



NCY-II

Cyber Security Department

Fall 2024

Assignment_3

Names: Umar Niazi & Eman Zafar

Roll Numbers: 22i-1551 & 22i-1567

Snort Installation and Verification

1.1 Installation Steps:

- Backing Up the sources.list File:

```
mv /etc/apt/sources.list /etc/apt/sources.list.bak
```

- This command creates a backup of the original sources.list file to restore later if needed.

- Removing Old Update Files:

```
find /var/lib/apt/lists -type f -exec rm {} \;
```

- This command removes old package lists to avoid conflicts when updating.

- Modifying the sources.list to Include Ubuntu Repositories for Snort:

```
nano /etc/apt/sources.list
```

```
deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports focal main restricted  
universe multiverse
```

```
deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports focal-updates main  
restricted universe multiverse
```

```
deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports focal-security main  
restricted universe multiverse
```

```
deb [arch=i386,amd64] http://us.archive.ubuntu.com/ubuntu/ focal main restricted  
universe multiverse
```

```
deb [arch=i386,amd64] http://us.archive.ubuntu.com/ubuntu/ focal-updates main  
restricted universe multiverse
```

```
deb [arch=i386,amd64] http://security.ubuntu.com/ubuntu focal-security main  
restricted universe multiverse
```

- **Adding Required Public Keys:**

```
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 3B4FE6ACC0B21F32
```

```
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 871920D1991BC93C
```

- These commands add necessary public keys for package verification from the specified repositories.

- **Updating the System Package List:**

```
sudo apt update
```

- **Installing Snort**

```
sudo apt install snort
```

1.2 Verification:

- **Checking Snort Version:**

```
snort -V
```

- **Modifying snort.conf to Set HOME_NET:**

- Edit the configuration file to define the network IP range for Snort monitoring.

```
nano /etc/snort/snort.conf
```

- Modify the line to your device IP or network :

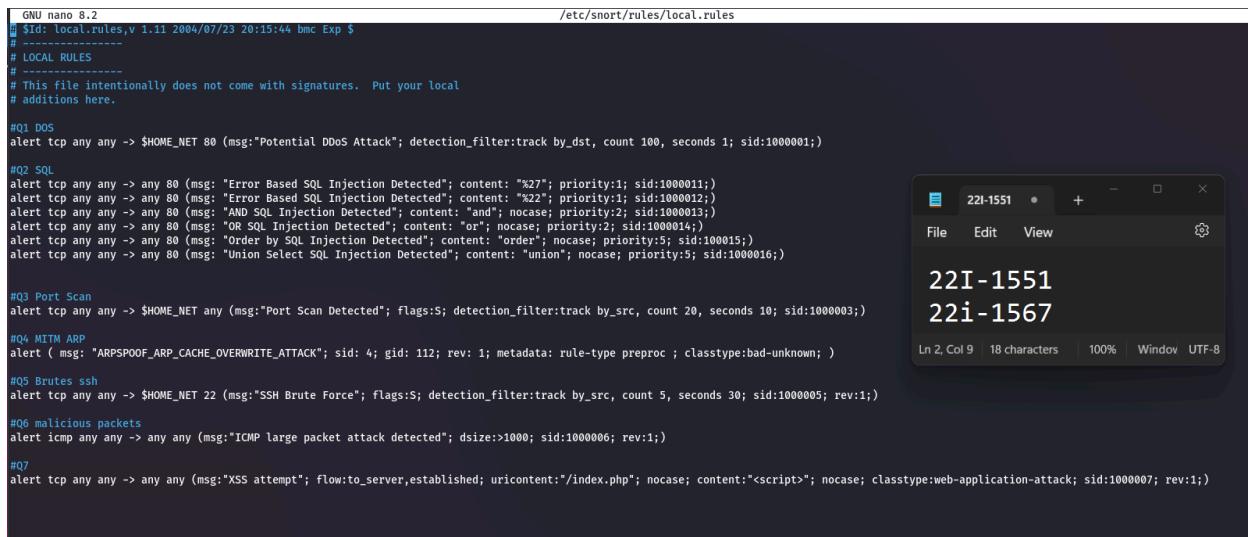
```
IPVAR HOME_NET 192.168.20.16/24
```

- **Running Snort in IDS Mode:**

```
snort -A console -q -c /etc/snort/snort.conf -i eth0
```

- This command runs Snort in IDS mode, using the specified configuration file and network interface.

2. Custom Snort Rules



```
GNU nano 8.2                               /etc/snort/rules/local.rules
$Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures. Put your local
# additions here.

#Q1 DOS
alert tcp any any -> $HOME_NET 80 (msg:"Potential DDoS Attack"; detection_filter:track by_dst, count 100, seconds 1; sid:1000001;)

#Q2 SQL
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%27"; priority:1; sid:1000011)
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%22"; priority:1; sid:1000012)
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected"; content: "and"; nocase; priority:2; sid:1000013)
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected"; content: "or"; nocase; priority:2; sid:1000014)
alert tcp any any -> any 80 (msg: "Order by SQL Injection Detected"; content: "order"; nocase; priority:5; sid:1000015)
alert tcp any any -> any 80 (msg: "Union Select SQL Injection Detected"; content: "union"; nocase; priority:5; sid:1000016)

#Q3 Port Scan
alert tcp any any -> $HOME_NET any (msg:"Port Scan Detected"; flags:S; detection_filter:track by_src, count 20, seconds 10; sid:1000003;)

#Q4 MITM ARP
alert ( msg: "ARPSPoof_ARP_CACHE_OVERWRITE_ATTACK"; sid: 4; gid: 112; rev: 1; metadata: rule-type preproc ; classtype:bad-unknown; )

#Q5 Brutes ssh
alert tcp any any -> $HOME_NET 22 (msg:"SSH Brute Force"; flags:S; detection_filter:track by_src, count 5, seconds 30; sid:1000005; rev:1;)

#Q6 malicious packets
alert icmp any any -> any any (msg:"ICMP large packet attack detected"; dsizes:>1000; sid:1000006; rev:1;)

#Q7
alert tcp any any -> any any (msg:"XSS attempt"; flow:to_server,established; uricontent:"/index.php"; nocase; content:<script>; nocase; classtype:web-application-attack; sid:1000007; rev:1;)
```

The document provides the custom Snort rules we wrote for specific attack scenarios:

- **DDoS/DoS Attack on Web Server:**

```
alert tcp any any -> $HOME_NET 80 (msg:"Potential DDoS Attack";
detection_filter:track by_dst, count 100, seconds 1; sid:1000001;)
```

- **alert:** Generate alert
- **tcp:** Protocol
- **any any:** Any source
- **->:** Traffic direction
- **\$HOME_NET 80:** Destination, HTTP
- **msg:** Alert message
- **detection_filter:** Trigger on 100 packets/sec
- **sid:** Rule ID

- **SQL Injection on a Web Application:**

```
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%27"; priority:1; sid:1000011;)
```

```
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%22"; priority:1; sid:1000012;)
```

```
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected"; content: "and"; nocase; priority:2; sid:1000013;)
```

```
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected"; content: "or"; nocase; priority:2; sid:1000014;)
```

```
alert tcp any any -> any 80 (msg: "Order by SQL Injection Detected"; content: "order"; nocase; priority:5; sid:1000015;)
```

```
alert tcp any any -> any 80 (msg: "Union Select SQL Injection Detected"; content: "union"; nocase; priority:5; sid:1000016;)
```

- **alert:** Generate alert
- **tcp:** Protocol
- **any any -> any 80:** From any source to port 80 (web server)
- **msg:** Alert message
- **content:** detect single quote (%27)/double quote(%22)/and/or/order/union
- **priority:** Severity level 1
- **sid:** Rule ID

- **Port Scanning and Information Gathering:**

```
alert tcp any any -> $HOME_NET any (msg:"Port Scan Detected"; flags:S; detection_filter:track by_src, count 20, seconds 10; sid:1000003;)
```

- **alert:** Issues an alert when the rule matches.
- **tcp:** The protocol monitored is TCP.
- **any any -> \$HOME_NET any:** It watches traffic from any source IP/port to any destination within **\$HOME_NET** across any ports.
- **msg:"Port Scan Detected":** The alert message.
- **flags:** Checks for TCP SYN flags (indicates the start of a connection attempt).

- **detection_filter**
by_src, count 20, seconds 10: The rule will trigger if the same source sends 20 connection attempts (SYN packets) within 10 seconds.
- **sid:1000003:** Unique identifier for the rule.

- **Man-in-the-Middle (MITM) Attack (ARP Spoofing):**

```
alert ( msg: "ARPSPOOF_ARP_CACHE_OVERWRITE_ATTACK"; sid: 4; gid: 112;
rev: 1; metadata: rule-type preproc ; classtype:bad-unknown; )
```

- **alert:** Generate alert
- **msg:** Alert message
- **sid:** Rule ID
- **gid:** Preprocessor group
- **rev:** Rule version
- **metadata:** Rule type
- **classtype:** Attack category

- **Brute Force Login Attempts:**

```
alert tcp any any -> $HOME_NET 22 (msg:"SSH Brute Force"; flags:S;
detection_filter:track by_src, count 5, seconds 30; sid:1000005; rev:1;)
```

- **alert:** Generate alert
- **tcp:** Protocol
- **any any -> \$HOME_NET 22:** Target SSH port on Victim IP
- **msg:** Alert message
- **flags:** SYN flag
- **detection_filter:** 5 attempts in 30 sec
- **sid:** Rule ID
- **rev:** Rule version

- **Crafted Malicious Traffic (ICMP Large Packets):**

```
alert icmp any any -> any any (msg:"ICMP large packet attack detected";
dsiz:>1000; sid:1000006; rev:1;)
```

- **alert:** Generate alert
- **icmp:** Protocol
- **any any -> any any:** Source to destination
- **msg:** Alert message
- **dsize:** Data size > 1000 bytes
- **sid:** Rule ID
- **rev:** Rule version

- **Research Based Attack (XSS Cross Site Scripting):**

```
alert tcp any any -> any any (msg:"XSS attempt"; flow:to_server,established;
uricontent:"/index.php"; nocase; content:"<script>"; nocase;
classtype:web-application-attack; sid:1000007; rev:1;)
```

- **alert:** Generate alert
- **tcp:** Protocol
- **any any -> any any:** From any source to any destination
- **msg:** Alert message
- **flow:** Direction to server, established connection
- **uricontent:** Look for /index.php in URI
- **nocase:** Case-insensitive match
- **content:** Detect <script> tag
- **classtype:** Classify as web application attack
- **sid:** Rule ID
- **rev:** Rule version

3. Attack Scenarios and Generated Snort Alerts

Scenarios:

- Scenario 1: DDoS/DoS attack using hping.

-Attacker's commands:

```
sudo hping3 -S --flood -V 192.168.20.17 -p 80
```

The screenshot shows a terminal window on a Kali Linux system. The user has run the command `sudo hping3 -S --flood -V 192.168.20.17 -p 80`. Below the terminal, a window titled "22i-1551" displays two IP addresses: 22I-1551 and 22i-1567. The terminal output shows numerous Snort alerts for potential DDoS attacks, with several lines highlighted by red arrows pointing to them. The alerts are timestamped from 10/20/20-05:29:018257 to 10/20/20-05:29:018636, all reporting Port Scan Detected or Potential DDoS Attack events against the target IP 192.168.20.17 on port 80.

- Left side of the screenshot shows Potential DDoS Attack alerts generated.

- Scenario 2: SQL injection on a simple PHP based website I hosted on the victim's pc.

-Attacker's commands:

```
curl -X POST "http://192.168.20.17/index.php" -d  
"username=admin' OR 1=1; -- &password=any_password"
```

The screenshot shows a terminal window on a Kali Linux system where the curl command `curl -X POST "http://192.168.20.17/index.php" -d "username=admin' OR 1=1; -- &password=any_password"` has been executed. The response from the server indicates a successful login, showing the page content "22I-1551" and "22i-1567". Below the terminal, a browser window titled "22i-1551" also displays the same login information. The terminal output at the bottom shows a Snort alert for SQL Injection Detected at 10/20/20-05:24:18.409290.

- Left side of the screenshot shows **OR SQL Injection Detected** alert generated.
- **Scenario 3:** Port scanning using nmap

-Attacker's commands:

```
nmap -sS -p 1-65535 -T4 192.168.20.17
```

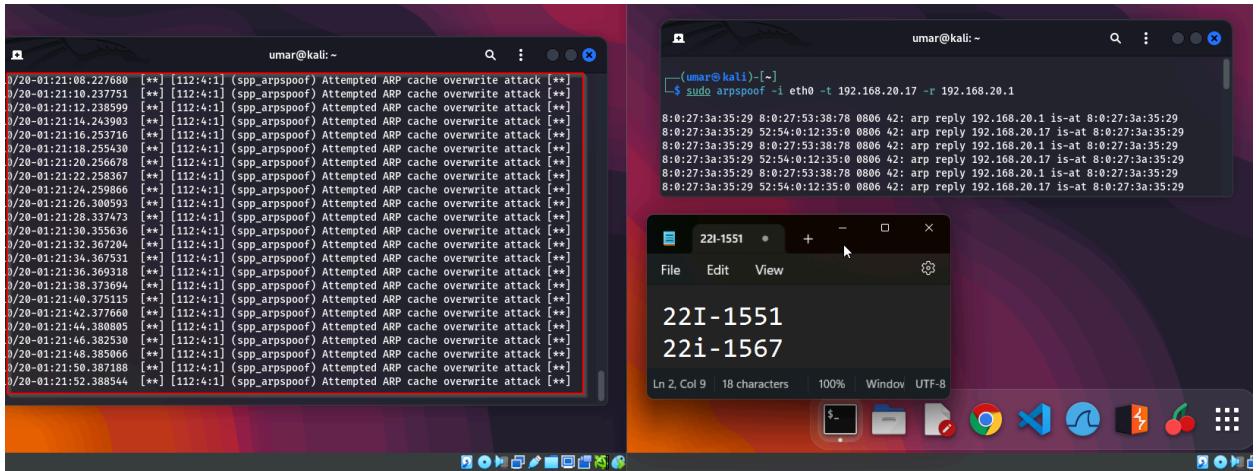
```
10/20-00:54:54.496367 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:59271
10/20-00:54:54.496368 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:125408
10/20-00:54:54.496369 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:62253
10/20-00:54:54.496369 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:38613
10/20-00:54:54.496368 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:58801
10/20-00:54:54.496368 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:6921
10/20-00:54:54.496369 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:25578
10/20-00:54:54.496362 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:62300
10/20-00:54:54.496362 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:33405
10/20-00:54:54.496362 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:15133
10/20-00:54:54.496362 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:63167
10/20-00:54:54.496362 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:20373
10/20-00:54:54.497304 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:23892
10/20-00:54:54.497304 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:65382
10/20-00:54:54.497304 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:13538
10/20-00:54:54.497304 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:50877
10/20-00:54:54.497678 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:1507
10/20-00:54:54.497678 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:52454
10/20-00:54:54.497678 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:49757
10/20-00:54:54.498100 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:26881
10/20-00:54:54.498110 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:11708
10/20-00:54:54.498110 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:19940
10/20-00:54:54.498110 [**] [1:1000003:0] Port Scan Detected [**] [Priority: 0] [TCP] 192.168.20.15:33302 -> 192.168.20.17:43942
```

- Left side of the screenshot shows **Port Scan Detected** alert generated.

- **Scenario 4:** MITM attack using arpspoof on victim's gateway ip

-Attacker's commands:

```
sudo arpspoof -i eth0 -t 192.168.20.17 -r 192.168.20.1
```

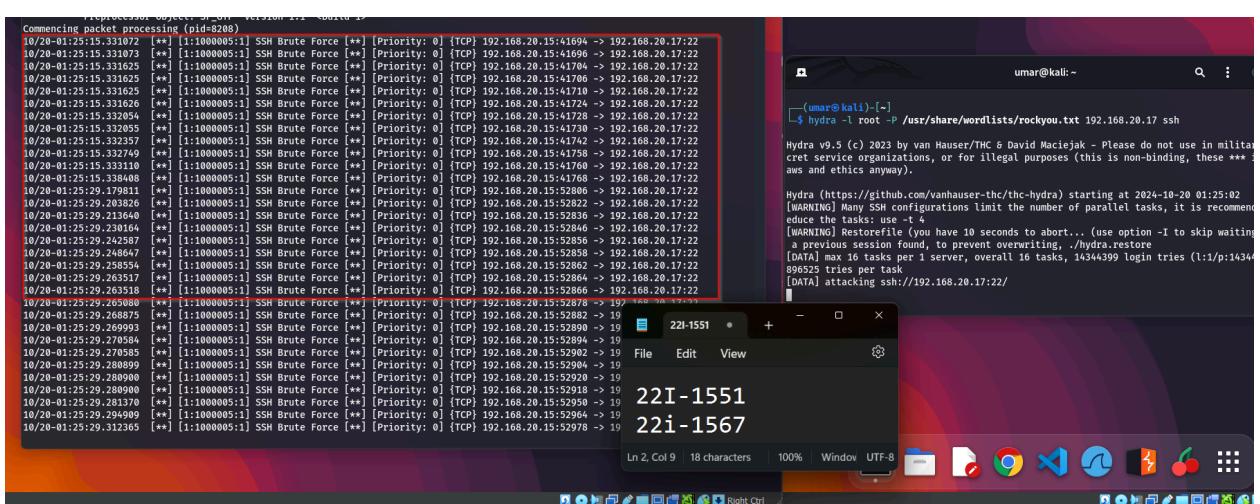


- Left side of the screenshot shows **Attempted ARP cache overwrite attack alerts generated.**

- **Scenario 5: Brute force SSH login using hydra**
 - **Attacker's command:**

```
hydra -l root -P /usr/share/wordlists/rockyou.txt 192.168.20.17
```

ssh



- Left side of the screenshot shows **SSH Brute Force** alerts generated.

- **Scenario 6:** Crafted malicious ICMP traffic using hping3

- #### - Attacker's command:

```
sudo hping3 -S --flood -V 192.168.20.17 -p 80
```

- Left side of the screenshot shows **ICMP large packet attack detected** alerts generated.

- **Scenario 7:** Performed XSS cross-site scripting on a PHP based website I hosted on the victim's pc.

- #### - Attacker's command:

```
curl -X POST -d "input=<script>alert('XSS')</script>"
```

<http://192.168.20.17/index.php>

```
Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_LDAP Version 1.0 <Build 1>
Preprocessor Object: SF_SQLP Version 1.0 <Build 4>
Preprocessor Object: SF_STP Version 1.1 <Build 1>
Preprocessor Object: SF_REPEATITION Version 1.1 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1> .....
Preprocessor Object: SF_DNS Version 1.1 <Build 4> .....
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DTP Version 1.1 <Build 1>
Commencing packet processing (pid=6414)

[0/20-00:11:18.091335 [**] [1:1407:6] WEB-MISC cross site scripting attempt [**] [Classification: Web Application Attack] [Priority: 1] [TCP
192.168.20.17:60986 -> 192.168.20.17:18]
[0/20-00:11:18.091335 [**] [1:100007:1] OR SQL Injection Detected [**] [Priority: 1] [TCP] 192.168.20.15:60986 -> 192.168.20.17:80
[0/20-00:11:18.091335 [**] [1:100007:1] XSS attempt [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.20.15:60986 -> 192.168.20.17:80

umar@kali:~
```

- Left side of the screenshot shows **XSS attempt** alerts generated.

4. Reflection on Snort's Effectiveness

Throughout our project, my partner and I analyzed Snort's performance based on the simulations we conducted together. We found that Snort demonstrated a strong ability to detect various network attacks in real time, including brute-force login attempts, ARP spoofing, and large ICMP packet attacks. Each of these incidents triggered alerts, showcasing Snort's effectiveness in monitoring network traffic.

However, we primarily used generic rules for detection, which sometimes resulted in false positives. For instance, legitimate traffic occasionally triggered alerts meant for specific attack detection. We noticed that sometimes rules for SQL injection were triggered during XSS (Cross-Site Scripting) attempts, highlighting the overlap in attack vectors and the potential confusion in rule applicability. This issue might stem from our surface-level knowledge of the various attacks and corresponding Snort rules. As we are still learning, it's possible that we didn't fully understand how to customize rules specifically for the threats we were testing, which emphasized how Snort can generate many false positives if generic commands are used without proper customization.

One thing we learned is that certain attacks, like ARP spoofing, require Snort's preprocessor to handle them. Unlike standard traffic inspections, ARP traffic needs to be processed differently, and the preprocessor is used to analyze this layer of traffic. This is something we weren't entirely familiar with at first, but we began to see how Snort treats ARP packets in a unique way through the preprocessor, inspecting them for patterns typical of spoofing attacks.

Ultimately, the effectiveness of Snort is entirely dependent on the rules implemented. Snort acts like a machine ticking boxes when certain requirements are met, and it will push out alerts based on those conditions. This project provided us with valuable insights into Snort's capabilities and limitations, as well as the importance of refining rules to match the specific context of the network being monitored. Our experience highlighted how critical it is to customize the rules properly to minimize false positives, as using generic rules often led to unnecessary alerts.