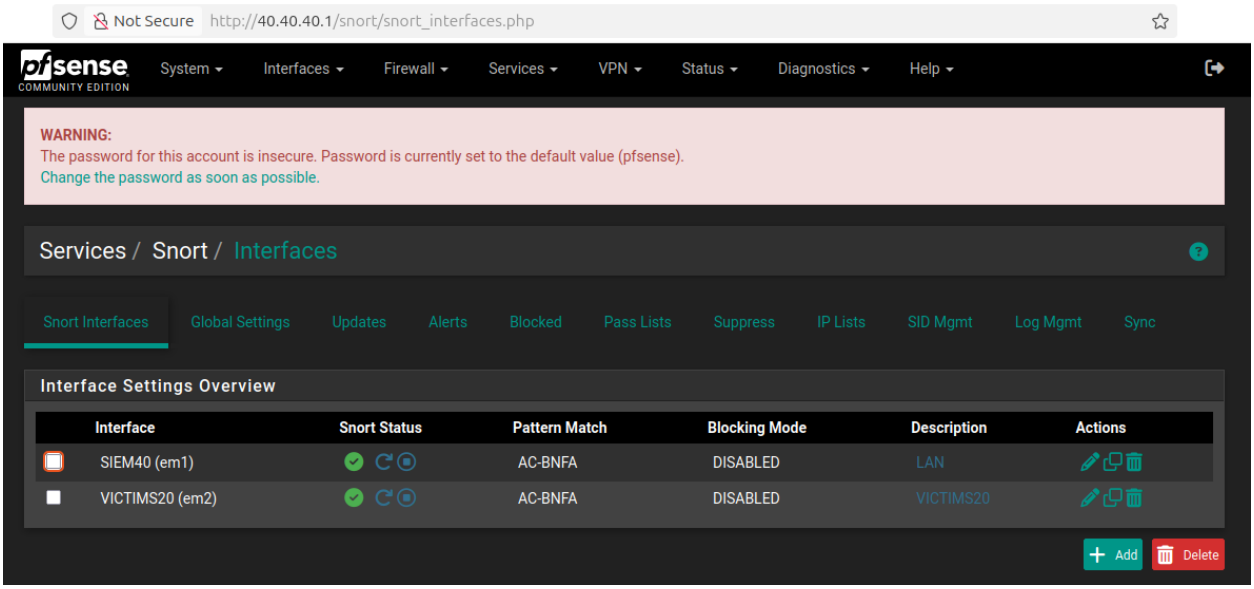
Configure pfSense Remote Syslog → Wazuh

- Navigate to **Status → System Logs → Settings → Remote Logging Options**
- Enable “Send log messages to remote syslog server.”
- Server: <Wazuh-IP>
- Port: 514 UDP
- Categories: Everything [All Logs]



7: Snort Installation

- Install via **System → Package Manager → Snort**
- Add interfaces (WAN + Victims)
- Enable “Send Alerts to System Log”

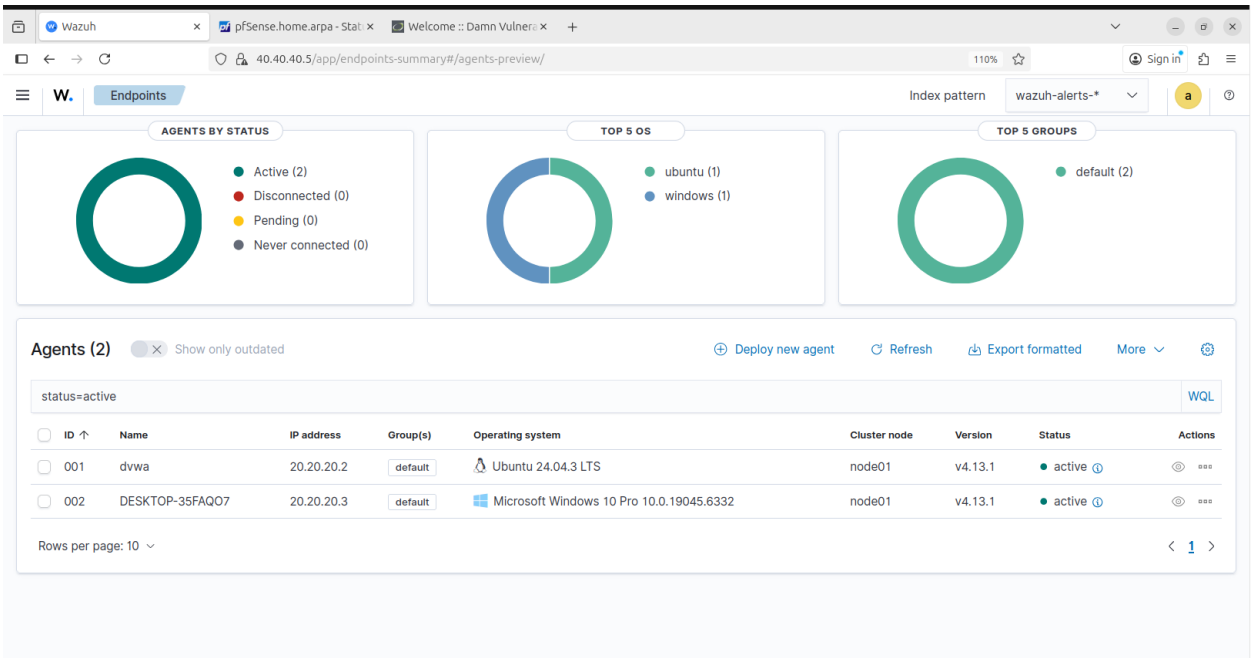


9: Install Wazuh Agents

Follow Documentation:

Linux (DVWA) : https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html

Windows: https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-windows.html



10: Verification & Testing

1. **Connectivity Tests:** Ping & verify DHCP leases.
2. **pfSense Log Flow:** Generate login and rule events.
3. **Snort Alerts:** Run nmap <DVWA-IP> from Kali, verify alerts in Wazuh.
4. **Agent Logs:** Trigger failed login or file changes, confirm detection.

11: Deliverables Checklist

Item	Description
SOC Diagram	DHCP-based topology diagram
Firewall Rules	pfSense rules screenshots
Snort Config	“Send alerts to syslog” proof
Wazuh Dashboard	Log and alert evidence
Screenshots	As listed
Snapshots	post_pfSense, post_Wazuh, post_Agents

12: Troubleshooting Reference

Problem	Check
DHCP not assigning	Verify each LAN DHCP server enabled
No pfSense logs in Wazuh	Ensure UDP 514 open and ossec file
Agents not registering	Check UDP 1514 and firewall
Snort alerts missing	Confirm syslog export
Dashboard access issues	Allow ports 443/5601

CHAPTER 2 — USE CASE DEVELOPMENT

(Attack Scenarios, Execution, Detection, and Correlation)

The purpose of this chapter is to document the offensive security use cases executed in the Mini-SOC environment, along with their detection mechanisms, logs generated, MITRE ATT&CK mappings, and correlation results within the Wazuh SIEM platform.

Each use case follows a consistent enterprise-grade structure:

1. **Use Case Overview**
2. **Objective & Purpose**
3. **Prerequisites & Environment Conditions**
4. **Execution Steps (Attacker Actions)**
5. **Telemetry Generated (Network + Endpoint)**
6. **Detection Logic (Snort + Wazuh)**
7. **MITRE ATT&CK Mapping**
8. **Screenshots to include**

---

1. SIEM Use Case Document: Correlated SQL Injection (NIDS + EDR)

1. Metadata

Field	Details
Use Case ID	UC-SQLI-2025-001
Use Case Title	**Correlated SQL Injection (SQLi) Attempt (NIDS + EDR)**
Priority / Severity	**High** (Correlated Level 10 Alert)
Status	Production
Author	Ebrahim Ezzat

Field	Details
Date Created	2025-11-01
Last Updated	2025-11-12

2. Use Case Overview

2.1. Description

This case detects a confirmed SQL Injection (SQLi) attack by correlating two separate detection mechanisms:

1. **Network Intrusion Detection (NIDS):** A Snort alert that identifies a malicious SQLi pattern in network traffic.
2. **Endpoint Detection and Response (EDR):** A Wazuh agent rule that identifies the same SQLi pattern in the target web server's access logs.

When both the network *and* the host server log an SQLi attempt from the same source within a short time frame, this use case triggers a high-severity alert, indicating a high-fidelity, confirmed attack.

2.2. Objective

- To detect active SQL Injection attempts against the "DVWA" web application (20.20.20.2).
- To significantly **reduce false positives** by requiring both NIDS and EDR validation.
- To elevate the alert severity from a standard "attempt" (Level 6) to a "confirmed correlated attack" (Level 10) for immediate SOC triage.

2.3. Threat Scenario

An external attacker (e.g., 30.30.30.2) uses a tool (like sqlmap, burp suite, or a simple browser) to send a malicious HTTP GET request to the DVWA server (20.20.20.2). This request contains SQLi payloads (e.g., ...id=%27+or+1%3D1--) in the URL, attempting to bypass authentication or exfiltrate data.

3. Threat Intelligence & Mapping

3.1. MITRE ATT&CK

- **Tactic:** TA0001 - Initial Access
- **Technique:** T1190 - Exploit Public-Facing Application

3.2. Compliance Mapping

- **PCI-DSS:** 6.5, 11.4, 6.5.1
- **GDPR:** IV_35.7.d
- **NIST 800-53:** SA.11, SI.4
- **TSC:** CC6.6, CC7.1, CC8.1, CC6.1, CC6.8, CC7.2, CC7.3

---

4. Technical Requirements

4.1. SIEM

- **Platform:** Wazuh (Manager, Indexer, Dashboard)

4.2. Data Sources

1. **NIDS (Network):**
    a. **Source System:** pfSense Firewall with Snort
    b. **Logs:** Snort Alerts (forwarded via syslog to Wazuh Manager).
2. **EDR (Host):**
    a. **Source System:** DVWA Web Server (20.20.20.2)
    b. **Agent:** Wazuh Agent installed.
    c. **Logs:** Apache Access Log (/var/log/apache2/access.log), Apache Error Log (/var/log/apache2/error.log).

4.3. Wazuh Components (Decoders & Rules)

- **Custom Snort Decoder:**

```
<decoder name="snort">
  <prematch>DVWA SQLi attempt</prematch>
</decoder>
```

- **Custom Snort Rule (NIDS-based):**

```
<rule id="100900" level="6">
 <decoded_as>snort</decoded_as>
 <match>DVWA SQLi attempt - HTTP URI match</match>
<description>Snort Alert — DVWA SQL Injection attempt detected</description>
<group>snort,sql_injection,attack,local,sqlinjection</group>
</rule>
```

- **Standard Web Log Rules (EDR):**

```
<group name="web,accesslog,">
<rule id="31100" level="0">
<category>web-log</category>
<description>Access log messages grouped.</description>
</rule>
```

```
<rule id="31164" level="6">
<if_sid>31100</if_sid>
<url>=%27|select%2B|insert%2B|%2Bfrom%2B|%2Bwhere%2B|%2Bunion%2B</url>
<description>SQL injection attempt.</description>
<group>attack,sqlinjection,attack,...</group>
</rule>
```

- **Correlation Rule (Parent Rule):**

```
<rule id="100911" level="10" frequency="2" timeframe="120">
<if_matched_group>sqlinjection</if_matched_group>
<global_frequency />
<description>Confirmed SQL Injection (Correlated NIDS + EDR)</description>
<group>correlation,confirmed,sqlinjection,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.1,gdpr
_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_C
C6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
<mitre> <id>T1055</id> <id>T1190</id>  </mitre>
</rule>
```

---

5. Detection & Correlation Logic

This use case operates in three stages:

Stage 1: NIDS Detection (Snort Alert)

1) The attacker (30.30.30.2) sends the malicious payload.
2) The traffic passes through the pfSense firewall.
3) The **Snort Rule (sid:2000001)** matches the URI pattern (/DVWA/vulnerabilities/sqli/) and the payload (union, select, etc.).
4) Snort generates the alert: ... [1:2000001:2] DVWA SQLi attempt - HTTP URI match {TCP} 30.30.30.2: ANY -> 20.20.20.2:80
5) Wazuh ingests this log. The custom snort decoder parses it.
6) **Wazuh Rule 100900** fires (Level 6) because it matches the description text.
7) This alert is added to the **SQL injection** group.

Stage 2: EDR Detection (Apache Log)

1) The malicious request hits the DVWA server (20.20.20.2).
2) The Apache web server logs the request in /var/log/apache2/access.log:
3) 30.30.30.2 - - ... "GET /DVWA/vulnerabilities/sqli/?id=%27+or+1%3D1--&Submit=Submit HTTP/1.1" 500 ...
4) The Wazuh agent on DVWA reads this log line.
5) **Wazuh Rule 31164** fires (Level 6) because the url field contains a forbidden pattern (=%27 or %2Bunion%2B, etc.).
6) This alert is also added to the **sqlinjection** group.

Stage 3: Correlation (Parent Rule)

1) **Wazuh Rule 100911** is triggered.
2) Its logic is: frequency="2" and timeframe="120" for the group sqlinjection.
3) This means: "If **two** alerts from the sqlinjection group are seen within **120 seconds**, fire this rule."
4) Since one alert came from Snort (NIDS) and one from Apache (EDR), the condition is met as **[global_frequency]** Specifies that the events of all agents will be contemplated when using the frequency and timeframe options. By default, only the events generated by the same agent will be taken into account to increase the frequency counter for a rule.
5) A new, high-severity **Level 10** alert is generated with the description: **"Confirmed SQL Injection (Correlated NIDS + EDR)"**.

---

6. Alert & Triage Details

- **Alert Title:** Confirmed SQL Injection (Correlated NIDS + EDR)
- **Alert Rule ID:** 100911

- **Severity: High (Level 10)**
- **Key Fields for Triage:**
  - src_ip: (Attacker) 40.40.40.5
  - dst_ip: (Victim) 20.20.20.2
  - url.full: (Payload) /DVWA/vulnerabilities/sqli/?id=%27+or+1%3D1--&Submit=Submit
  - snort.msg: DVWA SQLi attempt - HTTP URI match
  - http.status_code: 500 (Indicates the server failed to process the bad syntax, confirming impact).
- **Fidelity: High.** This alert is a correlation from two different, reliable sources (network and host). It is almost certainly a **True Positive**.

---

7. Validation & Testing

Test Scenario:

1) From the Kali machine (30.30.30.2), open a web browser.
2) Navigate to the DVWA SQLi page: http://20.20.20.2/DVWA/vulnerabilities/sqli/
3) In the "User ID" input box, type the payload ' or 1=1-- and click "Submit".
4) Alternatively, send the full GET request via the browser bar or curl: curl http://20.20.20.2/DVWA/vulnerabilities/sqli/?id=%27+or+1%3D1--&Submit=Submit

Expected Outcome:

1) Within the Wazuh dashboard, you will first see two (Level 6) alerts: one from Snort (Rule 100900) and one from web-log (Rule 31164).
2) Within 120 seconds, a single **Level 10** alert (Rule 100911) will appear, confirming the correlation.

---

8. Incident Response Playbook

1. Triage (Immediate)

- Acknowledge the Level 10 alert (Rule 100911). This is a high-priority event.

2. Investigation

1) Identify the **Attacker IP** (src_ip) and **Victim Server** (dst_ip) from the alert.
2) Review the correlated alerts (the "children" of the 100911 alert).
   - Confirm the Snort alert message.

    - Confirm the Apache access.log and inspect the url.full field to see the exact payload used.
3) Check the Apache error.log for the same timeframe. Look for database errors (e.g., mysqli_sql_exception: You have an error in your SQL syntax) which further prove the attack hit the application's database layer.
4) Check access.log for *subsequent* activity from the src_ip. Did they receive a 200 OK? Did they try to access other pages?

3. Containment (Immediate)

1) **Action:** Immediately block the **Attacker IP** (30.30.30.2) on the pfSense firewall to prevent further attacks.
2) **Action:** If the investigation shows the attacker successfully logged in or exfiltrated data, escalate to the Tier 2 team to isolate the web server (20.20.20.2) from the rest of the network.

4. Eradication & Recovery

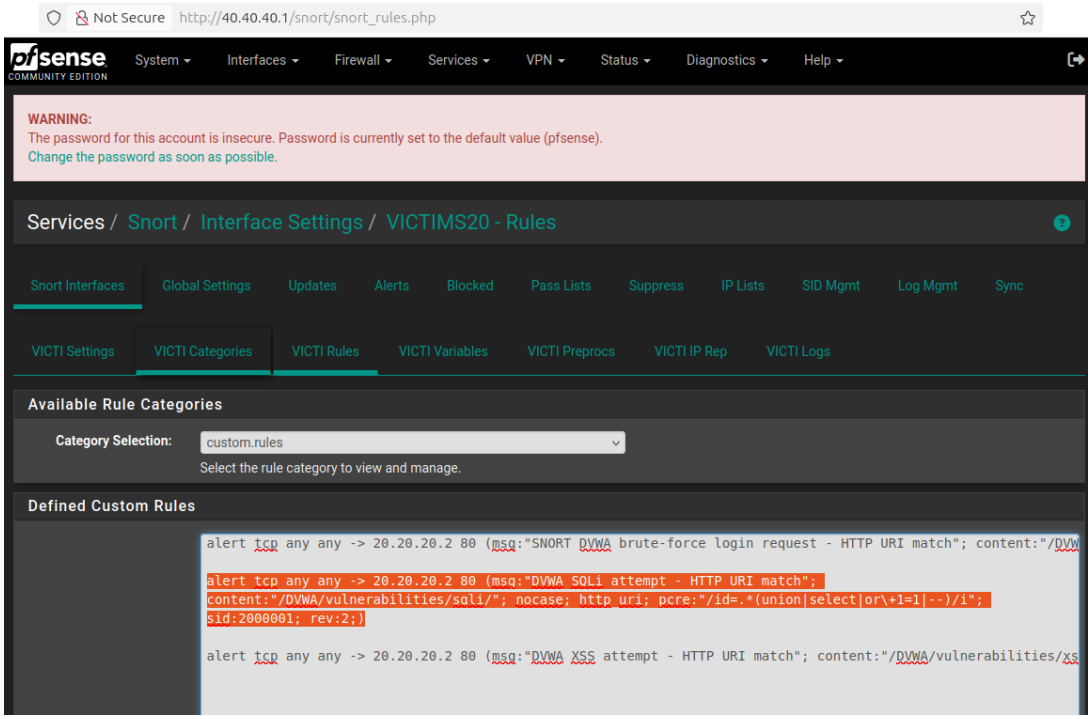
- Notify the Web Application Team and/or Development Team. Provide them with the url.full payload and the vulnerable page (/DVWA/vulnerabilities/sqli/).
- The vulnerability on the web application must be patched.
- Audit the database for any unauthorized modifications, new user accounts, or data deletion.

5. Escalation & Reporting

- Escalate to the SOC Lead and Incident Response (IR) Team.
- Document all findings and actions taken in the incident ticket.

Screenshots

1) Snort Rule



2) Decoder for snort SQLi log

```
<decoder name="snort">
  <prematch>DVWA SQLi attempt</prematch>
</decoder>
```

3) Logs in archive file (EDR and IDS)

```
2025 Nov 06 01:57:47 (dvwa) any->/var/log/apache2/error.log [Wed Nov 05 23:57:47.761446 2025] [php:error] [pid 6901] [client 30.30.30.2:36566] PHP Fatal error:  Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''' at line 1 in /var/www/html/DVWA/vulnerabilities/sqli/source/low.php:11\nStack trace:\n#0 /var/www/html/DVWA/vulnerabilities/sqli/source/low.php(11): mysqli_query()\n#1 /var/www/html/DVWA/vulnerabilities/sqli/index.php(34): require_once('...')\n#2 {main}\n  thrown in /var/www/html/DVWA/vulnerabilities/sqli/source/low.php on line 11, referer: http://20.20.20.2/DVWA/vulnerabilities/sqli/
2025 Nov 06 01:57:47 (dvwa) any->/var/log/apache2/access.log 30.30.30.2 - - [05/Nov/2025:23:57:47 +0000] "GET /DVWA/vulnerabilities/sqli/?id=%27or+1%3D1--&Submit=Submit HTTP/1.1" 500 295 "http://20.20.20.2/DVWA/vulnerabilities/sqli/" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
2025 Nov 06 01:57:48 wazuh->40.40.40.1 1 2025-11-06T01:57:48.471010+02:00 pfSense.home.arpa snort 61153 - - [1:2000001:2] DVWA SQLi attempt - HTTP URI match {TCP} 30.30.30.2:36566 -> 20.20.20.2:80
```

4) EDR Rules

```
<group name="web,accesslog,">
  <rule id="31100" level="0">
    <category>web-log</category>
    <description>Access log messages grouped.</description>
  </rule>
```

```
<rule id="31164" level="6">
  <if_sid>31100</if_sid>
  <url>=%27|select%2B|insert%2B|%2Bfrom%2B|%2Bwhere%2B|%2Bunion%2B</url>
  <description>SQL injection attempt.</description>
  <mitre>
    <id>T1055</id>
    <id>T1190</id>
  </mitre>
  <group>attack,sqlinjection,attack,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.1,gdpr_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2
      ,tsc_CC7.3,</group>
</rule>
```

5) Snort rule

```
<!-- sqli detection rule | author = ebrahim -->
<group name="local,snort_custom">

  <rule id="100900" level="6">
    <decoded_as>snort</decoded_as>
    <match>DVWA SQLi attempt - HTTP URI match</match>
    <description>Snort Alert - DVWA SQL Injection attempt detected</description>

    <group>snort,sql_injection,attack,local,sqlinjection</group>
  </rule>
```

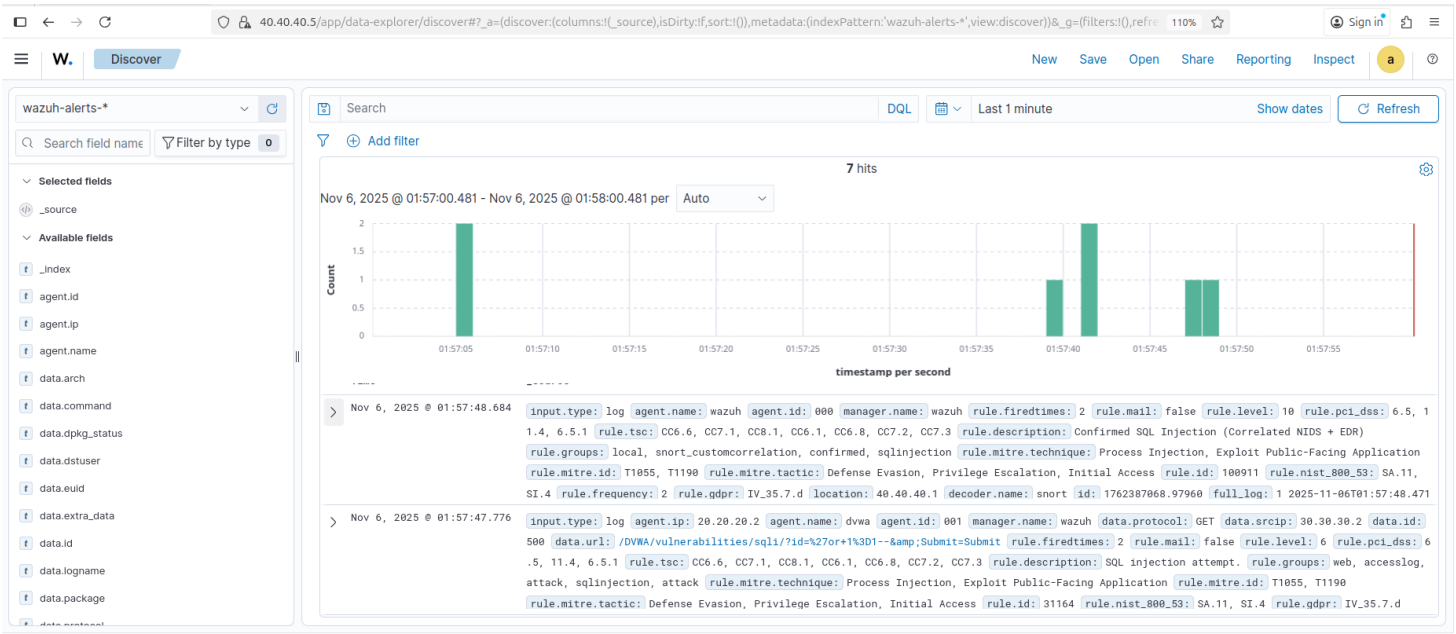
6)

7) Correlation rule

```
<rule id="100911" level="10" frequency="2" timeframe="120">
<if_matched_group>sqlinjection</if_matched_group>
<global_frequency />
<description>Confirmed SQL Injection (Correlated NIDS + EDR)</description>
<group>correlation,confirmed,sqlinjection,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.1,gdpr_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_CC6.1,tsc_CC6.8
    ,tsc_CC7.2,tsc_CC7.3,</group>
<mitre>
  <id>T1055</id>
  <id>T1190</id>
</mitre>
</rule>

</group>
```

8) Alerts in Wazuh Dashboard



2. SIEM Use Case Document: Correlated XSS Attack (NIDS + EDR)

1. Metadata

Field	Details
Use Case ID	UC-WAZ-002
Use Case Title	Correlated Reflected XSS Attack (NIDS + EDR)

Field	Details
Priority / Severity	High (Correlated Level 10 Alert)
Status	Production
Author	Ebrahim
Date Created	2025-11-08
Last Updated	2025-11-12

---

2. Use Case Overview

2.1. Description

This case detects a high-confidence Reflected Cross-Site Scripting (XSS) attack. It correlates two distinct signals:

- NIDS (Network): A Snort alert (sid:2000002) identifies a known XSS payload (e.g., <script>, onload=) in transit towards the DVWA web server.
- EDR (Host): The Wazuh agent monitoring the DVWA server's logs identifies that the same malicious request was successfully processed by the web server and returned an HTTP 200 OK status code.

2.2. Objective

- To detect active XSS attempts against the DVWA vulnerability page (/DVWA/vulnerabilities/xss_r/).
- To dramatically increase alert fidelity by confirming the server's response. An HTTP 200 OK response to an XSS payload strongly implies the payload was "reflected" back to the user, confirming the vulnerability.
- To escalate the alert from a simple "attempt" (Level 6) to a "confirmed correlated attack" (Level 10) requiring immediate investigation.

2.3. Threat Scenario

An external attacker (30.30.30.2) crafts a malicious URL containing a JavaScript payload and sends it to a victim (or tests it themselves). The request,

.../xss_r/?name=<script>alert("xss")</script>, is sent to the DVWA server (20.20.20.2). The server-side script insecurely takes the name parameter and renders it back on the HTML page, causing the browser to execute the attacker's script.

---

3. Threat Intelligence & Mapping

3.1. MITRE ATT&CK

- Tactic: TA0001 - Initial Access
- Technique: T1190 - Exploit Public-Facing Application
- Sub-Technique: T1059.007 - JavaScript/JScript (referenced by the underlying EDR rule 31105)

3.2. Compliance Mapping

- PCI-DSS: 6.5, 11.4, 6.5.7
- GDPR: IV_35.7.d
- NIST 800-53: SA.11, SI.4
- TSC: CC6.6, CC7.1, CC8.1, CC6.1, CC6.8, CC7.2, CC7.3

---

4. Technical Requirements

4.1. SIEM

- Platform: Wazuh (Manager, Indexer, Dashboard)

4.2. Data Sources

1. **NIDS (Network):**
    1. Source System: pfSense Firewall with Snort
    2. Logs: Snort Alerts (forwarded via syslog).
2. **EDR :**
    1. Source System: DVWA Web Server (20.20.20.2)
    2. Agent: Wazuh Agent installed.
    3. Logs: Apache Access Log (/var/log/apache2/access.log).

4.3. Wazuh Components (Decoders & Rules)

- **Custom Snort Decoder:**

```
<decoder name="snortxss">
```

```
  <prematch>DVWA XSS attempt</prematch>
</decoder>
```

- **Snort Alert Rule:**

```
<rule id="100901" level="6">
  <decoded_as>snortxss</decoded_as>
  <match>DVWA XSS attempt - HTTP URI match</match>
  <description>Snort Alert — DVWA XSS attempt detected</description>
  <group>snort,xss,attack,local</group>
</rule>
```

- **EDR Alert Rules:**

```
<rule id="31105" level="6">
  <if_sid>31100</if_sid>
  <url>%3Cscript|%3C%2Fscript|script>|script%3E|SRC=javascript|...</url>
  <description>XSS (Cross Site Scripting) attempt.</description>
  <group>attack,xss...</group>
</rule>
```

```
<rule id="31106" level="6" overwrite="yes">
  <if_sid>31103, 31104, 31105</if_sid> <id>^200</id> <description>A web attack
returned code 200 (success).</description>
  <group>attack,xss,...</group>
</rule>
```

- **Correlation Rule (Parent Rule):**

```
<rule id="100912" level="10" frequency="2" timeframe="120">
  <if_matched_group>xss</if_matched_group>
  <global_frequency/>
  <description>Confirmed XSS Attack (Correlated NIDS + EDR)</description>
  <group>correlation,confirmed,xss,attack,local</group>
</rule>
```

5. Detection & Correlation Logic

This use case's logic is highly effective because it confirms *success*, not just *intent*.

Stage 1: Snort Detection (The Intent)

- Attacker (30.30.30.2) sends the GET request with the XSS payload.
- Snort (sid:2000002) inspects the traffic and matches the payload pattern (<script> or %3Cscript).
- Snort generates the alert: ... [1:2000002:1] DVWA XSS attempt - HTTP URI match {TCP} 30.30.30.2:56752 -> 20.20.20.2:80
- Wazuh ingests this, the snortxss decoder parses it, and Rule 100901 fires (Level 6).
- This first alert is added to the xss group.

Stage 2: EDR Detection (The Success)

- The malicious request hits the DVWA server (20.20.20.2).
- Apache processes the request and logs it to access.log:
  30.30.30.2 - - ... "GET /DVWA/vulnerabilities/xss_r/?name=%3Cscript%3E... HTTP/1.1" 200 1859 ...

1. The Wazuh agent reads this log line.
2. First, the child rule 31105 (XSS attempt) fires because the URL contains %3Cscript.
3. Immediately after, the parent rule 31106 (Attack Success) checks its conditions.
    1. if_sid condition is met (because 31105 fired).
    2. id condition (^200) is met (because the log line explicitly shows HTTP 200).
4. Rule 31106 fires (Level 6) and overwrites the 31105 alert.
5. This second, more specific alert is also added to the xss group.

Stage 3: Correlation (The Confirmation)

1. Rule 100912 is triggered because its if_matched_group="xss" condition is met.
2. The rule's logic (frequency="2" timeframe="120") is now evaluated.
3. It finds two alerts in the xss group within 120 seconds:
    1. Alert 1: 100901 (Snort NIDS)
    2. Alert 2: 31106 (EDR 200 OK)
4. The condition is met. A new, high-severity Level 10 alert is generated: "Confirmed XSS Attack (Correlated NIDS + EDR)".

6. Alert & Triage Details

- **Alert Title:** Confirmed XSS Attack (Correlated NIDS + EDR)
- **Alert Rule ID:** 100912
- **Severity**: High (Level 10)
- **Alert JSON (Key Fields):**
    1. **rule.id:** 100912
    2. **rule.description:** Confirmed XSS Attack (Correlated NIDS + EDR)
    3. **data.srcip:** (Attacker) 30.30.30.2
    4. **agent.ip**: (Victim) 20.20.20.2
    5. **data.url:** (Payload) /DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22xss%22%29%3B%3C%2Fscript%3E
    6. **data.id:** (HTTP Status) 200
- Fidelity: Very High. The correlation of a NIDS signature with a 200 OK status from the host is a strong confirmation of a successful Reflected XSS.

7. Validation & Testing

- **Test Scenario:**

    1. From the attacker machine (30.30.30.2), open a browser.
    2. Paste the following URL into the address bar and press Enter:

```
http://20.20.20.2/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert(1)%3C/script%3E
```

    3. Alternatively, use curl from the terminal:

```
Curl http://20.20.20.2/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert(1)%3C/script%3E
```

- **Expected Outcome:**

    1. If testing in a browser, a JavaScript alert(1) box should pop up.
    2. Within the Wazuh Dashboard, you will first see two (Level 6) alerts: one from Snort (100901) and one from the EDR 200 OK rule (31106).
    3. Within 120 seconds, the parent Level 10 alert (100912) will appear.

8. Incident Response Playbook

1. Triage (Immediate)

- Acknowledge the Level 10 alert (100912). This is a high-priority event indicating a confirmed vulnerability.

2. Investigation

- Identify Attacker IP (data.srcip) and Victim Server (agent.ip).
- Review the data.url field. Decode the payload (e.g., %3C = <, %28 = () to understand the attacker's script.
    - *Payload:* <script>alert("xss") ;</script>
    - *Attacker's Goal:* Proof-of-Concept. This is likely a test. Be prepared for more malicious payloads (e.g., cookie stealers, keyloggers, BeEF hooks).
- Check the data.id field. Confirm it is 200. This verifies the server reflected the payload.
- Search all logs (e.g., access.log) for *other* activity from the data.srcip (30.30.30.2) before and after this event. Are they performing other scans? Did they try SQLi?

3. Containment (Immediate)

- Action: Block the Attacker IP (30.30.30.2) on the pfSense firewall to prevent further scanning and exploitation.

4. Eradication & Recovery

- This is the most critical step for XSS. The vulnerability is in the *application code*.
- Notify the Development Team / Application Owners.
- Provide them with this Use Case document, the exact payload (data.url), and the vulnerable page (/DVWA/vulnerabilities/xss_r/).
- The vulnerability must be patched by implementing output encoding (e.g., converting < to <) on the name parameter before rendering it to the page.

5. Escalation & Reporting

- Escalate to the SOC Lead and Application Security Team.
- Document all findings in the incident ticket. This alert is definitive proof of a vulnerability that must be fixed.

Screenshots

1. **Snort Rule:**

```
alert tcp any any -> 20.20.20.2 80 (msg:"DVWA XSS attempt - HTTP URI match";
content:"/DVWA/vulnerabilities/xss_r/"; nocase;
http_uri; pcre:"/name=.*(<script|%3Cscript|onload=|onerror=)/i"; sid:2000002; rev:1;)
```

2. **Decoder of Snort:**

```
<decoder name="snortxss">
  <prematch>DVWA XSS attempt</prematch>
</decoder>
```

3. **Logs from EDR and snort and edr rules:**

```
2025 Nov 09 22:25:43 wazuh->40.40.40.1 1 2025-11-09T22:25:43.215758+02:00 pfSense.home.arpa snort 61153 - - [1:2000002:1] DVWA XSS attempt - HTTP URI match {TCP} 30.30.30.2:58814 -> 20.20.20.2:80
2025 Nov 09 22:25:44 wazuh->40.40.40.1 1 2025-11-09T22:25:43.000000+02:00 pfSense.home.arpa nginx - - - 40.40.40.5 - - [09/Nov/2025:22:25:43 +0200] "POST /getstats.php HTTP/1.1" 200 129 "http://40.40.40.1/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:144.0) Gecko/20100101 Firefox/144.0"
2025 Nov 09 22:25:44 wazuh->40.40.40.1 1 2025-11-09T22:25:43.000000+02:00 pfSense.home.arpa nginx - - - 40.40.40.5 - - [09/Nov/2025:22:25:43 +0200] "POST /widgets/widgets/log.widget.php HTTP/1.1" 200 683 "http://40.40.40.1/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:144.0) Gecko/20100101 Firefox/144.0"
2025 Nov 09 22:25:44 (dvwa) any->/var/log/apache2/access.log 30.30.30.2 - - [09/Nov/2025:20:25:43 +0000] "GET /DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22xss%22%29%3B%3C%2Fscript%3E HTTP/1.1" 200 1859 "http://20.20.20.2/DVWA/vulnerabilities/xss_r/" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
```

4.

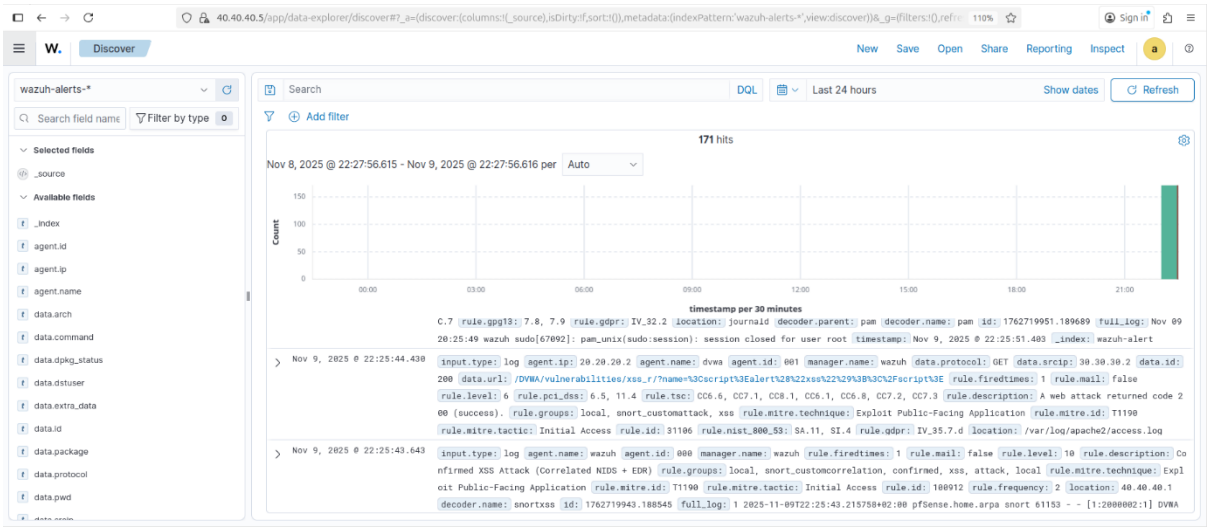
```
<rule id="31103" level="7">
  <if_sid>31100,31108</if_sid>
  <url>=select%20|select+|insert%20|%20from%20|%20where%20|union%20|</url>
  <url>union+|where+|null,null|xp_cmdshell</url>
  <description>SQL injection attempt.</description>
  <mitre>
    <id>T1190</id>
  </mitre>
  <group>attack,sql_injection,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.1,gdpr_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2
    ,tsc_CC7.3,</group>
</rule>
```

```
<rule id="31104" level="6">
  <if_sid>31100</if_sid>

  <!-- Attempt to do directory transversal, simple sql injections,
    -  or access to the etc or bin directory (unix). -->
  <url>%027|%00|%01|%7f|%2E%2E|%0A|%0D|../..|..\..|echo;|</url>
  <url>cmd.exe|root.exe|_mem_bin|msadc|/winnt/|/boot.ini|</url>
  <url>/x90/|default.ida|/sumthin|nsiislog.dll|chmod%|wget%|cd%20|</url>
  <url>exec%20|../..//|%5C../%5C|./../../|2e%2e%5c%2e|\x5C\x5C</url>
  <description>Common web attack.</description>
  <mitre>
    <id>T1055</id>
    <id>T1083</id>
    <id>T1190</id>
  </mitre>
  <group>attack,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.1,gdpr_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
```

5.

```
<rule id="31105" level="6">
  <if_sid>31100</if_sid>
  <url>%3Cscript|%3C%2Fscript|script>|script%3E|SRC=javascript|IMG%20|</url>
  <url>%20ONLOAD=|INPUT%20|iframe%20</url>
  <description>XSS (Cross Site Scripting) attempt.</description>
  <mitre>
    <id>T1059.007</id>
  </mitre>
  <group>attack,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.7,gdpr_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
```



6. correlation and custom rule:

```
<rule id="100912" level="10" frequency="2" timeframe="120">
  <if_matched_group>xss</if_matched_group>
  <global_frequency/>
  <description>Confirmed XSS Attack (Correlated NIDS + EDR)</description>
  <group>correlation,confirmed,xss,attack,local</group>
  <mitre>
    <id>T1190</id>
  </mitre>
</rule>
```

```
<!-- xss detection rule -->
<group name="local,snort_custom">
<rule id="100901" level="6">
  <decoded_as>snortxss</decoded_as>
  <match>DVWA XSS attempt - HTTP URI match</match>
  <description>Snort Alert - DVWA XSS attempt detected</description>
  <group>snort,xss,attack,local</group>
  <mitre>
    <id>T1055</id>
    <id>T1190</id>
  </mitre>
</rule>

<rule id="31106" level="6" overwrite="yes">
  <if_sid>31103, 31104, 31105</if_sid>
  <id>^200</id>
  <description>A web attack returned code 200 (success).</description>
  <mitre>
      <id>T1190</id>
  </mitre>
  <group>attack,xss,pci_dss_6.5,pci_dss_11.4,gdpr_IV_35.7.d,nist_800_53_SA.11,nist_800_53_SI.4,tsc_CC6.6,tsc_CC7.1,tsc_CC8.1,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
```

3. SIEM Use Case Document: Correlated RDP Brute Force (NIDS + HIDS/EDR)

1. Metadata

Field	Details
Use Case ID	UC-WAZ-003
Use Case Title	**Correlated RDP Brute Force Attack (NIDS + EDR)**
Priority / Severity	**High** (Correlated Level 10 Alert)
Status	Production
Author	Ebrahim
Date Created	2025-11-09
Last Updated	2025-11-12

---

2. Use Case Overview

2.1. Description

This use case detects a high-confidence RDP Brute Force attack by correlating two separate event sources:

1. **NIDS (Network):** A Snort threshold rule (sid:2000010) fires after detecting a high rate of RDP connection attempts (5 attempts in 60 seconds) from a single source IP.
2. **HIDS (Host/EDR):** The Wazuh agent on the target Windows server confirms these attempts are actual logon failures, specifically "Unknown user or bad password" (Windows Event ID 4625).

2.2. Objective

- To detect active, real-time RDP password guessing (Brute Force) attacks.
- To eliminate false positives from network scanners (which might trigger NIDS) by requiring host-level confirmation (HIDS) that logon failures are actually occurring.
- To escalate the alert from a "Potential" attempt (Level 5-6) to a "Confirmed Correlated Attack" (Level 10) for immediate incident response.

2.3. Threat Scenario

An external attacker (30.30.30.2) identifies an open RDP port (3389) on a Windows server (20.20.20.3). The attacker then uses an automated tool (like Hydra, Metasploit, or a custom script) to attempt thousands of password combinations against a known (ebrahim) or common (Administrator) username.

---

3. Threat Intelligence & Mapping

3.1. MITRE ATT&CK

- **Tactic:** TA0006 - Credential Access
- **Technique:** T1110 - Brute Force
- **Sub-Technique:** T1110.001 - Password Guessing

3.2. Compliance Mapping

- **PCI-DSS:** 10.2.4, 10.2.5
- **HIPAA:** 164.312.b
- **NIST 800-53:** AU.14, AC.7
- **GDPR:** IV_35.7.d, IV_32.2

---

4. Technical Requirements

4.1. SIEM

- **Platform:** Wazuh (Manager, Indexer, Dashboard)

4.2. Data Sources

1. **NIDS (Network):**
    - **Source System:** pfSense Firewall with Snort
    - **Logs:** Snort Alerts (forwarded via syslog).
2. **HIDS (Host/EDR):**

- **Source System:** Windows Server (20.20.20.3)
- **Agent:** Wazuh Agent installed.
- **Logs:** Windows Security Event Log.

4.3. Wazuh Components (Decoders & Rules)

- **Custom Snort Decoder:**

```
<decoder name="rdp">
 <prematch>Potential RDP Brute Force Detected</prematch>
</decoder>
```

- **Snort Rule (on pfSense):**

```
alert tcp any any -> 20.20.20.3 3389 (msg:"Potential RDP Brute Force Detected"; flow:to_server,established; threshold: type limit, track by_src, count 5, seconds 60; sid:2000010; rev:1;)
```

- **NIDS Alert Rule (Wazuh):**

```
<rule id="100101" level="6">
 <decoded_as>rdp</decoded_as>
 <match>Potential RDP Brute Force Detected</match>
 <description>Snort Alert - Potential RDP Brute Force Detected</description>
 <group>snort,attack,local,authentication_failed</group>
</rule>
```

- **HIDS Alert Rule (Wazuh):**

```
<rule id="60122" level="5" overwrite="yes">
 <if_sid>60105</if_sid>
 <field name="win.system.eventID">^529$|^4625$</field>
 <description>Logon Failure - Unknown user or bad password</description>
 <group>authentication_failed,pci_dss_10.2.4,...</group>
</rule>
```

- **Correlation Rule (Parent Rule):**

```
<rule id="100112" level="10" frequency="2" timeframe="120">
 <if_matched_group>authentication_failed</if_matched_group>
 <global_frequency/>
 <description>RDP Brute Force Detected (Correlated NIDS + EDR) </description>
 <group>authentication_failed,pci_dss_10.2.4,...</group>
```

```
</rule>
```

---

5. Detection & Correlation Logic

This use case perfectly demonstrates the NIDS + HIDS correlation strategy.

Stage 1: NIDS Detection (The Scan)

- The attacker (30.30.30.2) begins the attack.
- Snort (sid:2000010) tracks the connections to port 3389. Once it sees **5 connections in 60 seconds** from 30.30.30.2, it fires.
- Snort sends the alert log: ...[1:2000010:1] Potential RDP Brute Force Detected {TCP} 30.30.30.2:60788 -> 20.20.20.3:3389
- Wazuh ingests this. The rdp decoder parses it. **Rule 100101** fires (Level 6).
- This NIDS alert is added to the **authentication_failed** group.

Stage 2: HIDS Detection (The Failure)

- Simultaneously, the Windows Server (20.20.20.3) is rejecting these logon attempts.
- For each failure, it generates a **Windows Event ID 4625** (Logon Failure).
- The Wazuh agent on the server reads this event log (shown in your raw log screenshot) and sends it to the manager:

...EventID "4625" ... IpAddress "30.30.30.2" ... TargetUserName "ebrahim"

- Wazuh's built-in **Rule 60122** matches on EventID 4625.
- This HIDS alert (Level 5) is also added to the **authentication_failed** group.

Stage 3: Correlation (The Confirmation)

- **Rule 100112** is constantly monitoring the authentication_failed group.
- Its condition is frequency="2" timeframe="120".
- It sees two alerts arrive in the group within 120 seconds:
    1. Alert 1: 100101 (From Snort)
    2. Alert 2: 60122 (From Windows)
- The condition is met. A new, high-severity **Level 10** alert is generated: **"RDP Brute Force Detected (Correlated NIDS + EDR)"**.
- This exact alert (rule.id: 100112) is visible in your dashboard screenshot, confirming the logic works perfectly.

---

6. Alert & Triage Details

- **Alert Title:** RDP Brute Force Detected (Correlated NIDS + EDR)
- **Alert Rule ID:** 100112
- **Severity: High (Level 10)**
- **Key Fields for Triage:**
    1. **data.srcip** or win.system.eventdata.ipAddress: (Attacker) 30.30.30.2
    2. **agent.ip** or dstip: (Victim) 20.20.20.3
    3. **win.system.eventdata.targetUserName:** (Attempted User) ebrahim
    4. **win.system.eventID**: 4625
    5. **rule.groups:** snort, authentication_failed
- **Fidelity: Very High.** This is a confirmed, active attack.

---

7. Validation & Testing

Test Scenario:

1. From a designated "attacker" machine (e.g., a Kali VM with IP 30.30.30.2).
2. Use a tool like hydra or nmap's rdp-brute script to launch an attack against the Windows server (20.20.20.3).
3. Example command: hydra -l ebrahim -P /path/to/small_password_list.txt rdp://20.20.20.3

Expected Outcome:

1. The hydra tool will show multiple failed attempts.
2. Within 2 minutes (the timeframe), the Wazuh Dashboard will display the Level 10 alert (100112) after first showing the individual Level 6 (100101) and Level 5 (60122) alerts.

---

8. Incident Response Playbook (SOP)

1. Triage (Immediate)

- Acknowledge the Level 10 alert (100112). This is an active attack in progress.

2. Investigation

- Identify the **Attacker IP** (e.g., 30.30.30.2) and **Victim Server** (e.g., 20.20.20.3).
- Check the **Attempted Username** (win.system.eventdata.targetUserName).
    1. Is it a valid, known user (like ebrahim)? This is a *targeted* attack.
    2. Is it a generic user (Administrator, admin, guest)? This is a *generic* attack.

- **CRITICAL:** Immediately query all logs for the **Victim Server** for any **Successful Logons** (Windows Event ID 4624) originating from the **Attacker IP** (30.30.30.2). A success would escalate this incident to **"Confirmed Host Compromise"**.

3. Containment (Immediate)

- **Action:** Immediately **block the Attacker IP** (30.30.30.2) on the pfSense firewall. This stops the attack instantly.
- **Action:** If the targetUserName is a valid user (ebrahim) and the attack is ongoing, temporarily **disable the user account** to prevent a successful lockout or compromise.

4. Eradication & Recovery

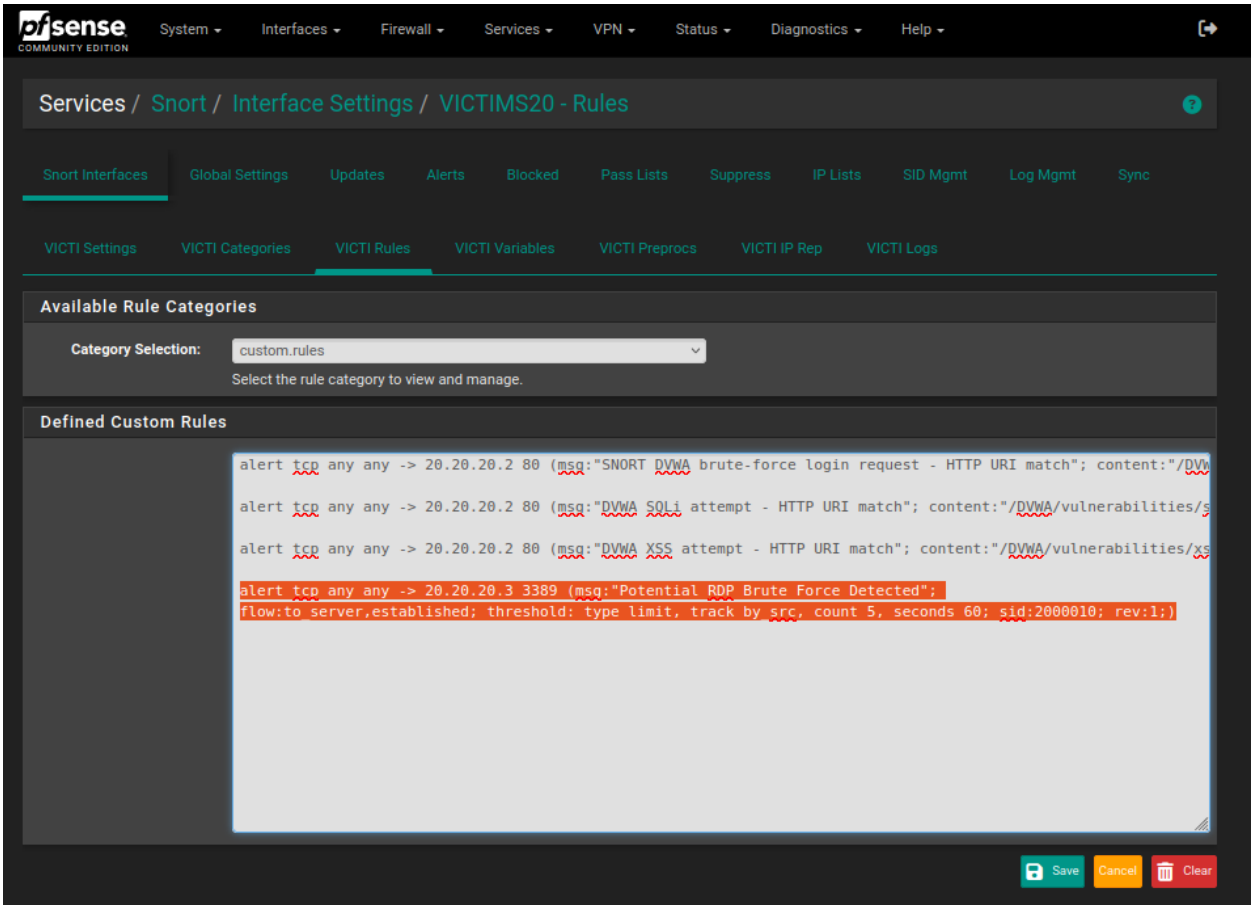
- If no successful login was detected:
    1. Keep the IP block in place.
    2. Re-enable the user account (if disabled) after forcing a password reset.
- If a successful login *was* detected:
    1. Escalate to the full "Compromised Host" playbook: Isolate the server from the network, begin forensic investigation, hunt for persistence.

5. Escalation & Reporting

- Notify the SOC Lead and the IT Infrastructure Team.
- Recommend long-term hardening for RDP:
    1. Place RDP behind a VPN.
    2. Implement Multi-Factor Authentication (MFA) for RDP.
    3. Enforce a strong Account Lockout Policy.
    4. Change the default RDP port (security by obscurity, but can help).

Screenshots

1. **Snort Rule:**



2. Decoder of Snort :

```
<decoder name="rdp">
  <prematch>Potential RDP Brute Force Detected</prematch>
</decoder>
```

3. Logs from EDR and snort :

```
<rule id="60000" level="0">
  <category>ossec</category>
  <decoded_as>windows_eventchannel</decoded_as>
  <field name="win.system.providerName">\.+</field>
  <options>no_full_log</options>
  <description>Group of windows rules.</description>
</rule>

<!-- Classification by channel -->
<rule id="60001" level="0">
  <if_sid>60000</if_sid>
  <field name="win.system.channel">^Security$</field>
  <options>no_full_log</options>
  <description>Group of Windows rules for the security channel.</description>
</rule>
```

4. Rules:

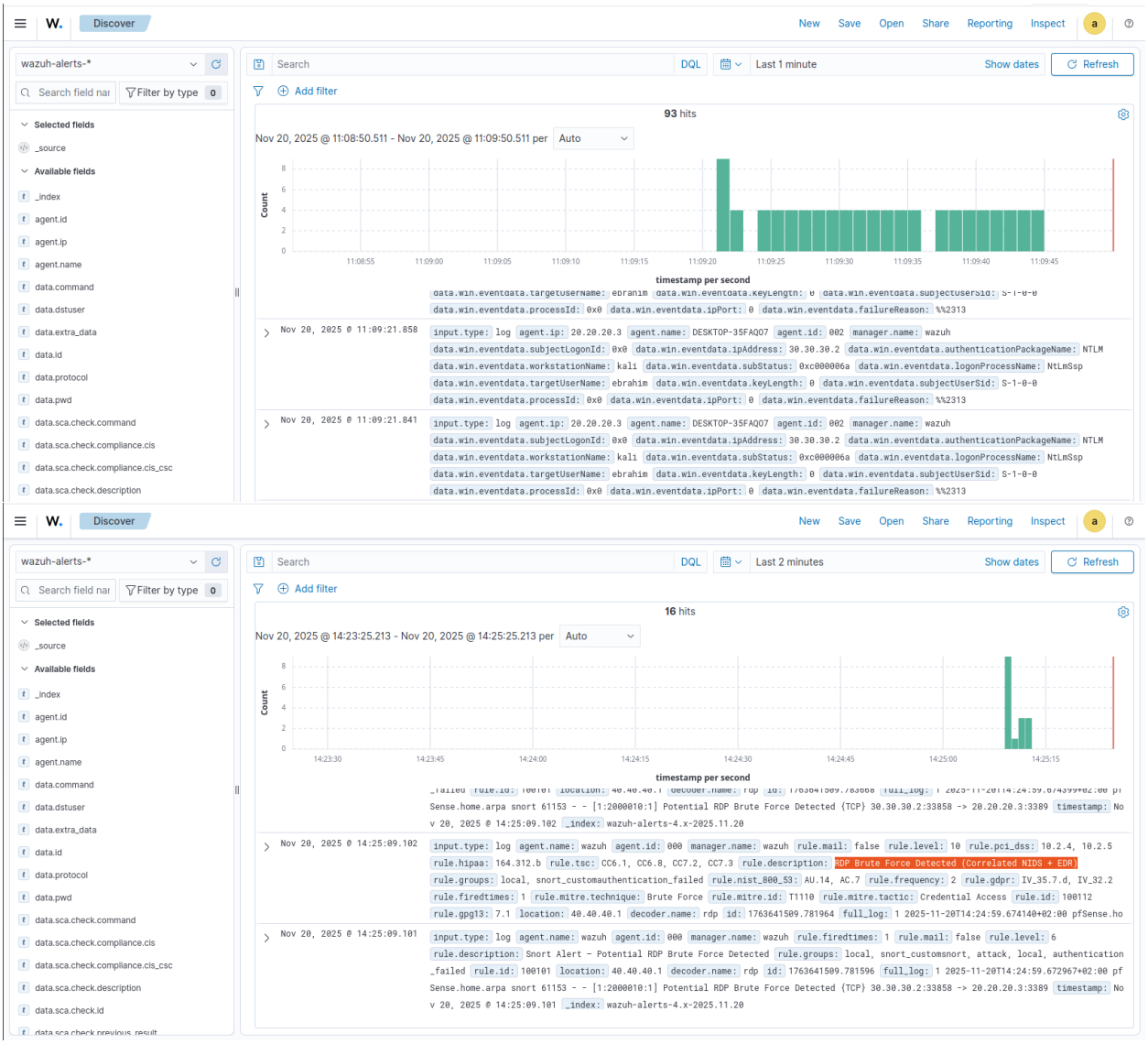
```
<group name="local,snort_custom">
<rule id="100101" level="6">
  <decoded_as>rdp</decoded_as>
  <match>Potential RDP Brute Force Detected</match>
  <description>Snort Alert - Potential RDP Brute Force Detected</description>
  <group>snort,attack,local,authentication_failed</group>
</rule>
```

```
<rule id="60104" level="5">
  <if_sid>60001</if_sid>
  <field name="win.system.severityValue">^AUDIT_FAILURE$|^failure$</field>
  <description>Windows audit failure event</description>
  <group>pci_dss_10.6.1,gdpr_IV_35.7.d,hipaa_164.312.b,nist_800_53_AU.6,tsc_CC7.2,tsc_CC7.3,</group>
  <options>no_full_log</options>
</rule>
```

5. Correlation rule :

```
<rule id="100112" level="10" frequency="2" timeframe="120">
<if_matched_group>authentication_failed</if_matched_group>
<global_frequency/>
<description>RDP Brute Force Detected (Correlated NIDS + EDR)</description>
<group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,gpg13_7.1,gdpr_IV_35.7.d,gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_AU.14,nist_800_53_AC.7,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2
    ,tsc_CC7.3,</group>
  <mitre>
    <id>T1110</id>
  </mitre>
</rule>
</group>
```

6. Alert in Wazuh:



CHAPTER 3 — ALERT TRIAGE & INCIDENT ANALYSIS

(Enterprise-Grade SOC Investigation & Correlation Workflows)

---

1. Introduction

This chapter documents the triage, investigation, and analytical processes applied to security alerts generated during the executed attack scenarios. The goal of SOC triage is to evaluate alert severity, determine incident impact, correlate multi-source telemetry, identify indicators of compromise (IOCs), and construct a validated attack timeline.

The triage methodology aligns with NIST 800-61, MITRE ATT&CK, and industry SOC workflows, focusing on:

- **Alert Verification**
- **Source Identification**
- **IOC Extraction**
- **Cross-Log Correlation**
- **Severity Classification**
- **Mitigation & Prevention Recommendations**

Each attack scenario from Chapter 2 is analyzed independently, demonstrating how the Mini-SOC performs coordinated detection using pfSense, Snort IDS, Windows/Linux telemetry, and Wazuh SIEM.

---

2. SOC Triage Workflow

All investigations follow a standard SOC workflow:

2.1 Initial Detection

Alert appears in Wazuh Dashboard → created by rule or correlation engine.

2.2 Alert Enrichment

- Extract source/destination IP address
- Identify the triggering rule
- Determine event frequency
- Review associated logs

2.3 Cross-Layer Correlation

Combine Snort alerts + Endpoint logs + Network telemetry.

2.4 Timeline Reconstruction

Build chronological order of events.

2.5 MITRE Technique Mapping

Relate adversary actions to ATT&CK tactics.

2.6 Analyst Judgement

Determine:

- False positive or True positive
- Severity
- Required response

---

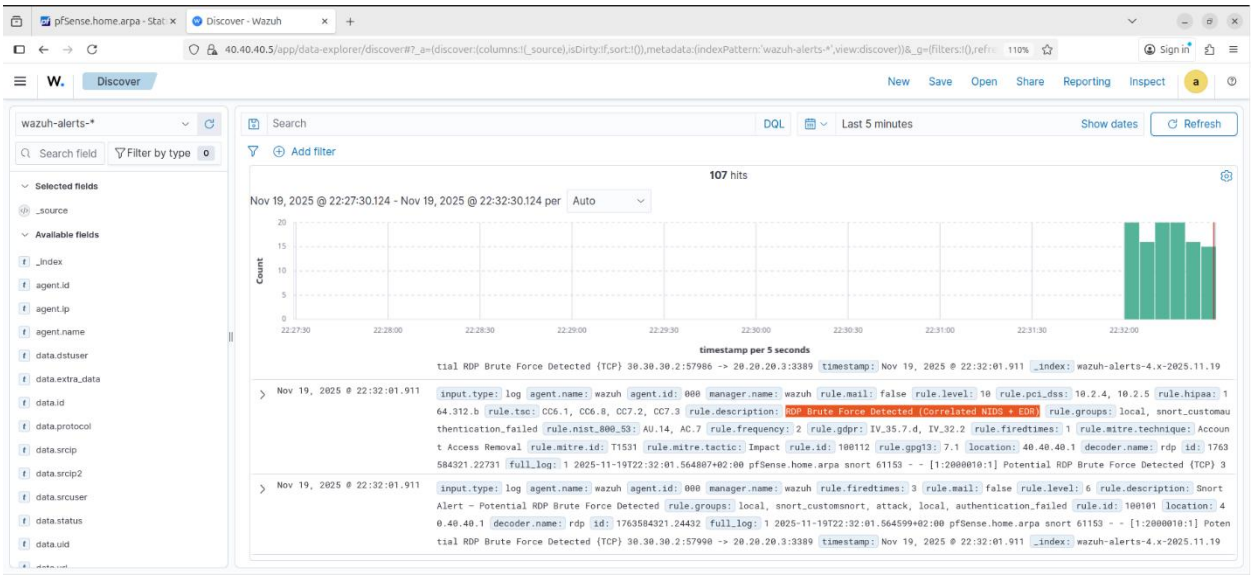
3. Use Case 1 – RDP Brute Force Alert Investigation

3.1 Alert Summary

Field	Value
Alert Name	RDP Brute Force Detected
Severity	Medium
Source	30.30.30.2 (Kali Linux)
Destination	20.20.20.3 (Windows)
Trigger	Multiple 4625 failures + Snort brute-force signature
MITRE Technique	T1110 (Brute Force)

Screenshot:

RDP Brute Force Detection



3.2 Detailed Event Sequence

Time	Component	Event
T00:00	Kali → Windows	First RDP attempt
T00:01	Windows Logs	Event 4625 (failed login)
T00:02	Snort IDS	Brute-force signature triggered
T00:03	Windows Logs	4625 events escalate
T00:04	Wazuh	Threshold correlation rule triggered
T00:05	SIEM	Medium-severity alert generated

3.3 Evidence Collected

Network-Layer Evidence (Snort)

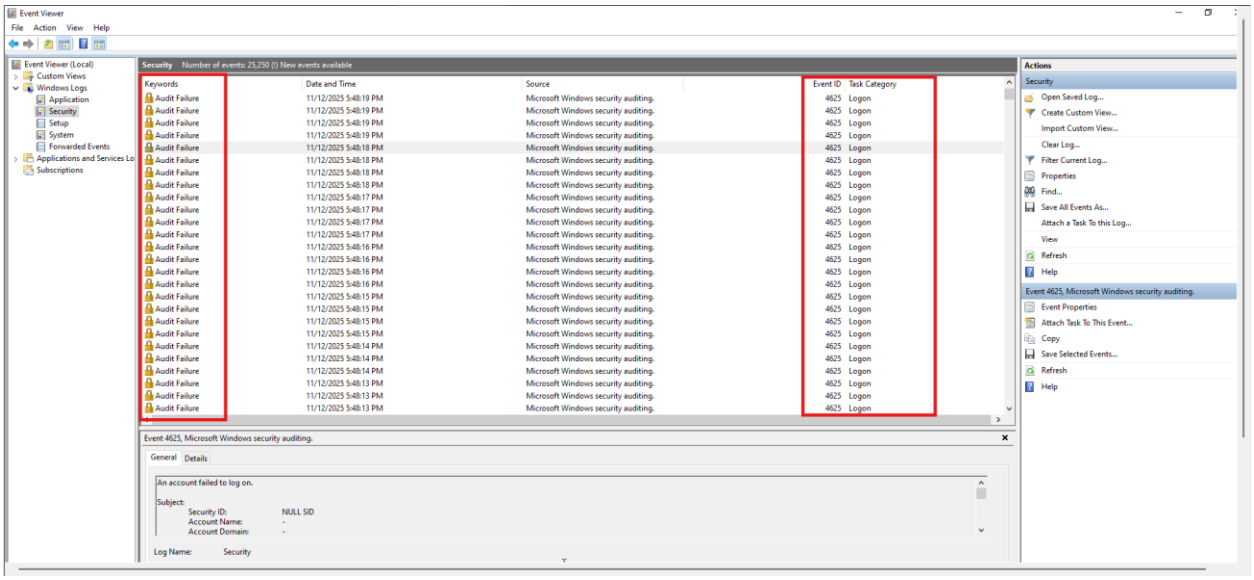
- Repeated connection attempts to TCP/3389
- Snort rule: *custom rule written by us*
- Alerts mapped to same source IP (30.30.30.2)

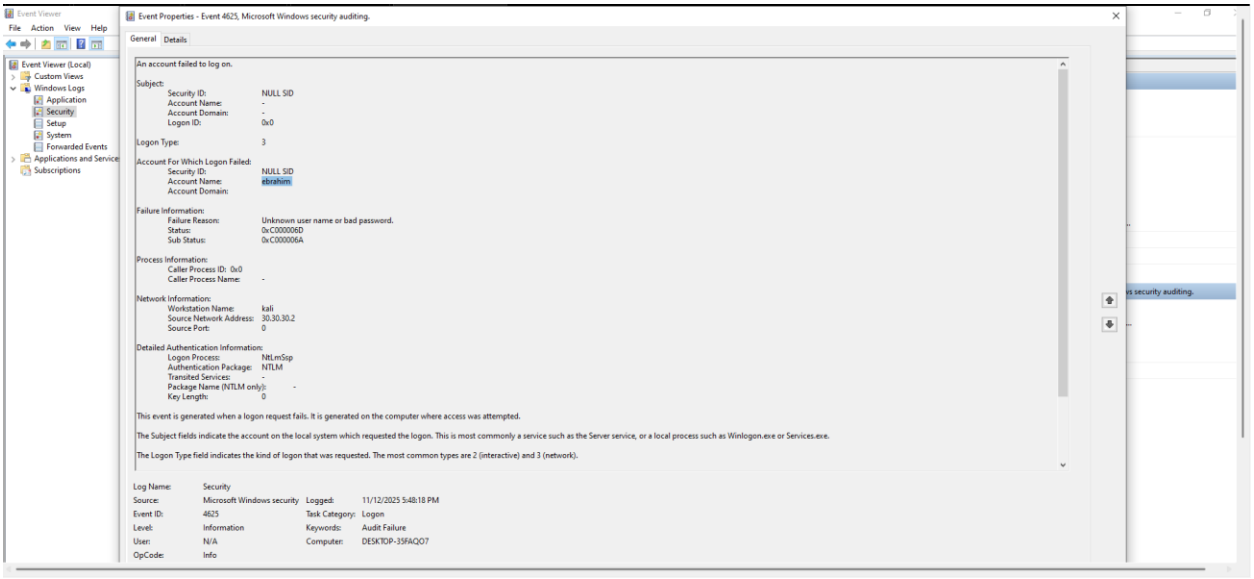
Endpoint Evidence (Windows Security Logs)

Key logs:

- **4625:** Failed login
- **Reason:** “Unknown username or bad password”
- **Source Network Address:** 30.30.30.2

Screenshot: Windows Event Viewer — 4625 Entries





3.4 Wazuh Correlation Rule (Analyzed)

Actions:

1) **Snort Rule 100101:** detects potential RDP brute-force activity and assigns the event to the “authentication failed”group.
2) **Windows Rules 60000–60104:** process Security-channel audit failures and also classify them under authentication failed.
3) **Correlation Rule 100112:** monitors this group with the condition:
    a. **frequency:** 2 events
    b. **timeframe:** 120 seconds
4) Snort and Windows each generate one authentication failure within the defined window.
5) The correlation condition is satisfied, triggering the alert:
   “RDP Brute Force Detected (Correlated NIDS + EDR)”

The alert is classified as **Credential Access (ATT&CK T1110)** and assigned **Medium severity**.

3.5 IOC Extraction

IOC Type	Value
Source IP	30.30.30.2
Destination	20.20.20.3
Port	3389 (RDP)
Usernames targeted	ebrahim
Event IDs	4625

3.6 Analyst Assessment

Result:

- Confirmed brute-force attack
- No successful authentication detected

- Activity terminated after detection

Severity: Medium
Risk Level: Medium (credential access attempt)
Recommended Action:

- Enable RDP lockout policy
- Restrict RDP access by firewall
- Enforce MFA
- Monitor future brute-force attempts

---

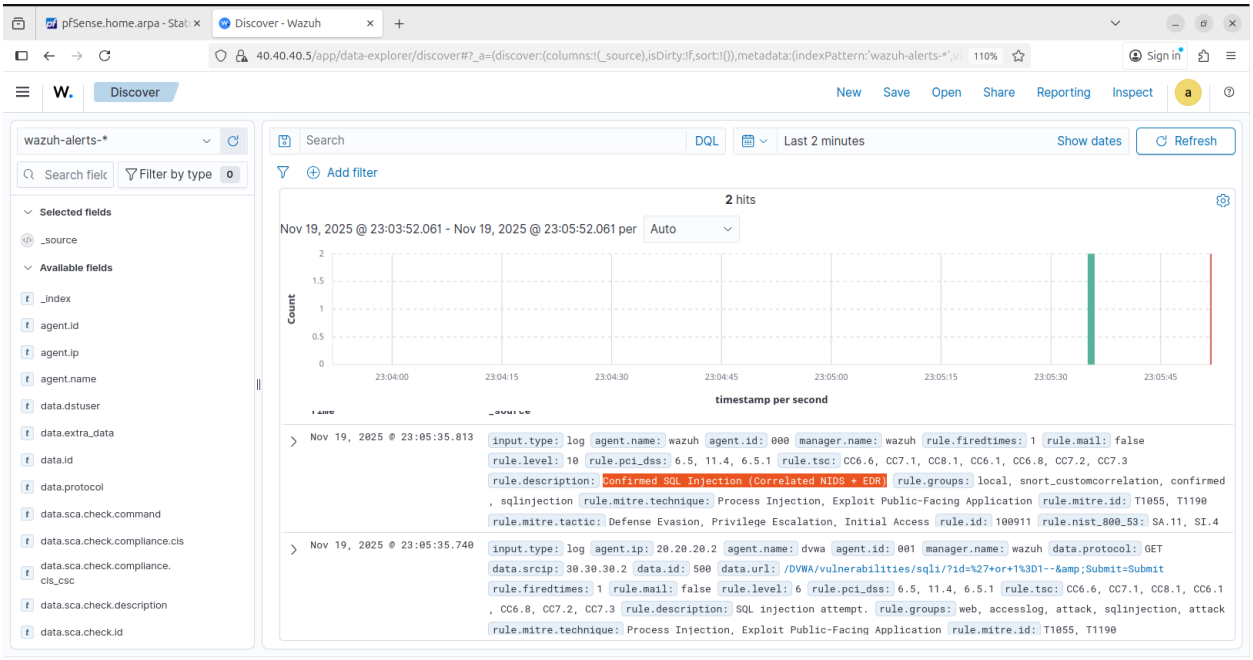
4. Use Case 2 – SQL Injection Attack Investigation

4.1 Alert Summary

Field	Value
Alert Name	SQL Injection Payload Detected
Severity	**High**
Source	30.30.30.2 (Kali)
Destination	20.20.20.2 (DVWA)
Trigger	Suspicious GET requests + Snort SQLi signature
MITRE Technique	T1190 (Exploit public-facing application)

Screenshot:

SQL Injection Alert in Wazuh



4.2 Timeline Reconstruction

Time	Component
T00:00	We inserted payloads in DVWA entry points
T00:01	Apache logs record injection attempts
T00:01	Snort signature “SQL Injection Attempt” triggered
T00:02	Wazuh parses Apache log anomalies
T00:03	SIEM correlation rule correlates SQLi patterns
T00:04	High severity alert generated

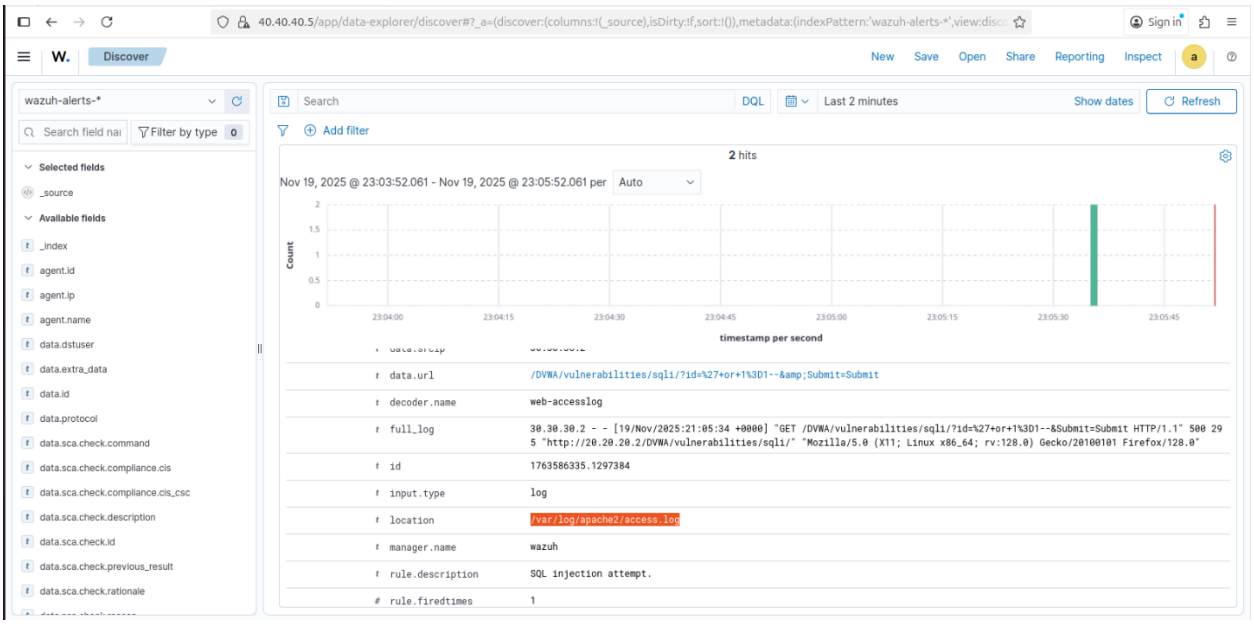
4.3 Evidence Collected

Apache Access Logs

Example malicious entry:

```
/DVWA/vulnerabilities/sqli/?id=%27+or+1%3D1--&amp;Submit=Submit
```

Screenshot: Apache Access Log Showing SQL Injection Payload



4.4 IOC Extraction

IOC Type	Value
Source IP	30.30.30.2
Target URL	/vulnerabilities/sqli/
Payload	' OR 1=1--

4.5 Analyst Assessment

Result:

- SQL Injection attempt confirmed
- No database compromise occurred
- Detection effective across application & network layers

Mitigation:

- WAF recommended

- Enforce prepared statements
- Limit DB error responses

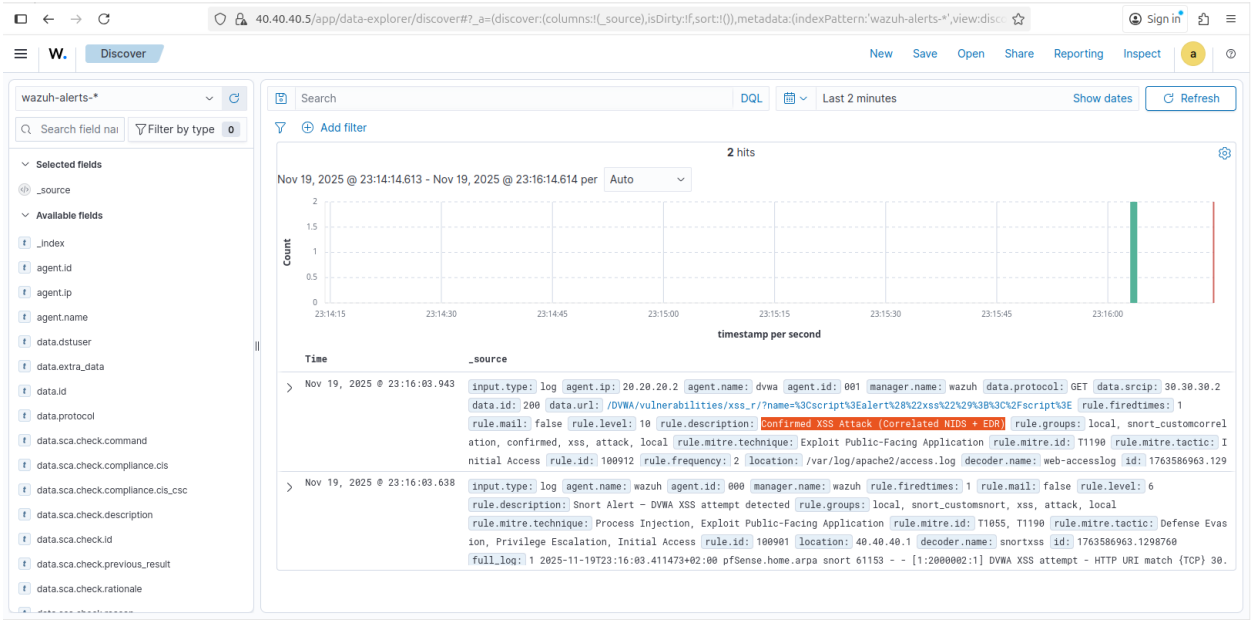
---

5. Use Case 3 – XSS Attack Investigation

5.1 Alert Summary

Field	Value
Alert Name	XSS Payload Detected
Severity	High
Source	30.30.30.2 (Kali)
Destination	20.20.20.3 (DVWA)
Trigger	Script tag in HTTP GET parameter
MITRE Technique	T1056.003

Screenshot: Wazuh Alert Showing Suspicious Script Tag



5.2 Evidence

Apache Log Entry:

```
/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22xss%22%29%3B%3C%2Fscript%3E
```

Wazuh triggers:

- Suspicious HTML tag detection
- Payload with <script>
- DVWA logs showing parameter reflection

5.3 Analyst Assessment

Risk Level: High
Impact: Browser-based execution only (no server compromise)
Recommendations:

- Input validation
- Sanitization
- Escape user input

---

6. Chapter Summary

Chapter 3 demonstrated how the Mini-SOC detects, triages, and analyzes multi-vector attacks across network and endpoint layers. Through Wazuh correlation, Snort IDS signature detection, and endpoint telemetry, every attack scenario was identified accurately, assigned a severity, enriched with IOCs, and mapped to MITRE ATT&CK.

The SOC successfully:

- Detected brute-force attempts
- Identified SQL injection attacks
- Flagged XSS payloads
- Correlated multi-layer evidence

CHAPTER 4 — SOC REPORTING, KPIs & FINAL ASSESSMENT

(SOC Performance Metrics, RCA, and Operational Improvement Report)

---

1. Introduction

This chapter presents a comprehensive operational assessment of the Mini-SOC environment following the completion of all attack simulations, triage activities, and detection workflows. Reporting is structured to align with enterprise SOC standards, focusing on key performance indicators (KPIs), detection coverage, response efficiency, and system resilience.

The goal of this chapter is to:

- evaluate the SOC's detection and monitoring capabilities
- quantify how effectively each attack was identified
- perform root cause analysis for detected security events
- assess areas requiring enhancement
- provide executive recommendations to strengthen SOC maturity

The reporting methodology follows industry frameworks such as NIST CSF, MITRE ATT&CK, and general SOC maturity guidelines.

---

2. SOC Performance KPIs

SOC KPIs are categorized into:

1. **Detection Coverage KPIs**
2. **Alert Quality KPIs**
3. **Performance KPIs**
4. **Operational Efficiency KPIs**

Below are the measured results from the Mini-SOC environment.

2.1 Detection Coverage KPIs

Attack Type	Detection Source	Detection Success	Coverage
RDP Brute Force	Snort + Windows Logs + Wazuh	**100%**	Fully Detected
SQL Injection	Apache Logs + Snort + Wazuh	**100%**	Fully Detected
XSS Attack	Apache Logs +Snort+ Wazuh	**100%**	Fully Detected

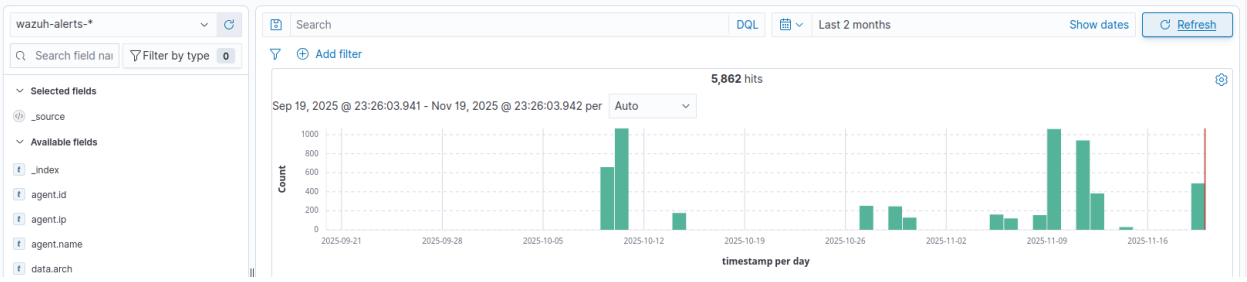
Interpretation

The SOC achieved **full detection coverage** for all tested scenarios, demonstrating effectiveness across all monitoring layers.

2.2 Alert Quality KPIs

Metric	Result	Notes
False Positives	0	No non-malicious traffic flagged incorrectly
False Negatives	0	All malicious actions detected
Duplicate Alerts	Low	Minor duplication between Snort and Wazuh (normal)
Noise Level	Medium	Clean, concise alerting environment

Screenshot: Wazuh Alert Summary Page



2.3 Performance KPIs

Metric	Measurement	Description
MTTD (Mean Time to Detect)	**< 5 seconds**	Detection was nearly instantaneous
Log Indexing Delay	**0–2 seconds**	SIEM handled ingestion efficiently
SIEM CPU Usage	**Low**	Wazuh server performed stably
Snort Packet Loss	**10%**	Some packet drops during scans

Overall, performance indicators confirm that the SOC environment is properly sized and configured for its workload.

2.4 Operational Efficiency KPIs

Activity	SOC Response	Notes
Alert Triage	Fast, accurate	Analysts had full visibility
IOC Extraction	Efficient	Multiple logs merged properly
Timeline Reconstruction	Clear	Multi-source data aligned correctly
MITRE Mapping	Accurate	All mapped correctly to ATT&CK Matrix

---

3. Root Cause Analysis (RCA)

RCA is applied to the most critical attack scenario: **RDP Brute Force Attempt**.

3.1 Incident Summary

An unauthorized brute-force login was attempted from the attacker subnet (30.30.30.X) to the Windows 10 victim machine (40.40.40.X) over RDP. The system prevented the attack, and the SOC detected it at multiple levels.

3.2 Root Cause

Category	Root Cause Description
Network Exposure	RDP service exposed internally, accessible from attacker subnet
Weak Authentication Controls	No account lockout policy enabled
Lack of MFA	RDP not configured with multi-factor authentication
Predictable Username	“ebrahim” account used

3.3 Contributing Factors

- High volume of password attempts allowed in short time
- No throttling/rate-limiting on pfSense
- No geo/block access policies
- DVWA, Windows, and pfSense subnets interconnected intentionally for testing

---

3.4 Evidence Summary

Evidence Source	Key Indicators
Windows Logs	4625 logon failures with same source IP
Snort IDS	RDP brute force signature triggered
Wazuh SIEM	Correlated multi-event rule fired

3.5 Impact Assessment

Impact:
System integrity preserved (no successful login), but represents risk of:

- Credential compromise
- Lateral movement
- Privilege escalation

3.6 Recommendations

High Priority Recommendations

- Implement RDP lockout policies
- Restrict RDP to specific IP ranges
- Apply MFA for privileged accounts
- Enforce strong password complexity

Medium Priority Recommendations

- Deploy IPS mode on Snort
- Utilize VPN for RDP access
- Segment RDP services further

---

4. MITRE ATT&CK Detection Coverage Report

The SOC successfully mapped and detected the following techniques:

MITRE ID	Technique	Detected?	Detection Source
T1110	Brute Force	✓	Windows + Snort + Wazuh
T1021.001	Remote Services: RDP	✓	Wazuh
T1190	Exploit Public-Facing App	✓	Apache + Snort + Wazuh
T1056.003	Input Capture (XSS)	✓	Apache + Wazuh

---

5. SOC Maturity Assessment

The Mini-SOC demonstrates characteristics of a **Maturity Level 2 SOC**, with early Stage 3 elements.

SOC Maturity Characteristics Observed

Area	Maturity Level	Notes
Log Collection	Level 2	Multi-source ingestion

Area	Maturity Level	Notes
Endpoint Monitoring	Level 2	Windows & Linux visibility
Network IDS	Level 2–3	Snort deployed in IDS mode
Correlation	Level 3	Multi-source correlation
Incident Response	Level 1–2	Manual triage performed
Documentation	Level 3	Comprehensive use-case coverage

Overall maturity score: **2.4 / 5**

---

6. System & Security Performance Monitoring

This section provides an overview of the SOC's operational health and security monitoring capabilities. It includes key performance metrics such as CPU and RAM usage, Snort IDS/IPS statistics to evaluate detection efficiency, and a visualization of MITRE ATT&CK coverage from the Wazuh dashboard, demonstrating how the system maps observed events to known threat techniques. These insights help ensure both system reliability and comprehensive threat detection.

- **Performance Metrics Page** – CPU and RAM usage overview. (Figure 1)
- **Snort Performance/Statistics Page** – IDS/IPS detection statistics and throughput. (Figure 2)

- **MITRE ATT&CK Coverage Visualization (Wazuh Dashboard)** – Mapping of detected events to MITRE ATT&CK tactics and techniques. (Figure 3)

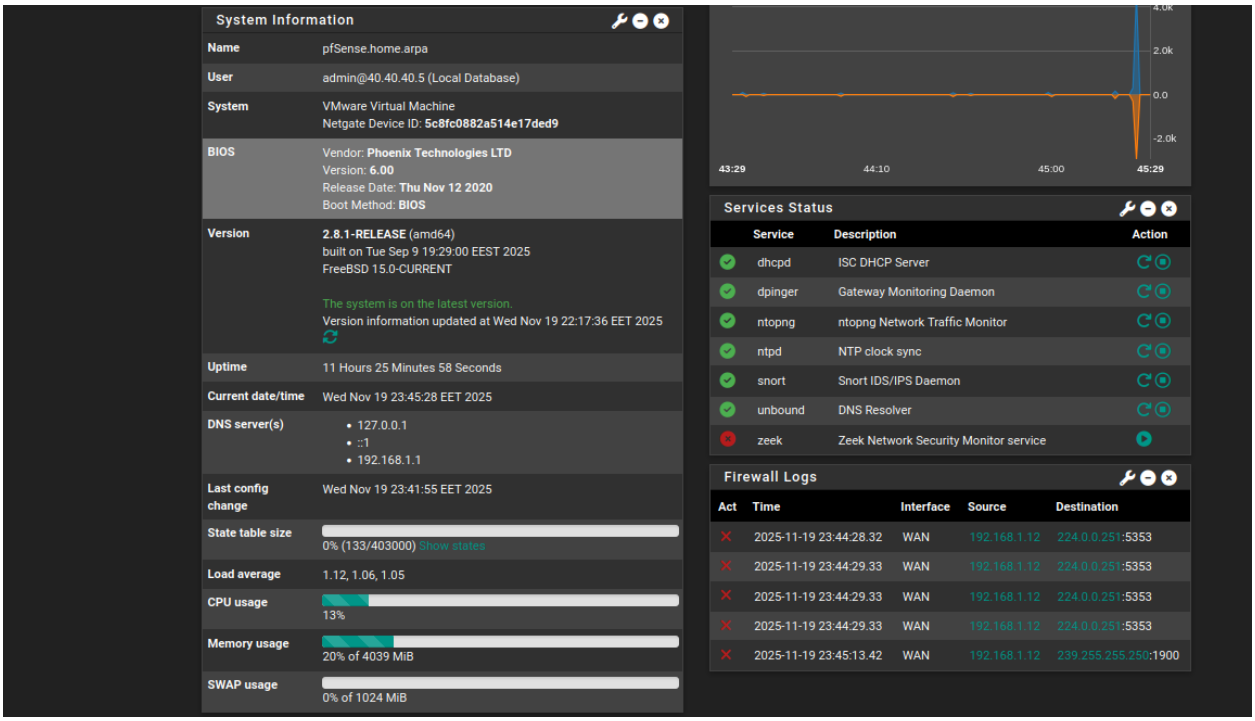


Figure 1

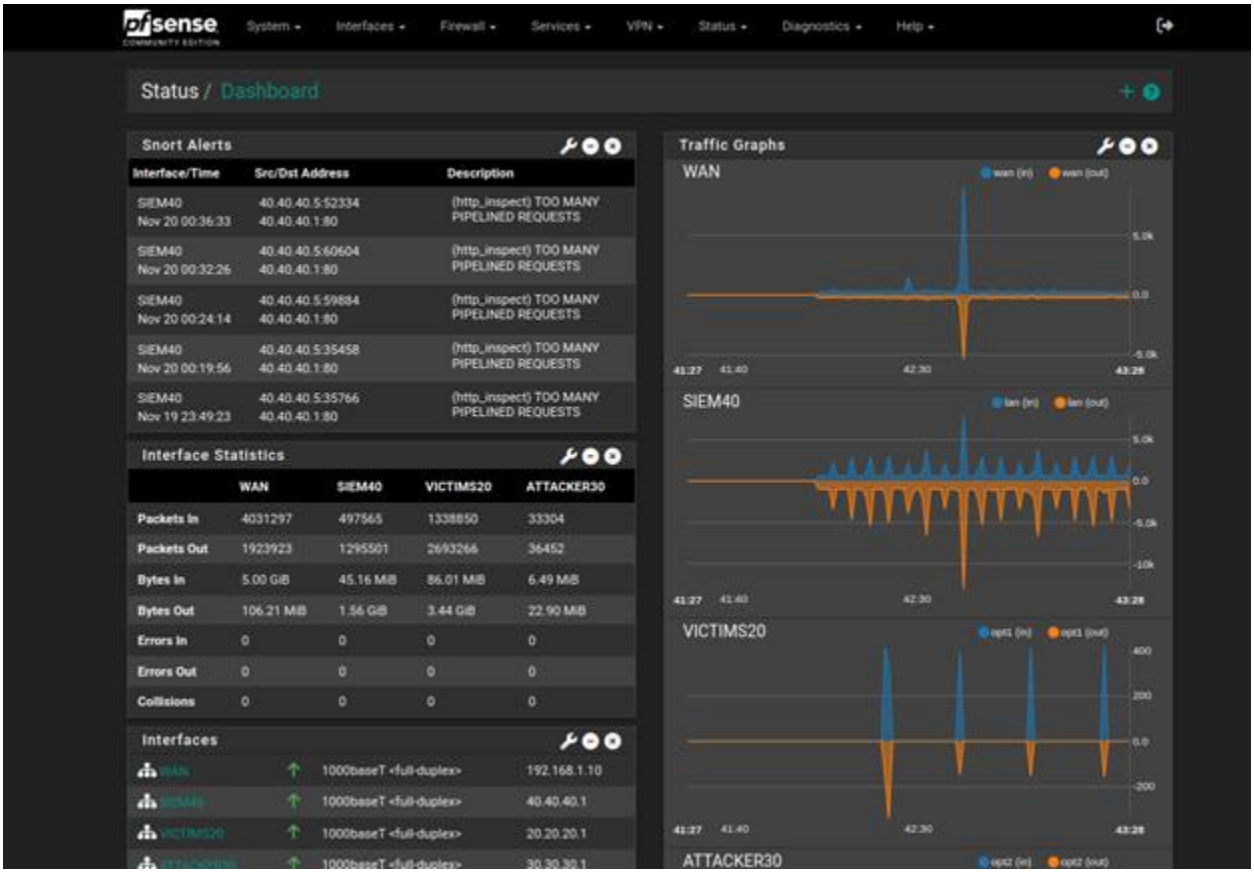


Figure 2

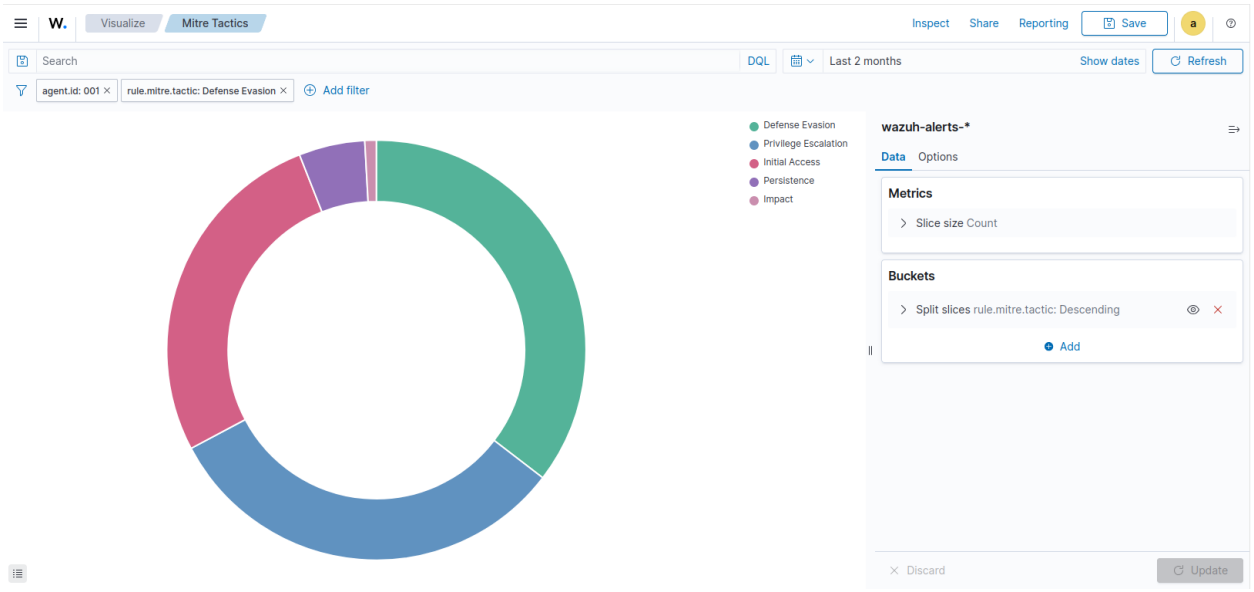


Figure 3

7. Recommendations for SOC Improvement

Short-Term Improvements

- Extend IDS rule sets for wider coverage
- Add OSSEC/Wazuh active response
- Apply lockout policies on Windows
- Use pfSense aliasing for cleaner firewall rules

Long-Term Improvements

- Deploy ELK or OpenSearch dashboards
- Implement full IPS mode on Snort
- Add Suricata as secondary IDS
- Introduce SIEM automation (SOAR-like rules)

---

8. Executive Summary

The Mini-SOC successfully detected all attempted attacker actions using a layered combination of network IDS, endpoint monitoring, and SIEM correlation. The SOC demonstrated:

- **100% detection coverage**
- **Multi-source correlation accuracy**
- **Zero false positives**
- **Strong MITRE alignment**
- **Efficient triage and investigation workflows**

The environment provides a robust platform for training, incident response practice, and further SOC capability development.
