Project Report

Snort IDS/IPS Detection and Evasion

2nd August, 2025

This report confirms the successful deployment and validation of Snort as a network-based Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) on a Windows 10 host.

Basic Network Monitoring

Reconnaissance Detection

Intrusion Detection

Intrusion Prevention

This proof-of-concept (PoC) highlights Snort's effectiveness as an open-source solution for network security monitoring and proactive defense.

Security Analyst

Athar Imran

Snort IDS/IPS Detection and Evasion

This report confirms the successful deployment and validation of **Snort** as a network-based **Intrusion Detection System (IDS)** and **Intrusion Prevention System (IPS)** on a **Windows 10 host**.
Using a **Kali Linux VM** to simulate attacks, we demonstrated Snort's ability to **detect and block** various malicious activities, from basic sniffing to advanced threat prevention.

This proof-of-concept (PoC) highlights Snort's effectiveness as an open-source solution for network security monitoring and proactive defense.

Author: Athar Imran
Date: August 2, 2025

This project utilizes **Snort**, an open-source IDS/IPS, to demonstrate its practical application. The goal was to deploy Snort on Windows 10, simulate common attack scenarios from a Kali Linux VM, and prove Snort's detection and prevention capabilities. The PoC covers:

- **Basic Network Monitoring:** Sniffing and logging traffic.
- **Reconnaissance Detection:** Identifying scanning and information gathering.
- **Intrusion Detection:** Recognizing attack signatures.
- **Intrusion Prevention:** Actively blocking malicious traffic.

Contents

Phase 0: Lab Environment Setup

The following components are used to establish the lab environment:

- **Host Machine:** Windows 10 (as the target/protected system)
    - **Network Adapter Configuration:** Ensure the network adapter used by Snort is set to *promiscuous mode* to capture all network traffic on the segment.

- **Virtual Machine:** Kali Linux (as the attacking system)
    - **Network Adapter Configuration:** Configured to be on the *same network segment* as the Windows 10 host (e.g., Bridged Adapter or Host-only Network).

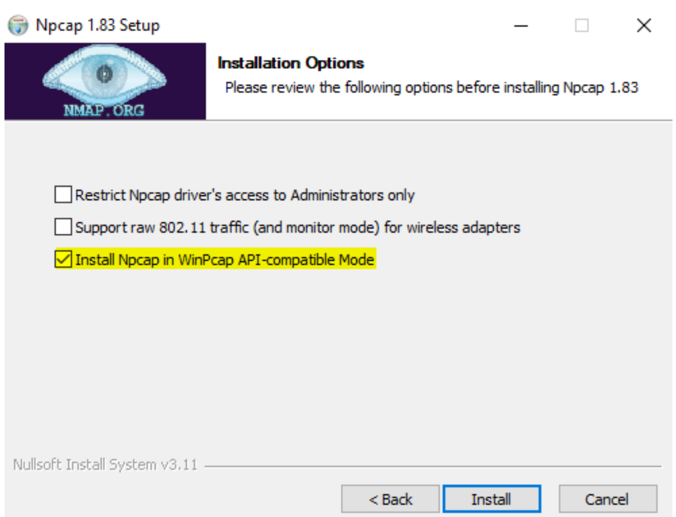
0.1 - Windows 10 Host Setup (Snort IDS/IPS):

(Alternatively, follow https://letsdefend.io/blog/how-to-install-and-configure-snort-on-windows)

Npcap:

Snort requires a packet capture library. Npcap is generally recommended as it is newer and actively maintained.

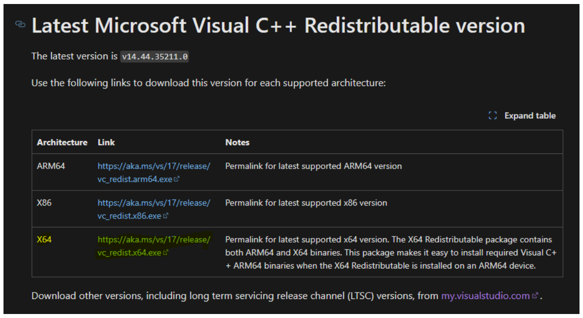
1. Install it from: `https://nmap.org/npcap/`
2. And ensure "Install Npcap in WinPcap API-compatible Mode" is selected during installation.



Visual C++ Redistributable:

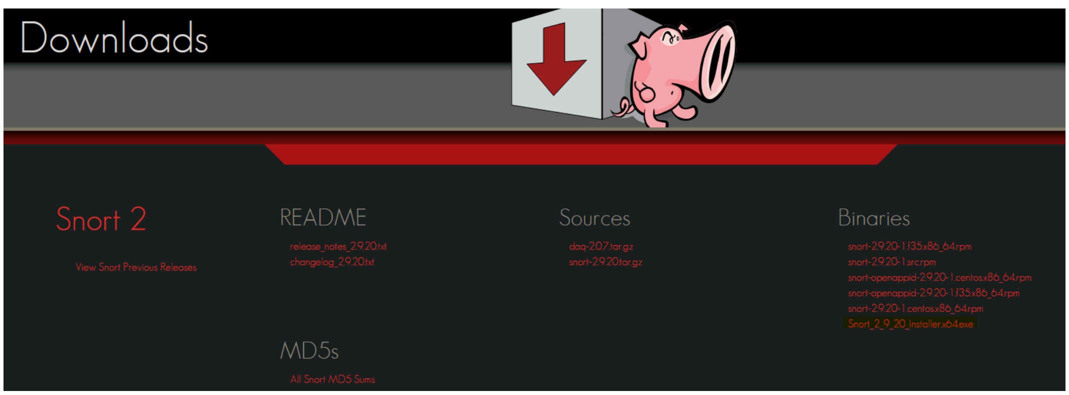
Snort relies on certain Microsoft Visual C++ runtime components.

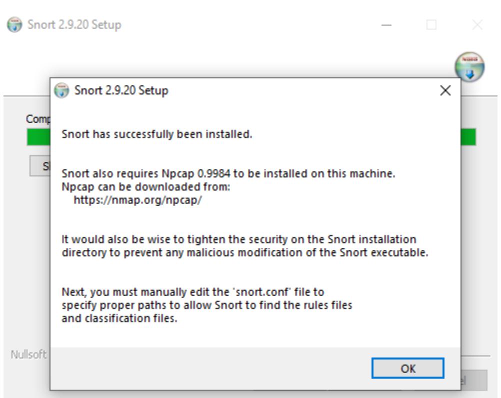
1. Download and install the latest Visual C++ Redistributable for Visual Studio from Microsoft's official website.



Snort:

1. Download and install the latest Snort from `https://www.snort.org/downloads`.





Configure Snort:

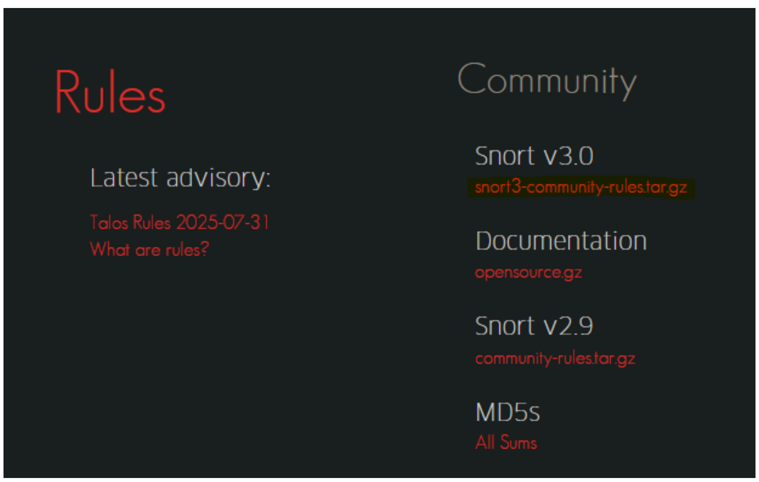
1. Navigate to C:\Snort\etc and modify the snort.conf file.
2. Replace the ‘any’ with the network interface IP (find it from `ipconfig /all` command).

```
###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.100.0/24
```

3. Configure the rule storage location:

```

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH c:\snort\rules
var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH ../preproc_rules
```

4. Download rule files from the Snort website:



5. Extract and place this rule file into the default Snort rule folder.
6. Modify the snort.conf to include those rules:

```

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH c:\snort\rules
var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH ../preproc_rules

include $RULE_PATH/snort3-community-rules\snort3-community.rules
```

Test Snort:

1. Open CMD as admin.
2. Navigate to the Snortnort installed directory.
3. And use this command for confirmation of completion of installation:

```
C:\Snort\bin>snort.exe -i 9 -A console
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{DD57E85D-B0E0-41FC-832B-6EA06F8260DC}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=1584)
WARNING: No preprocessors configured for policy 0.
08/03-14:50:16.985495 192.168.100.86:46579 -> 239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:19758 IpLen:20 DgmLen:153 DF
Len: 125
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

4. Find the interface using:

```
C:\Snort\bin>snort.exe -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Index  Physical Address     IP Address       Device Name     Description
-----  ----------------     ----------       -----------     -----------
    1  00:00:00:00:00:00    disabled         \Device\NPF_{69E0AB8C-FDA5-4F81-AC76-02340A00D154}      WAN Miniport (Network Monitor)
    2  00:00:00:00:00:00    disabled         \Device\NPF_{357F933D-FBAF-4241-AC58-53C424009920}      WAN Miniport (IPv6)
    3  00:00:00:00:00:00    disabled         \Device\NPF_{F82AD281-34FC-4815-856D-A00456F8005A}      WAN Miniport (IP)
    4  00:15:5D:69:C5:C1    172.19.64.1      \Device\NPF_{1CE0618D-2535-41A6-B3B0-33F1A38EC747}      Hyper-V Virtual Ethernet Adapter #5
    5  00:15:5D:16:91:48    172.21.144.1     \Device\NPF_{8A8CDA18-F6B9-4338-BCC2-FBF1FE45D15B}      Hyper-V Virtual Ethernet Adapter #4
    6  00:15:5D:C1:1F:67    172.21.128.1     \Device\NPF_{03928504-5F42-48A4-9451-CF2209A39A4F}      Hyper-V Virtual Ethernet Adapter #3
    7  00:15:5D:4B:83:0C    172.25.144.1     \Device\NPF_{85053AE2-4A42-4280-BDCF-B6DDF93DC6CB}      Hyper-V Virtual Ethernet Adapter #2
    8  64:27:37:82:FB:E5    169.254.157.61   \Device\NPF_{1133AF34-DB1C-4438-ADD4-12CEAB0082B7}      Bluetooth Device (Personal Area Network)
    9  9C:B7:0D:46:40:54    192.168.100.119 \Device\NPF_{DD57E85D-B0E0-41FC-832B-6EA06F8260DC}      Broadcom 802.11n Network Adapter
   10  00:50:56:C0:00:08    192.168.226.1    \Device\NPF_{AD343047-34AB-418A-AB14-9E49D0A1D27D}      VMware Virtual Ethernet Adapter for VMnet8
   11  00:50:56:C0:00:01    192.168.121.1    \Device\NPF_{76EB0BD4-889F-49A2-9269-05ABF2B86A22}      VMware Virtual Ethernet Adapter for VMnet1
   12  9E:B7:0D:46:48:54    169.254.71.208   \Device\NPF_{D8A04014-F59E-4157-A23E-6862EEE1E646}      Microsoft Wi-Fi Direct Virtual Adapter #3
   13  9E:B7:0D:46:40:54    169.254.106.148 \Device\NPF_{C2893B30-F0F9-40DB-AAF0-C2E3FEE5D3D0}      Microsoft Wi-Fi Direct Virtual Adapter #2
   14  00:15:5D:A0:64:3E    172.18.176.1     \Device\NPF_{F839497A-00A5-4C65-86FD-7F50E2CCF4A2}      Hyper-V Virtual Ethernet Adapter
   15  00:00:00:00:00:00    0000:0000:0000:0000:0000:0000:0000:0000 \Device\NPF_Loopback    Adapter for loopback traffic capture
   16  D0:67:E5:44:23:3E    169.254.142.13   \Device\NPF_{271E3F68-1536-4E0B-BF45-0D84BDF7D621}      Intel(R) 82579LM Gigabit Network Connection
```

0.2 - Kali Linux VM Setup (Attacker):

Network Configuration: Ensure the Kali Linux VM's network adapter is configured to be on the same network as the Windows 10 host. This allows direct communication and simulation of network attacks.

Tools: Kali Linux comes pre-installed with the necessary tools for attack simulation, including:

- `Nmap` for port scanning and OS fingerprinting.
- `hping3` for crafting custom packets (e.g., for DoS simulation).
- Potentially Metasploit Framework for exploit attempts (though this project emphasizes detection rather than full compromise)

Phase 1: Basic Network Monitoring and Analysis

Objective: *To demonstrate Snort's capability as a packet sniffer and logger.*

In this phase, Snort is used to generate a real-time packet capture display onthe terminal and also savea the log, which is used later to find out who is pinging ts.

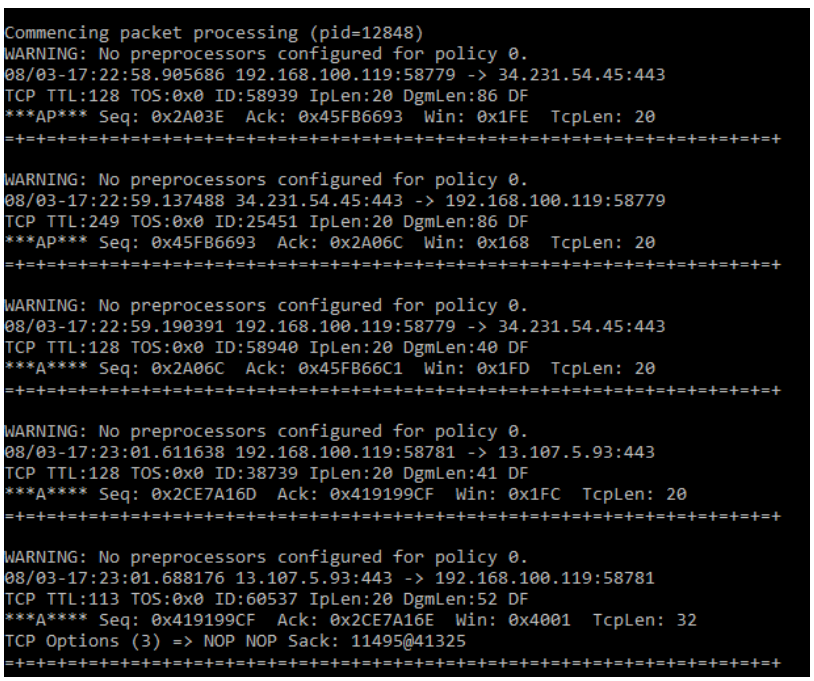
1.1 - Sniff Mode:

Windows Host (Snort):

- Run Snort in sniffer mode: `snort -v -i 9`

Observation (Windows Host):

- **Sniffer Mode:** Observe real-time packet headers scrolling in the command prompt. This demonstrates Snort's ability to capture and display live network traffic.



```
Commencing packet processing (pid=12848)
WARNING: No preprocessors configured for policy 0.
08/03-17:22:58.905686 192.168.100.119:58779 -> 34.231.54.45:443
TCP TTL:128 TOS:0x0 ID:58939 IpLen:20 DgmLen:86 DF
***AP*** Seq: 0x2A03E  Ack: 0x45FB6693  Win: 0x1FE  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
08/03-17:22:59.137488 34.231.54.45:443 -> 192.168.100.119:58779
TCP TTL:249 TOS:0x0 ID:25451 IpLen:20 DgmLen:86 DF
***AP*** Seq: 0x45FB6693  Ack: 0x2A06C  Win: 0x168  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
08/03-17:22:59.190391 192.168.100.119:58779 -> 34.231.54.45:443
TCP TTL:128 TOS:0x0 ID:58940 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x2A06C  Ack: 0x45FB66C1  Win: 0x1FD  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
08/03-17:23:01.611638 192.168.100.119:58781 -> 13.107.5.93:443
TCP TTL:128 TOS:0x0 ID:38739 IpLen:20 DgmLen:41 DF
***A**** Seq: 0x2CE7A16D  Ack: 0x419199CF  Win: 0x1FC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
08/03-17:23:01.688176 13.107.5.93:443 -> 192.168.100.119:58781
TCP TTL:113 TOS:0x0 ID:60537 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x419199CF  Ack: 0x2CE7A16E  Win: 0x4001  TcpLen: 32
TCP Options (3) => NOP NOP Sack: 11495@41325
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

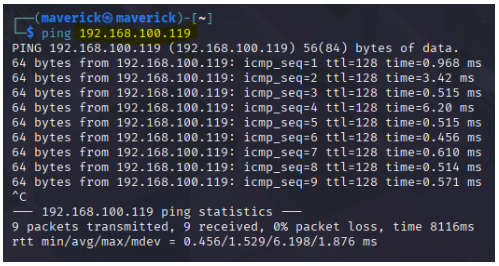
1.2 - Logger Mode:

Windows Host (Snort):

- Run Snort in packet logger mode: `snort -dev -l C:\Snort\log -i 9`

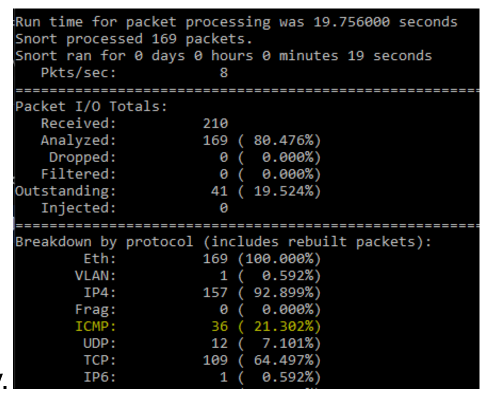
Kali Linux VM (Attacker):

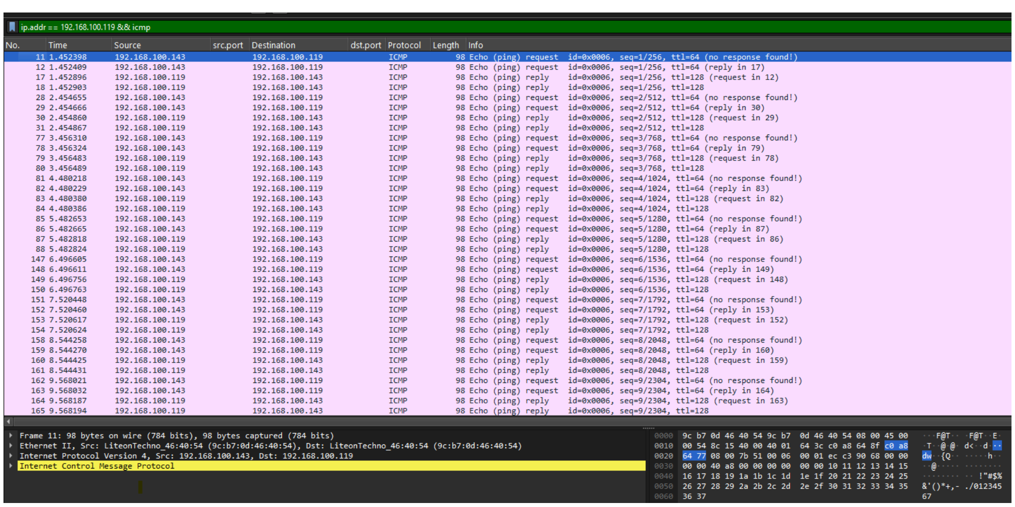
- Perform basic network activity, e.g., `ping 192.168.100.119`, `traceroute 192.168.100.119`, browse a website (if accessible).



Observation (Windows Host):

- **Packet Logger Mode:** After stopping Snort (Ctrl+C), navigate to `C:\Snort\log`. You will find subdirectories containing snort.log.<timestamp>.These logs can be analyzed with tools like Wireshark (wireshark -r <log_file>) to view the captured packets, proving Snort's logging capability.





Phase 2: Reconnaissance Detection

Objective: *To detect various reconnaissance techniques used by attackers to gather information about the target.*

In this phase, we will configure Snort to detect if someone is scanning our network.

2.1 - Configuration on Windows Host:

1. In the file *C:\Snort\etc\snort.conf*, ensure that the `sfportscan preprocessor` is uncommented.

```
# Portscan detection.  For more information, see README.sfportscan
preprocessor sfportscan: proto  { all } memcap { 10000000 } sense_level { high }
```

2. We will modify the rule file (CC:\Snort\rules\local.rules.rulesrules) to detect:

   a. Stealth & Overt Port Scan:

```
# Port Scans (Stealth & Overt)

alert tcp any any -> $HOME_NET any (msg:"Possible Stealth Scan (SYN-ACK without SYN)"; flags:SA,12; flow:stateless; sid:1000001; rev:1;)

alert tcp any any -> $HOME_NET any (msg:"Possible FIN Scan"; flags:FPU,12; flow:stateless; sid:1000002; rev:1;)

alert tcp any any -> $HOME_NET any (msg:"Possible NULL Scan"; flags:0,12; flow:stateless; sid:1000003; rev:1;)

alert tcp any any -> $HOME_NET any (msg:"Possible XMAS Scan"; flags:FPU,12; flow:stateless; sid:1000004; rev:1;)

alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"Possible UDP Scan"; dsize:0; sid:1000005; rev:1;)

##########################################################################################################
```

   b. Nmap OS Fingerprinting:

```
# Nmap OS Fingerprinting

alert ip any any -> $HOME_NET any (msg:"Possible Nmap OS Fingerprint (TTL Mismatch)"; ttl:!64,128; sid:1000006; rev:1;)
```

   c. Network Sweeps or Ping Sweeps:

```
# Network Sweeps (Ping Sweeps)

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Possible Ping Sweep"; itype:8; icode:0; detection_filter:track by_src, count 10, seconds 5; sid:1000007; rev:1;)
```

2.2 - Execute Snort: that

Now the rules are confi we'llwe'll run the Snort before we attacker runs the recon scan from the Kali Linux VM, using the command on the Windows host Snort:

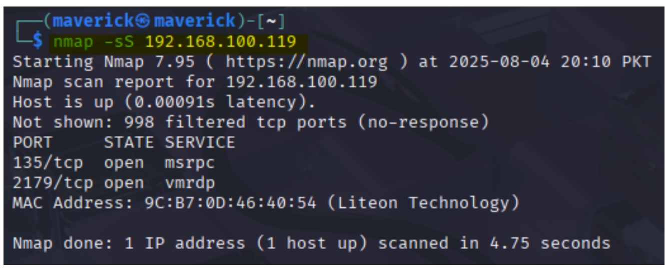
```
Snort -A console -c C:\Snort\etc\snort.conf -i 10
```

And let it continue.

2.3 - Recon Simulation:

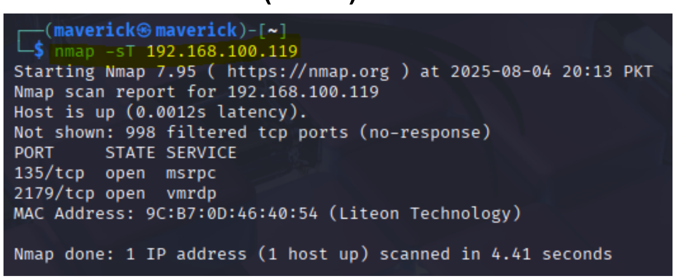
1. Port Scans:

1. **TCP SYN Scan (Stealth):**



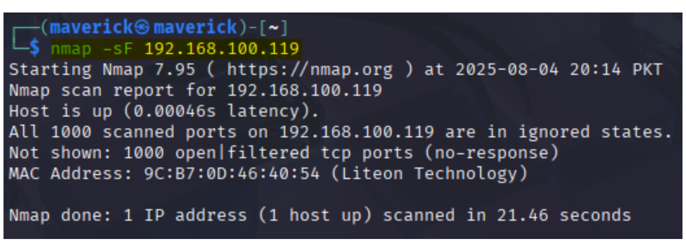
```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sS 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:10 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00091s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
2179/tcp open  vmrdp
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 4.75 seconds
```

2. **TCP Connect Scan (Overt):**



```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sT 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:13 PKT
Nmap scan report for 192.168.100.119
Host is up (0.0012s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
2179/tcp open  vmrdp
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 4.41 seconds
```

3. **FIN Scan:**



```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sF 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:14 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00046s latency).
All 1000 scanned ports on 192.168.100.119 are in ignored states.
Not shown: 1000 open|filtered tcp ports (no-response)
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.46 seconds
```

4. NULL Scan:

```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sN 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:15 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00044s latency).
All 1000 scanned ports on 192.168.100.119 are in ignored states.
Not shown: 1000 open|filtered tcp ports (no-response)
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.43 seconds
```

5. XMAS Scan:

```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sX 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:16 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00042s latency).
All 1000 scanned ports on 192.168.100.119 are in ignored states.
Not shown: 1000 open|filtered tcp ports (no-response)
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.46 seconds
```

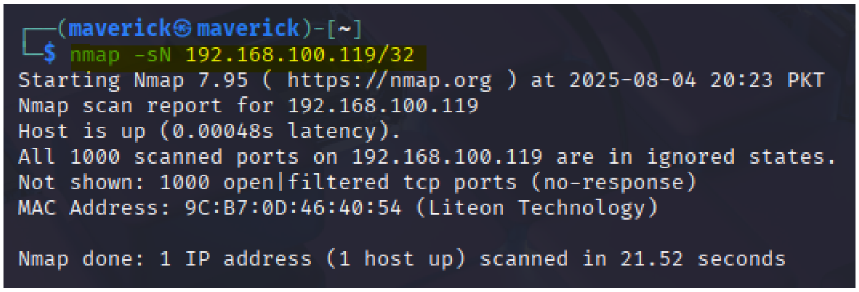
6. UDP Scan:

```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sU 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:19 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00049s latency).
All 1000 scanned ports on 192.168.100.119 are in ignored states.
Not shown: 1000 open|filtered udp ports (no-response)
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.77 seconds
```

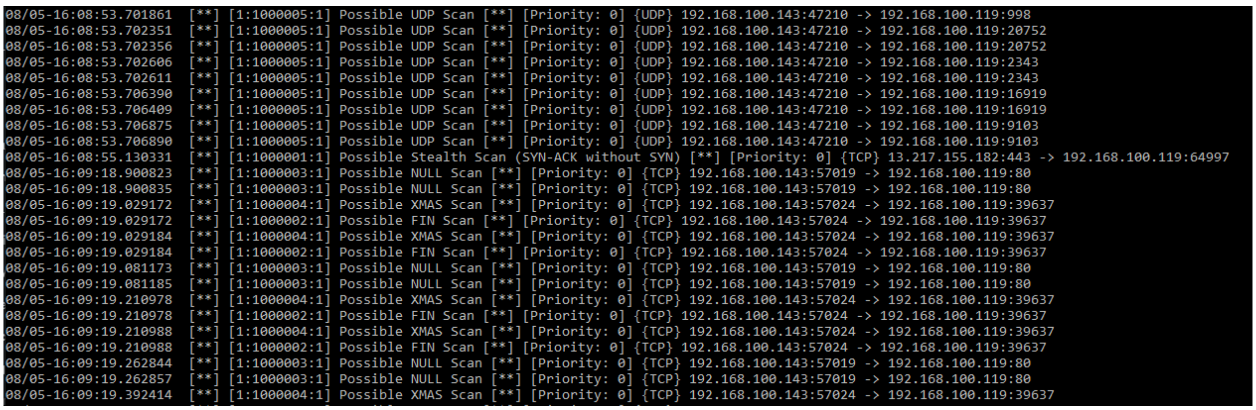
2. Operating System Fingerprinting:

```
┌──(maverick㉿maverick)-[~]
└─$ nmap -O 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 20:21 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00060s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
2179/tcp open  vmrdp
MAC Address: 9C:B7:0D:46:40:54 (Liteon Technology)
Warning: OSScan results may be unreliable because we could not find at least
1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 10|11|2019 (97%)
OS CPE: cpe:/o:microsoft:windows_10 cpe:/o:microsoft:windows_11 cpe:/o:micros
oft:windows_server_2019
Aggressive OS guesses: Microsoft Windows 10 1803 (97%), Microsoft Windows 10
1903 - 21H1 (97%), Microsoft Windows 11 (94%), Microsoft Windows 10 1809 (92%
), Microsoft Windows 10 1909 (91%), Microsoft Windows 10 1909 - 2004 (91%), W
indows Server 2019 (91%), Microsoft Windows 10 20H2 (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.o
rg/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.07 seconds
```

3. Network Sweeps (Ping Sweeps):



2.4 - Observation (Windows Host - Snort Console/Logs):



On a respective network scan through Nmap, the alert is generated on the console for the respective scan type.

Phase 3: Intrusion Detection System (IDS)

Objective: *To demonstrate Snort's ability to detect various intrusion attempts using signature-based detection and protocol analysis.*

3.1 - Rules creation:

First, we will create the rules in Snort to detect possible intrusion. I have written fthe ollowing rules according to how I am going to pose an intrusion on the host.

1- Malware Signature:

```
# Malware Signatures
alert tcp any any -> $HOME_NET any (msg:"Potential Malware Download (EXE over HTTP)"; content:"MZ";
classtype:trojan-activity; flow:established,to_client; file_data; sid:1000009; rev:1;)
```

This malware signature will look for file names that include malware andbeg dowdownloadedom any server over the internet.

2- Denial-of-Service (DoS):

If there are 50TCP requests from a single IP within a 5-second window, this rule will get triggered:

```
# Denial-of-Service (DoS) Attacks (SYN Flood)
alert tcp any any -> $HOME_NET any (msg:"Potential SYN Flood Attack"; flags:S,12; flow:stateless;
detection_filter:track by_src, count 50, seconds 5; classtype:attempted-dos; sid:1000011; rev:1;)
```

3- Specific Protocol Anomalies (Example: FTP Brute Force):

Assume we are having a highly confidential port, FTP. So this rule will get triggered if someone tries to attempt 5 failed logins within 10 seconds:

```
# Specific Protocol Anomalies (Example: FTP Brute Force)
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP Login Failure Brute Force"; content:"530 Login incorrect";
classtype:attempted-user; detection_filter:track by_src, count 5, seconds 10; sid:1000012; rev:1;)
```

4- Content Matching (Example: Specific keyword in HTTP traffic):

If there's a certain keyword we don't allow with HTTP, we can create an alert with this rule:

```
# Content Matching (Example: Specific keyword in HTTP traffic)
alert tcp any any -> $HOME_NET $HTTP_PORTS (msg:"Keyword 'Confidential' Detected in HTTP Traffic";
content:"Confidential"; nocase; http_uri; sid:1000013; rev:1;)
```

This rule will specifically look for the prohibited word specified in HTTP contents.

3.2 - Snort IDS scan :

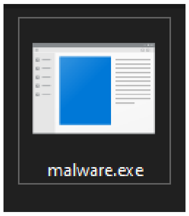
Now that the rules are created, run the IDS scan to detect malicious activities:

```
snort -A console -c C:\Snort\etc\snort.conf -i 10
```

3.3 - Attack Simulation & Observation:

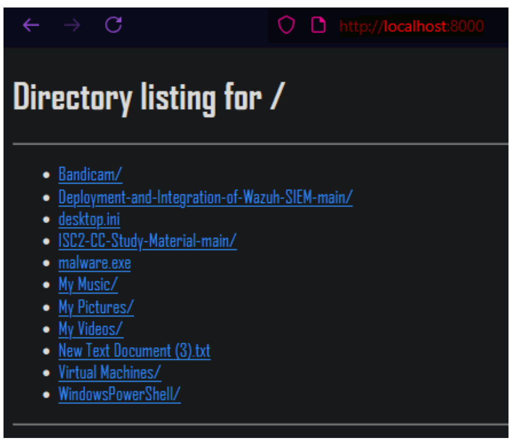
1- Malware Simulation:

1. For this step, we create a simple file that will act as malware for us. Within empty text, write anything like: “This is malware.exe”. And save the file as malware.exe:



2. Now host this malware file on the HTTP server:

```
c:\Users\Maverick\Documents>python -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::ffff:127.0.0.1 - - [05/Aug/2025 18:25:11] "GET / HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [05/Aug/2025 18:25:11] code 404, message File not found
::ffff:127.0.0.1 - - [05/Aug/2025 18:25:11] "GET /favicon.ico HTTP/1.1" 404 -
```



Observation:

3. Download the malware and look for the alert in Snortnort IDS running console:

```
Commencing packet processing (pid=12044)
08/05-18:28:07.817738  [**] [1:1000009:1] Potential Malware Download (EXE over HTTP) [**] [Classification:
 A Network Trojan was detected] [Priority: 1] {TCP} 44.213.8.194:443 -> 192.168.100.119:49334
```

2- Denial-of-Service (SYN Flood):

Target the Windows host from Kali Linux for a DOS attack:

```
┌──(maverick㉿maverick)-[~]
└─$ sudo hping3 -S -p 80 --flood 192.168.100.119
[sudo] password for maverick:
HPING 192.168.100.119 (eth0 192.168.100.119): S set, 40 headers + 0 data byte
s
hping in flood mode, no replies will be shown
^C
--- 192.168.100.119 hping statistic ---
28454 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Observation:

```
08/05-18:35:22.778039  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29856 -> 192.168.100.119:80
08/05-18:35:22.778146  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29857 -> 192.168.100.119:80
08/05-18:35:22.778150  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29857 -> 192.168.100.119:80
08/05-18:35:22.778256  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29859 -> 192.168.100.119:80
08/05-18:35:22.778260  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29859 -> 192.168.100.119:80
08/05-18:35:22.778366  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29860 -> 192.168.100.119:80
08/05-18:35:22.778370  [**] [1:1000011:1] Potential SYN Flood Attack [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.100.143:29860 -> 192.168.100.119:80
```

3- Protocol Anomaly (FTP Brute Force):

1. Ensure an FTP server is running on Windows (e.g., FileZilla Server).
2. From Kali, attempt multiple failed FTP logins:
    - `ftp <Windows_Host_IP>`
    - Enter incorrect usernames/passwords repeatedly.

   You will observe the FTP rule being triggered on the attempt and time specified.

4- Content Matching:

1. Keep the HTTP server running on Windows, and create an HTML file with the word “Confidential” in it.
2. Download it and observe the rule being triggered.

Phase 4: Intrusion Prevention System (IPS)

Objective: *To demonstrate Snort's ability to actively block malicious traffic.*

In this phase, we will block, instead of alerting to the malicious traffic.

4.1 - Rules creation:

```
# Drop SYN-ACK without SYN
drop tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Blocked Stealth Scan (SYN-ACK without SYN)"; flags:SA,12;
flow:stateless; sid:1000001; rev:2;)

# Drop SYN Flood attempts after threshold
drop tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Blocked SYN Flood Attack"; flags:S,12; flow:stateless;
detection_filter:track by_src, count 50, seconds 5; classtype:attempted-dos; sid:1000011; rev:2;)
```

For demonstration, we will use those two rules, which will block Stealth scan and flood attempts.

4.2 - Execution (Windows Host - IPS Mode):

```
Snort -A console -Q -c C:\Snort\etc\snort.conf -i 10
```

The -Q flag enables inline/IPS mode.

4.3 - Attack Simulations (Kali Linux VM):

We will repeat the same steps as below:

Stealth scan:

```
┌──(maverick㉿maverick)-[~]
└─$ nmap -sS 192.168.100.119
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-05 20:45 PKT
Nmap scan report for 192.168.100.119
Host is up (0.00045s latency).
Not shown: 993 filtered tcp ports (no-response)
PORT      STATE SERVICE
```

SYN Flood (DoS):

```
┌──(maverick㉿maverick)-[~]
└─$ sudo hping3 -S -p 80 --flood 192.168.100.119
HPING 192.168.100.119 (eth0 192.168.100.119): S set, 40 headers + 0 data byte
s
hping in flood mode, no replies will be shown
```

4.4 - Observation (Windows Host - Snort Console/Logs):

- Observe Snort alerts for `Suspicious Custom Command Detected` and, if using a `drop` rule, verify the packet is blocked.

- On the Windows listener (if used), confirm if the data was received, or if the connection was terminated, indicating prevention.

Conclusion

This proof of concept successfully demonstrated the robust capabilities of Snort as both an IDS and an IPS on a Windows 10 host. We effectively moved from basic network monitoring and packet logging to detecting complex reconnaissance techniques, identifying various intrusion attempts based on signatures and anomalies, and finally, actively preventing these threats.

The project highlighted the importance of:

- **Comprehensive Rule Sets:** Utilizing both community-provided and custom-created rules is essential for broad and specific threat detection.
- **Contextual Configuration:** Properly defining `HOME_NET` and configuring preprocessors is critical for accurate detection.
- **Operational Modes:** Understanding Snort's sniffer, logger, IDS, and IPS modes allows for flexible deployment based on security objectives.
- **Proactive Defense:** The transition from IDS (detection) to IPS (prevention) significantly enhances the security posture by actively mitigating threats.

While this lab environment provides a controlled setting, deploying Snort in a production environment requires careful planning, performance tuning, and integration with other security tools (e.g., SIEM for alert aggregation and analysis).

Recommendations

Based on this PoC, the following recommendations are made for further development and real-world application:

- **Regular Rule Updates:** Continuously update Snort rules to stay protected against emerging threats.
- **Performance Tuning:** Optimize Snort configuration for performance in production environments, especially when handling high traffic volumes.
- **Integration with SIEM:** Integrate Snort alerts with a Security Information and Event Management (SIEM) system (e.g., Splunk, ELK Stack, Security Onion) for centralized logging, correlation, and advanced analytics.
- **Automated Response:** Explore advanced IPS functionalities or scripting to automate responses beyond simple packet dropping, such as dynamically updating firewall rules or isolating compromised hosts.
- **Behavioral Anomaly Detection:** Supplement signature-based detection with behavioral anomaly detection techniques for identifying unknown or zero-day attacks.

- **Network Segmentation:** Implement network segmentation to limit the blast radius of any successful intrusion, even with an effective IPS.

This project serves as a strong foundation for understanding and implementing Snort in a security analyst role, providing practical experience in detecting and mitigating cyber threats.

About me

Find me on:

Linkedin

Github