**Femlalogy SOC Report**

**Project Title:** Threat Detection & Incident Response Using Wireshark, pfSense, and Wazuh  
**Organization:** Femlalogy  
**Analysts:** Pascal Brice NGONO

A computer screen with a lock

AI-generated content may be incorrect.

**Role:** Security Operations Center (SOC) Analyst

**Table of Contents**

1. Executive Summary …………………………………………………………………………3
2. Project Introduction …………………………………………………………………………4
3. Methodology …………………………………………………………………………………..5
4. Phase-by-Phase Analysis…………………………………………………………………..6
   * Phase 1: Wireshark – Network Traffic Capture & Analysis ……………..6
   * Phase 2: pfSense and Snort – Firewall & Policy Enforcement …………20
   * Phase 3: Wazuh – Security Event Monitoring & Response ……..………33
5. Final Findings & Impact …………………………………………………………………….36
6. Recommendations …………………………………………………………………………..38
7. Conclusion ……………………………………………………………………………………..39
8. References ……………………………………………………………………………………..40
9. **Executive Summary**

The purpose of this project was to strengthen Femlalogy’s cybersecurity posture by deploying and integrating three key tools: Wazuh, Wireshark, and pfSense. In this project, we simulated real-world attacks, detected network anomalies, enforced firewall rules, and responded to security incidents effectively and promptly.

* **Wireshark:** Used for Capturing and analyzing suspicious traffic (e.g., Nmap scans, ping sweeps).
* **pfSense (with Snort):** Used for Configuring and enforcing network/firewall policies, blocking ransomware-related IPs and malicious ICMP traffic.
* **Wazuh:** Used for Providing security event correlation and monitoring, resulting in actionable alerts and incident reports.

This project highlights the importance of proactive monitoring and layered defense in protecting organizational assets against evolving cyber threats.

**2. Project Introduction**

Cybersecurity is critical for organizational success in the digital age. This project aimed to build and assess a SOC for Femlalogy, designed to detect, monitor, and respond to cyber threats in real time.

* **Tools Used:** Wazuh, pfSense, Wireshark
* **Simulation Goal:** Test organizational readiness against cyber threats
* **Roles:** Wireshark for packet analysis, Wazuh for event correlation/reporting, pfSense as the network barrier

**3. Methodology**

* **Wireshark:** Captured suspicious traffic (Nmap scans, ping sweeps)
* **pfSense & Snort:** Blocked ICMP packets, provided intrusion detection
* **Wazuh:** Tracked logs, created alerts, generated executive reports
* **Lab Setup:** Kali Linux (attacker), Ubuntu (victim & Wazuh agent)

**4. Phase-by-Phase Analysis**

**Phase 1: Wireshark – Network Traffic Capture & Analysis**

*Goal: Capture and analyze suspicious traffic like Nmap scans and ping sweeps.*

**Launch Wireshark**

**-** Open Wireshark from your applications menu.

- Select the correct network interface:

   - Usually named `eth0`, `ens33`, or `enp0s3`

   - Look for one with active traffic (packets increasing)

Click **Start Capture** (blue shark fin icon).

*This is like turning on a surveillance camera for your network.*

**Apply Filters to Focus Your View**

*Wireshark captures everything, so use filters to zoom in.*

**Common Filters:**

`ip.addr == 192.168.1.xxx` : Show traffic to/from victim IP

`icmp`: Show ping packets (used in DoS attacks)

`dns` : Show DNS queries and responses

`http` : Show web browsing traffic

`tcp.port == 22` : Show SSH traffic

`tcp.flags.syn == 1 && tcp.flags.ack == 0` : Show Nmap SYN scans

*Filters are like turning the camera to focus only on suspicious activity.*

**Simulate Normal Traffic**

On Ubuntu (Victim):

Open Firefox and visit:

https://www.google.com

In Wireshark:

Apply filter:

ip.addr == 192.168.1.xxx and http

You’ll see:

- `GET /canonical.html HTTP/1.1`

- `HTTP/1.1 200 OK`

*This shows normal browsing behavior, useful for baseline comparison.*

**Simulate an Attack from Kali**

Simulate a Ping Flood (DoS Attack)

On Kali (Attacker):

Run:

ping -f 192.168.1.xxx

*This floods the victim with ICMP packets.*

In Wireshark:

Apply filter:

Icmp

You’ll see:

- Repeated ICMP Echo Requests from Kali to Ubuntu

- Possibly dropped packets if the victim is overwhelmed

*This simulates a denial-of-service attack*

**Simulate an Nmap Scan**

On Kali:

Run:

nmap -sS 192.168.1.xxx

ping -c 5 192.168.1.xxx

*This sends stealthy SYN packets to probe open ports.*

In Wireshark:

Apply filter:

tcp.flags.syn == 1 && tcp.flags.ack == 0

You’ll see:

- TCP SYN packets from Kali to Ubuntu

- Destination ports like 22 (SSH), 80 (HTTP), etc.

*This is like a burglar checking which doors are unlocked.*

**A screenshot of a computer

AI-generated content may be incorrect.**

*The screenshot above displays the normal traffic on wireshark on kali with no filters to show there was an internet connection.*

**Analyze DNS Traffic**

Filter:

ip.addr == 192.168.1.xxx and dns

You’ll see:

- DNS queries like `A www.google.com`

- Responses from DNS server (e.g., `192.168.1.1`)

*DNS is like asking for directions, every website visit starts with a DNS query.*

**A screenshot of a computer

AI-generated content may be incorrect.**

*Analyze dns traffic*

**A screenshot of a computer

AI-generated content may be incorrect.***The image above shows the filters used to display the traffic for when the victim IP 192.168.1.xxx visited google.com on mozilla firefox browser.*

Reminder, Ping flood was ran to simulate a Denial Of Service (DOS) attack on the victim machine, Using the “icmp” keyword filter in wireshark, it displayed only the ping which was launched from the attacker onto the victim IP.

*Simulate a Ping Flood (DoS Attack)*

*On Kali (Attacker):*

*Run:*

*ping -f 192.168.1.xxx*

*This floods the victim with ICMP packets.*

This is a simulated ping flood—a basic Denial of Service (DoS) tactic. The attacker sends rapid ICMP requests to overwhelm the victim.

In Wireshark, look for:

  - `ICMP` packets (ping)

  - `TCP SYN` packets (Nmap scan)

- Use filters:

  - `icmp`

  - `tcp.flags.syn == 1 && tcp.flags.ack == 0`

ICMP (Ping) Traffic Analysis

```

icmp

```

Press Enter.

**A screenshot of a computer

AI-generated content may be incorrect.**

What We’re Seeing:

- \*\*Echo Request\*\*: Sent from attacker to victim (e.g., Frame 485)

- \*\*Echo Reply\*\*: Sent from victim back to attacker (e.g., Frame 486)

- Each pair shows:

  - `seq=1/256`, `seq=2/512`, etc. (sequence number and payload size)

  - `ttl=64` (time to live)

  - `id=0x0001` (ICMP identifier)

Nmap was ran to scan from my kali using the -sS- to perform a stealth scan where Nmap sends SYN packets to start connections but never completes the TCP handshake and the -p- to scan all the ports.

**Nmap Stealth Scan Analysis**

Command Used:

nmap -sS -p- 192.168.1.xxx

- `-sS`: Stealth SYN scan

- `-p-`: Scan all 65,535 ports

**A screenshot of a computer program

AI-generated content may be incorrect.**

What It Does:

- Sends SYN packets to initiate a TCP handshake

- If the port is open, the victim replies with SYN-ACK

- Nmap then sends RST (reset) to avoid completing the handshake

This scan sends a SYN packet to each port. If the port is:

**Open** → the target replies with SYN-ACK

**Closed** → the target replies with RST-ACK

**Filtered** → no reply or ICMP unreachable

*This avoids detection by some firewalls and logging systems—making it a stealthy scan.*

Apply Filter in Wireshark:

To see SYN packets:

tcp.flags.syn == 1 && tcp.flags.ack == 0

To see SYN-ACK responses:

tcp.flags.syn == 1 && tcp.flags.ack == 1

To see RST packets:

tcp.flags.reset == 1

What You’ll See:

- SYN packets from `192.168.1.xxx` to `192.168.1.xxx`

- One open port detected: `22/tcp open ssh`

- All other ports responded with RST (closed)

*This is confirmation that SSH is open and reachable—this is a common target for brute-force attacks.*

This is the result of the nmap on wireshark:

A screenshot of a computer

AI-generated content may be incorrect.

**Open** → the target replies with SYN-ACK

*The filter “ip.src == 192.168.1.xxx and ip.dst == 192.168.1.xxx and tcp.flags.syn and tcp.flags.ack” displays the TCP handshake between the victim machine (ip.src) and the attacker machine (ip.dst). The red portrays the ports which are closed on the victim machine.*

**What Wireshark Captured**

**RST, ACK Responses`-sS`: SYN scan (stealth scan)**

- `-p-`: Scan all 65,535 TCP ports

This scan sends a SYN packet to each port. If the port is:

**Open** → the target replies with SYN-ACK

**Closed** → the target replies with RST-ACK

**Filtered** → no reply or ICMP unreachable

We saw many lines like:

```

[RST, ACK] Seq=1 Ack=1 Win=0 Len=0

```

These indicate:

- The port is **closed**

- The target (192.168.1.xxx) is rejecting the connection attempt

**SYN, ACK Response**

You saw one line like:

60 22 - 38840 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

This means:

- Port **22 (SSH)** is **\*\*open**

- The target responded with SYN-ACK, indicating it’s ready to establish a connection

*This is exactly what Nmap is looking for. It confirms that SSH is running and reachable.*

And these where there results from the nmap of the only open port, where we filtered the nmap scan using “ip.src == and tcp.flags.syn == 1 and tcp.flags.ack == 1”. The “tcps.flags.syn==1” means to give the packet where there was a connection to the three way handshake:

**To see SYN packets sent by Nmap:**

```wireshark

tcp.flags.syn == 1 && tcp.flags.ack == 0

```

**To see SYN-ACK responses (open ports):**

```wireshark

tcp.flags.syn == 1 && tcp.flags.ack == 1

```

**To see RST-ACK responses (closed ports):**

```wireshark

tcp.flags.reset == 1 && tcp.flags.ack == 1

A screenshot of a computer program

AI-generated content may be incorrect.

During the reconnaissance phase, an Nmap stealth scan (`nmap -sS -p-`) was launched from the attacker machine (192.168.1.xxx) targeting the victim machine (192.168.1.xxx). The scan was designed to probe all 65,535 TCP ports without completing the TCP handshake, making it stealthier than a full connect scan.

In Wireshark, the scan was captured and analyzed using the following filters:

`tcp.flags.syn == 1 && tcp.flags.ack == 0` to view SYN probes

`tcp.flags.syn == 1 && tcp.flags.ack == 1` to identify open ports

`tcp.flags.reset == 1 && tcp.flags.ack == 1` to identify closed ports

The results showed that:

**Port 22 (SSH)** responded with a **SYN-ACK**, indicating it is **open**

All other scanned ports responded with **RST-ACK**, confirming they are **closed**

*This behavior is consistent with a typical SYN scan, where the attacker sends SYN packets and interprets the type of response to determine port status. The scan was completed in approximately 174 seconds, and the only open service detected was SSH.*

***Bonus: hping3 Command***

We can also run:

```bash

sudo hping3 -c 100 -S -p 80 192.168.1.xxx

```

Breakdown

- `-c 100`: Send 100 packets

- `-S`: Set SYN flag (simulate connection request)

- `-p 80`: Target port 80 (HTTP)

- `192.168.1.XXX`: Victim IP

A screenshot of a computer

AI-generated content may be incorrect.

*This simulates an attacker repeatedly trying to initiate TCP handshakes to port 80, which is commonly used for web servers.*

What It Does:

- Sends 100 SYN packets (`-S`) to port 80 on the victim

- Simulates a SYN flood or stealth scan

- Useful for testing firewall and IDS/IPS detection

How to Analyze in Wireshark:

Use this filter:

ip.src == 192.168.1.xxx and tcp.dstport == 80 and tcp.flags.syn == 1

*This helps you verify that the SYN packets were sent and whether the victim responded with RST or SYN-ACK.*

A SYN flood was simulated using targeting HTTP port 80 to detect anomalous HTTP request behavior. The attack generated 100 SYN packets, which were captured and analyzed in Wireshark. To filter the traffic, the expression was applied, isolating relevant handshake attempts. The victim responded with RST packets, confirming that port 80 was closed and not accepting connections.

A screen shot of a computer

AI-generated content may be incorrect.

***Notes:*** *During reconnaissance, the attacker (192.168.1.xxx) repeatedly attempted TCP three-way handshakes to the victim machine (192.168.1.xxx) on port 80. Each attempt was met with a* ***RST*** *response, indicating that no service was listening on that port. This behavior aligned with Nmap scan results, which confirmed port 80 as closed.*

*To further validate this, a TCP SYN flood was simulated using hping3, sending 100 SYN packets to port 80 of the victim. The command used was:*

*sudo hping3 -c 100 -S -p 80 192.168.1.xxx*

*Wireshark was employed to capture and analyze the traffic. The filter tcp.flags.syn and tcp.flags.ack was applied to isolate SYN-ACK responses. However, the victim consistently replied with* ***RST, ACK*** *packets, reinforcing that port 80 was closed. This confirms the Nmap findings and illustrates how SYN floods can be used to probe for open ports—where the absence of SYN-ACKs and presence of RST-ACKs serve as clear indicators of a closed service.*

**Phase 2: pfSense and Snort – Firewall & Policy Enforcement**

Goal: Block malicious traffic and detect intrusions using Snort.

Objective:

Configure pfSense to:

- Block known ransomware IPs (Medusa, Lockbit)

- Reject ICMP traffic (e.g., ping floods)

- Monitor and enforce firewall rules

- Integrate Snort for intrusion detection

**Access pfSense Web Interface**

- Open browser

- Go to: `https://192.168.1.x` (or your pfSense LAN IP)

- Login:

  - Username: xxxxx

  - Password: xxxxxx

**Configure Interfaces**

- Assign LAN and WAN interfaces

- Set LAN IP: `192.168.1.xxx`

**3. Install Snort**

- Go to **System > Package Manager > Available Packages**

- Search for **Snort**

- Click **Install**

**Configure Snort**

- Go to **Services > Snort**

- Add a new interface (LAN)

- Enable Snort, select rules (e.g., Emerging Threats)

- Enable **Block Offenders**

**Create Firewall Rules**

- Go to **Firewall > Rules > LAN**

- Add rules to:

  - Block ICMP (ping)

  - Block known malicious IPs

**Verify:**

- Try pinging Ubuntu from Kali again

- Check **Status > System Logs > Firewall** for blocked attempts

**- Create Aliases for Ransomware IPs**

Aliases let you group IPs for easier rule management.

1. Go to **Firewall > Aliases**

2. Click **Add**

3. Name: `Medusa\_Ransomware`

4. Type: **Host(s)**

5. Add known Medusa IPs (e.g., `185.220.101.1`, `185.220.101.2`)

6. Save and Apply

Repeat for:

- `Lockbit\_Ransomware`

- Add known Lockbit IPs

**-Block Ransomware IPs on Port 80**

1. Go to Firewall > Rules > WAN

2. Click Add

3. Action: Block

4. Interface: WAN

5. Protocol: TCP

6. Source: `Medusa\_Ransomware` alias

7. Destination: WAN address

8. Destination Port: `80` (HTTP)

9. Description: `Block Medusa ransomware IPs on port 80`

10. Save and Apply

Repeat for `Lockbit\_Ransomware`.

This blocks web-based ransomware communication attempts.

**-Reject ICMP Traffic**

ICMP (ping) can be used for reconnaissance or DoS attacks.

1. Go to **Firewall > Rules > WAN**

2. Click **Add**

3. Action: **Reject**

4. Protocol: **ICMP**

5. Source: **any**

6. Destination: **any**

7. Description: `Reject ICMP protocol`

8. Save and Apply

This stops ping floods and network probing.

**A screenshot of a computer

AI-generated content may be incorrect.**

The above screenshot displays the firewall rules which block popular ransomware groups which are Medusa and Lockbit on port 80. It also displays the rule that blocks ICMP packets from happening in the network.

**-Enable Snort for IDS/IPS**

What to Do:

1. Go to **System > Package Manager > Available Packages**

2. Search for **Snort**

3. Click **Install**

**Configure Snort:**

1. Go to **Services > Snort**

2. Add Interface: **WAN**

3. Enable Snort

4. Choose rule sets:

   - **Emerging Threats**

   - **Snort GPLv2 Community Rules**

5. Enable **Block Offenders**

6. Save and Apply

*Snort watches for suspicious patterns and blocks them automatically.*

Use pfSense’s alias feature to group and block known Lockbit ransomware IPs on port 80 (HTTP), preventing potential command-and-control communication or payload delivery.

**Created an Alias**

- **Name**: `Lockbit\_ransomware`

- **Type**: Host(s)

- **Description**: Blocks Lockbit ransomware IP

- **Entries**:

  - `3.64.4.198`

  - `3.67.161.133`

  - `185.215.113.66`

  - `13.107.4.52`

  - `92.51.2.22`

Aliases simplify rule management by letting you group multiple IPs under one label.

**Firewall Rule Applied**

Rule Details:

- **Action**: Block

- **Interface**: WAN

- **Protocol**: TCP

- **Source**: `Lockbit\_ransomware` alias

- **Destination**: WAN address

- **Destination Port**: `80` (HTTP)

- **Description**: Block Lockbit ransomware IPs on port 80

This rule prevents any HTTP traffic from these IPs from reaching your network—cutting off potential ransomware delivery or callback attempts.

pfSense was configured to block known ransomware IPs associated with the Lockbit group. An alias named `Lockbit\_ransomware` was created to group multiple malicious IP addresses, including:

`3.64.4.198`

`3.67.161.133`

`185.215.113.66`

`13.107.4.52`

`92.51.2.22`

These are the lists IP addresses from Lockbit ransomware group blocked by the firewall.

A screenshot of a computer

AI-generated content may be incorrect.

*A firewall rule was applied to block TCP traffic from these IPs to the WAN interface on port 80 (HTTP). This prevents potential command-and-control communication or ransomware payload delivery via web protocols. The use of aliases allows for scalable and efficient rule management, especially when dealing with multiple threat sources.*

**-IP Aliases for Ransomware Groups**

We created three aliases:

Alias Name, IPs Included, Threat Group(Lockbit, Medusa, Phobos)

`Lockbit\_ransomware` :`3.64.4.198`, `3.67.161.133`, `185.215.113.66`, `13.107.4.52`, `92.51.2.22` (Lockbit)

`Medusa\_Ransomeware`: `18.158.58.205`, `18.197.239.109` (Medusa)

`Phobos\_ransomware`: `194.165.16.4`, `45.9.74.14`, `147.78.47.224`, `185.202.0.111` (Phobos)

These aliases group known threat IPs for easy rule management and rapid updates**.**

**A screenshot of a computer

AI-generated content may be incorrect.**

*The above screenshot shows the list of IP addresses blocked by the firewall and grouped by the threats.*

**Firewall Rules, IP Aliases, and GeoIP Blocking in pfSense**

Objective:

Use pfSense to:

- Block known ransomware IPs (Lockbit, Medusa, Phobos)

- Reject risky protocols (ICMP, SSH)

- Restrict inbound traffic from high-risk geographic regions using GeoIP filtering

**Firewall Rules Applied**

**Rule - Protocol - Source - Destination - Port - Action - Description**

Block Lockbit - TCP - `Lockbit\_ransomware` - WAN address - 80 (HTTP) - Block - Prevents web-based C2

Block Medusa - TCP - `Medusa\_Ransomeware` - WAN address - 80 (HTTP) - Block - Blocks Medusa IPs

Block Phobos - TCP - `Phobos\_ransomware` - WAN address | 80 (HTTP) - Block | Blocks Phobos IPs

Reject ICMP - ICMP - any - any - — - Reject - Stops ping floods and reconnaissance

Block SSH - TCP - any - WAN address - 22 (SSH) - Block - Prevents remote login attempts

These rules reduce exposure to known threats and prevent unauthorized access.

**A screenshot of a computer

AI-generated content may be incorrect.**

*The screenshot above shows the updated firewall rules where SSH is blocked on the WAN.*

Below is the firewall rule GeoIP blocking that restricts access from high-risk countries. Blocking most continent’s inbound traffic heavily reduces the attack surface and ensures all legitimate activity comes from inside your lab.

GeoIP Blocking:

Let’s implement GeoIP filtering to:

- Block inbound traffic from high-risk countries

- Limit access to only trusted geographic regions (e.g., your lab environment)

This configuration blocks unsolicited inbound traffic from entire continents, significantly reducing your attack surface. Only trusted regions (e.g., North America) are allowed, which is ideal for a lab or internal network.

**A screenshot of a computer

AI-generated content may be incorrect.**

*This dramatically reduces the attack surface by excluding entire regions known for malicious activity.*

*Notes: pfSense was configured to enforce advanced firewall policies targeting known ransomware threats and high-risk traffic sources. IP aliases were created for Lockbit, Medusa, and Phobos ransomware groups, each containing verified malicious IP addresses. These aliases were used to block HTTP traffic on port 80, preventing potential command-and-control communication.*

*Additional rules were applied to reject ICMP traffic, mitigating ping flood and reconnaissance attempts, and to block SSH access on the WAN interface, reducing the risk of brute-force login attacks. ICMP traffic was rejected to mitigate ping flood attacks and reconnaissance attempts. Snort was installed and configured with community rule sets to detect and block intrusion attempts. The firewall state table confirmed that the rules were actively enforced and effective.*

*GeoIP filtering was also implemented to restrict inbound traffic from high-risk countries. This ensures that only legitimate activity from trusted regions can reach the lab environment, significantly reducing the overall attack surface.*

We’ve now built a firewall strategy that’s proactive and scalable. We’re not just reacting to threats, we’re anticipating and neutralizing them before they reach your network.

**Testing**

Here is the test for the blocked ICMP firewall rule in the network. The ip pinged was the host network and thanks to the rule the pings were dropped, proving the effectiveness of the rule.

We run:

ping -c4 192.168.1.xxx

**A screenshot of a computer program

AI-generated content may be incorrect.**

*What we observed: From 192.168.1.xxx icmp\_seq=1 Destination Host Unreachable4 packets transmitted, 0 received, +4 errors, 100% packet loss*

This confirms that our ICMP reject rule is working as intended. The firewall is actively dropping ping requests, preventing reconnaissance and DoS attempts via ICMP.

pfBlockerNG was configured to enforce GeoIP-based filtering. Inbound traffic from high-risk regions—including Africa, Asia, Europe, Oceania, and South America—was denied. This significantly reduced the attack surface by limiting exposure to only trusted geographic zones.

Additionally, the ICMP protocol was explicitly rejected via a firewall rule. This was tested using the `ping` command from the attacker machine (192.168.1.xxx) to the victim (192.168.1.xxx). All ping attempts failed with 100% packet loss, confirming that the rule was effective in blocking ICMP traffic.

These measures demonstrate a layered defense strategy that combines geographic filtering with protocol-level enforcement to protect the network from both broad and targeted threats.

We’ve now completed Phase 2, implementing IP-based, protocol-based, and region-based defenses. We’ve tested your rules, confirmed their effectiveness.

**Snort Intrusion Detection System**

**-Enable Snort for IDS/IPS**

1. Go to **System > Package Manager > Available Packages**

2. Search for **Snort**

3. Click **Install**

**- Configure Snort:**

1. Go to **Services > Snort**

2. Add Interface: **WAN**

3. Enable Snort

4. Choose rule sets:

   - **Emerging Threats**

   - **Snort GPLv2 Community Rules**

5. Enable **Block Offenders**

6. Save and Apply

*Snort watches for suspicious patterns and blocks them automatically.*

Below’s screenshot shows the IDS rule which detects offenders and raises alerts in pfsense when there is an nmap or a ping flood on the victim.

**A screenshot of a computer

AI-generated content may be incorrect.**

Let’s move into **Phase 3: Wazuh – Security Event Monitoring & Response**, where we’ll correlate these threats and generate actionable alerts. Let’s keep going.

**Phase 3: Wazuh – Security Event Monitoring & Response**

* **Goal:** Correlate logs and respond to security incidents. Correlate logs from pfSense, Snort, and endpoints agents using Wazuh to detect, investigate, and respond to security incidents in real time.
* **Key Actions:** Configured log forwarding from pfSense and endpoints; detected privilege escalation and suspicious user behavior
* **Findings:** Multiple alerts correlated with anomalies identified in Wireshark

**-Configure Log Forwarding from pfSense to Wazuh**

**On pfSense:**

1. Go to **Status > System Logs > Settings**

2. Enable **Remote Logging**

3. Set Wazuh server IP (e.g., `192.168.1.xxx`)

4. Choose log types:

   - Firewall

   - Snort alerts

   - System logs

5. Set protocol: **UDP**

6. Port: `514`

**On Wazuh Manager:**

1. Confirm `rsyslog` is listening

2. Check `/var/ossec/logs/ossec.log` for incoming logs

This allows Wazuh to ingest logs from pfSense and correlate them with endpoint activity.

-**Confirm Wazuh Agent Is Running on Ubuntu**

```bash

sudo systemctl status wazuh-agent

```

You should see:

```

Active: active (running)

```

*This agent monitors system logs, user activity, and file integrity on the victim machine*.

**Simulate Attacks Again**

From Kali:

- Run Nmap scan:

  ```bash

  nmap -sS -p- 192.168.1.xxx

  ```

- Run ping flood:

  ```bash

  ping -f 192.168.1.xxx

  ```

- Run Medusa brute-force:

  ```bash

  medusa -h 192.168.1.xxx -u root -P /usr/share/wordlists/rockyou.txt -M ssh

  ```

*These actions trigger alerts across Wireshark, Snort, and Wazuh.*

**Investigate in Wazuh Dashboard**

**Open Wazuh Web UI**

- Go to: `https://192.168.1.xxx`

- Login:

**Navigate to:**

- **Security Events > Event Viewer**

- Filter by:

  - Agent: Ubuntu

  - Rule Level ≥ 8

  - Tactic: Credential Access, Reconnaissance

**-Review Alerts:**

- SSH brute-force attempts

- ICMP flood detection

- Nmap scan detection

- Privilege escalation (e.g., sudo misuse)

Each alert includes:

- Timestamp

- Source IP

- Rule ID

- MITRE ATT&CK mapping

- Severity level

**Snort Alert Integration**

Although Snort’s **blocking mode is disabled**, it still detects and logs:

- Nmap scans

- Ping floods

- Suspicious TCP flags

**These alerts are forwarded to Wazuh and appear as:**

- `Snort: NMAP TCP Scan`

- `Snort: ICMP Flood Detected`

*Even without blocking, Snort acts as a sensor feeding Wazuh with valuable threat intelligence.*

*Notes: In Phase 3, Wazuh was used to correlate logs and respond to security incidents across the Femlalogy SOC environment. Log forwarding was configured from pfSense and endpoint agents to the Wazuh manager. Simulated attacks, including Nmap scans, ping floods, and brute-force attempts—were detected and logged.*

*Wazuh correlated multiple alerts with anomalies previously identified in Wireshark and Snort. These included:*

*- SSH brute-force attempts from Kali*

*- ICMP flood targeting the victim*

*- Privilege escalation via sudo misuse*

*-Each alert was mapped to MITRE ATT&CK tactics such as Credential Access and Reconnaissance. The dashboard provided real-time visibility into the attack timeline and allowed for rapid investigation and response.*

**5. Final Findings & Impact**

The engagement confirmed that Femlalogy was susceptible to multiple attempts by an attacker to gain access to a victim machine within the network. Simulated attacks—including Nmap scans, ping floods, and brute-force login attempts—were successfully launched from the attacker machine (Kali) and captured using Wireshark, pfSense, and Wazuh.

Key findings include:

- **Open SSH Port (22)** on the victim machine was identified via Nmap, indicating potential exposure to brute-force attacks.

- **ICMP traffic** was initially allowed, enabling ping flood attacks until firewall rules were enforced.

- **Snort alerts** confirmed detection of reconnaissance activity and anomalous traffic patterns.

- **Wazuh SIEM** correlated multiple events, including privilege escalation attempts and suspicious user behavior, providing actionable insights.

- **GeoIP filtering** and ransomware IP blocking significantly reduced the attack surface and prevented inbound traffic from known malicious sources.

These findings demonstrate the importance of layered defense, proactive monitoring, and real-time alerting in securing organizational assets.

**6. Recommendations**

Based on the findings, the following recommendations are proposed:

1. **Enforce Least Privilege Access**

   Limit SSH access to trusted IPs and implement key-based authentication to reduce brute-force risk.

2. **Maintain Updated Blocklists**

   Regularly update pfSense aliases for ransomware IPs and integrate threat intelligence feeds.

3. **Enable Snort Blocking Mode**

   Activate Snort’s inline blocking to automatically drop malicious packets upon detection.

4. **Expand GeoIP Filtering**

   Continue refining GeoIP rules to block high-risk regions while allowing trusted zones.

5. **Implement Centralized Logging**

   Ensure all endpoints and network devices forward logs to Wazuh for unified visibility.

6. **Conduct Regular Simulations**

   Perform periodic red team exercises to test detection and response capabilities.

7. **Educate Staff on Threat Awareness**

   Provide training on phishing, password hygiene, and incident reporting protocols.

**7. Conclusion**

This SOC engagement demonstrated how Femlalogy can detect, analyze, and respond to cyber threats using an integrated stack of open-source tools. Wireshark provided deep packet inspection, pfSense enforced firewall and IDS policies, and Wazuh delivered real-time event correlation and alerting.

The project highlights the value of proactive defense, layered security, and continuous monitoring. With the recommended improvements, Femlalogy is well-positioned to strengthen its cybersecurity posture and respond effectively to evolving threats.

**8. References**

* Wireshark Documentation (https://www.wireshark.org/docs/)
* pfSense and IDS/IPS Configuration (<https://docs.netgate.com/pfsense/en/latest/>)

<https://docs.netgate.com/pfsense/en/latest/packages/snort/index.html>

Snort IDS: (https://www.snort.org/)

* Wazuh Official Guide

https://documentation.wazuh.com/current/installation-guide/wazuh-agent/index.ht ml

(https://documentation.wazuh.com/)

* MITRE ATT&CK Framework (<https://attack.mitre.org/>)
* Nmap Network Scanner (<https://nmap.org/>)
* hping3 Packet Generator (http://www.hping.org/)
* Logs and dashboards from the lab environment
* Raw logs, alert data, and full packet captures