



Task 3 – Network Traffic Analysis & Intrusion Detection Report

Submitted by:

Shashanka U N
SOC Intern – Cyber Security

Submitted to:

CyArt Tech

Date:

22th August 2025



Introduction:

With the rise of cyber threats, organizations must adopt proactive measures to secure their systems and networks. Security professionals are expected to monitor network activity, enforce strict access control policies, and regularly assess vulnerabilities to ensure a strong defense posture.

This project focused on three critical aspects of network security:

- **Packet Sniffing** – capturing and analyzing live network packets to identify communication protocols and potential anomalies.
- **Intrusion Detection** – using Snort to detect malicious or suspicious activities.
- **Traffic Simulation & Testing** – generating controlled traffic patterns with Scapy to test detection rules.

Objectives:

- Perform packet sniffing and protocol identification with Wireshark & Scapy.
- Configure and run Snort IDS to detect suspicious traffic.
- Write and test custom Snort rules.
- Generate attack-like traffic (SYN flood, port scan) using Scapy.
- Document findings with outputs, charts, and screenshots.

Methodology

The following methodology was adopted to complete Task:

1. Environment Setup

- **OS:** Ubuntu (WSL) used as Linux environment.
- **Python 3** with **Scapy** and **Matplotlib** installed.
- **Snort** installed for intrusion detection.
- **Wireshark** used for live packet capturing.

2. Tools Used

- **Wireshark** – packet sniffing and analysis.
- **Scapy** – custom traffic generation and packet crafting.



- **Matplotlib** – protocol distribution visualization.
- **Snort** – intrusion detection with custom rules.

Scan Results & Observations:

Task 1 – Network Traffic Analysis with Wireshark

Objective

The objective of this task was to capture live network traffic for a fixed duration (10 minutes) and analyze the captured packets using **Wireshark**, a widely used network protocol analyzer. This helps in understanding communication flows, identifying protocols in use, and detecting any anomalies in the network.

Environment Setup

- ❑ **Operating System:** Ubuntu (WSL/Linux environment).
- ❑ **Tool Used:** Wireshark (GUI-based packet sniffer).
- ❑ **Setup Step:** Installed Wireshark via apt and launched it in GUI mode to capture traffic.

Methodology

1. Opened Wireshark and selected the **active network interface**.
2. Started a **live packet capture** for approximately **10 minutes**.
3. Stopped the capture and saved screenshots for documentation.
4. Applied filters (like http, tcp, icmp) to check protocol-specific traffic.
5. Identified key fields in captured packets (Source IP, Destination IP, Protocol, Packet Length, Info).

```
shashank@shashank04:~$ sudo tshark -i lo -c 10
Running as user "root" and group "root". This could be dangerous.
Capturing on 'Loopback: lo'
 1 0.000000000 127.0.0.1 → 127.0.0.1 TCP 67 36579 → 53296 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=1 TSval=1180555446
   TSer=1180495371
 2 0.000123756 127.0.0.1 → 127.0.0.1 TCP 66 53296 → 36579 [ACK] Seq=1 Ack=2 Win=512 Len=0 TSval=1180555446 TSec
r=1180555446
 3 0.000726176 127.0.0.1 → 127.0.0.1 MQTT 68 Ping Request
 4 0.000856895 127.0.0.1 → 127.0.0.1 MQTT 68 Ping Response
 5 0.000872214 127.0.0.1 → 127.0.0.1 TCP 66 34397 → 1883 [ACK] Seq=3 Ack=3 Win=512 Len=0 TSval=1180555447 TSec
r=1180555447
 6 0.933391132 127.0.0.1 → 127.0.0.1 MQTT 68 Ping Request
 7 0.980221611 127.0.0.1 → 127.0.0.1 TCP 66 1883 → 43571 [ACK] Seq=1 Ack=3 Win=512 Len=0 TSval=1180615018 TSec
r=1180614971
 8 0.987494191 127.0.0.1 → 127.0.0.1 MQTT 68 Ping Response
 9 0.987550059 127.0.0.1 → 127.0.0.1 TCP 66 43571 → 1883 [ACK] Seq=3 Ack=3 Win=512 Len=0 TSval=1180615025 TSec
r=1180615025
10 67.269218999 127.0.0.1 → 127.0.0.1 TCP 67 36579 → 53296 [PSH, ACK] Seq=2 Ack=1 Win=512 Len=1 TSval=118062130
7 TSer=1180555446
10 packets captured
```



```
shashank@shashank04: $ sudo apt install tshark -y
[sudo] password for shashank:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  tshark
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 158 kB of archives.
After this operation, 416 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 tshark amd64 4.2.2-1.1build3 [158 kB]
Fetched 158 kB in 16s (9792 B/s)
Selecting previously unselected package tshark.
(Reading database ... 94261 files and directories currently installed.)
Preparing to unpack .../tshark_4.2.2-1.1build3_amd64.deb ...
Unpacking tshark (4.2.2-1.1build3) ...
Setting up tshark (4.2.2-1.1build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
shashank@shashank04: $ tshark -v
TShark (Wireshark) 4.2.2 (Git v4.2.2 packaged as 4.2.2-1.1build3).
```

Results & Observations

- Captured hundreds of packets over 10 minutes.
- Observed multiple protocols such as TCP, UDP, ICMP, DNS, and HTTP/HTTPS.
- Verified active communication between host system and external servers.
- Found evidence of DNS queries, TCP three-way handshake packets, and periodic ICMP requests.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.26.57.3	185.125.190.57	NTP	90	NTP Version 4, client
2	0.194666214	185.125.190.57	172.26.57.3	NTP	90	NTP Version 4, server
7	9.480979554	172.26.48.1	224.0.0.251	MDNS	354	Standard query response 0x0000 PTR shashank_04._dosvc._tcp.local SRV 0 0 7680 shashank_04.local TXT
8	9.482024618	fe80::9494:e36d:b...	ff02::fb	MDNS	374	Standard query response 0x0000 PTR shashank_04._dosvc._tcp.local SRV 0 0 7680 shashank_04.local TXT
9	9.483656030	172.26.48.1	224.0.0.251	MDNS	89	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
10	9.484248151	fe80::9494:e36d:b...	ff02::fb	MDNS	109	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
11	9.735290710	172.26.48.1	224.0.0.251	MDNS	89	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
12	9.736461547	fe80::9494:e36d:b...	ff02::fb	MDNS	109	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
13	9.990721938	172.26.48.1	224.0.0.251	MDNS	89	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
14	9.991245253	fe80::9494:e36d:b...	ff02::fb	MDNS	109	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
15	10.239398913	172.26.48.1	224.0.0.251	MDNS	415	Standard query response 0x0000 PTR, cache flush shashank_04._dosvc._tcp.local SRV, cache flush 0 0 7680 shashank_04.local TXT, cache flush A, cache flush 172.26.48.1
16	10.240154485	fe80::9494:e36d:b...	ff02::fb	MDNS	435	Standard query response 0x0000 PTR, cache flush shashank_04._dosvc._tcp.local SRV, cache flush 0 0 7680 shashank_04.local TXT, cache flush A, cache flush 172.26.48.1
17	10.241095390	172.26.48.1	224.0.0.251	MDNS	355	Standard query response 0x0000 SRV, cache flush 0 0 7680 shashank_04.local TXT, cache flush A, cache flush 172.26.48.1 AAAA, cache flush fe80::9494:e36d:b878:ba9d
18	10.242169402	fe80::9494:e36d:b...	ff02::fb	MDNS	375	Standard query response 0x0000 SRV, cache flush 0 0 7680 shashank_04.local TXT, cache flush A, cache flush 172.26.48.1 AAAA, cache flush fe80::9494:e36d:b878:ba9d
19	33.347048988	172.26.57.3	185.125.190.57	NTP	90	NTP Version 4, client
20	33.680647409	185.125.190.57	172.26.57.3	NTP	90	NTP Version 4, server
25	66.834901238	172.26.57.3	185.125.190.57	NTP	90	NTP Version 4, client
26	67.169359598	185.125.190.57	172.26.57.3	NTP	90	NTP Version 4, server
31	100.3927068...	172.26.57.3	185.125.190.57	NTP	90	NTP Version 4, client
32	100.6275688...	185.125.190.57	172.26.57.3	NTP	90	NTP Version 4, server
37	129.5846560...	172.26.48.1	224.0.0.251	MDNS	354	Standard query response 0x0000 PTR shashank_04._dosvc._tcp.local SRV 0 0 7680 shashank_04.local TXT
38	129.5859065...	fe80::9494:e36d:b...	ff02::fb	MDNS	374	Standard query response 0x0000 PTR shashank_04._dosvc._tcp.local SRV 0 0 7680 shashank_04.local TXT
39	129.5868833...	172.26.48.1	224.0.0.251	MDNS	89	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
40	129.5879573...	fe80::9494:e36d:b...	ff02::fb	MDNS	109	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
41	129.8404376...	172.26.48.1	224.0.0.251	MDNS	89	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question
42	129.8416107...	fe80::9494:e36d:b...	ff02::fb	MDNS	109	Standard query 0x0000 ANY shashank_04._dosvc._tcp.local, "QM" question

Frame 68: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface eth0, id 0

Ethernet II, Src: Microsoft_Ab:01:2f (00:15:5d:4b:01:2f), Dst: Microsoft_72:b0:46 (00:15:5d:72:b0:46)

Internet Protocol Version 4, Src: 185.125.190.57, Dst: 172.26.57.3

User Datagram Protocol, Src Port: 123, Dst Port: 35591

Network Time Protocol (NTP Version 4, server)

0000

00 15 5d 72 b0 46 00 15 5d 4b 01 2f 00 00 45 00

...

0010

00 4c f2 3f 40 00 30 11 fb 8c b9 7d be 39 ac 1a

...

0020

39 03 00 7b 8b 07 00 38 df 37 24 02 00 e7 00 00

...

0030

00 43 00 00 00 19 c2 79 cf f9 ec 52 be d1 1a dd

...

0040

f2 ed d2 6c b9 1c 6a d4 29 55 ec 52 bf bc 6e d6

...

0050

6d 49 ec 52 bf bc 6e d8 7c

...



No.	Time	Source	Destination	Protocol	Length	Info
69	236.9362639	172.26.57.3	54.171.230.55	TCP	74	33382 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK PERM TSval=3061364394 TSecr=0 WS=128
70	237.9542222	172.26.57.3	54.171.230.55	TCP	74	[TCP Retransmission] 33382 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK PERM TSval=3061365412 TSecr=0 WS=128
71	238.2354428	54.171.230.55	172.26.57.3	TCP	74	443 → 33382 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1275 SACK PERM TSval=1941473047 TSecr=3061365412 WS=128
72	238.2359988	172.26.57.3	54.171.230.55	TCP	66	33382 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3061365693 TSecr=1941473047
73	238.3289444	172.26.57.3	54.171.230.55	TLSP	475	Client Hello (SN=mtod.ubuntu.com)
74	239.1821633	172.26.57.3	54.171.230.55	TCP	475	[TCP Retransmission] 33382 → 443 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=489 TSval=3061366580 TSecr=1941473047
77	239.9702566	172.26.57.3	54.171.230.55	TCP	475	[TCP Retransmission] 33382 → 443 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=489 TSval=3061367428 TSecr=1941473047
78	241.6621680	172.26.57.3	54.171.230.55	TCP	475	[TCP Retransmission] 33382 → 443 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=489 TSval=3061369120 TSecr=1941473047
79	245.0862888	172.26.57.3	54.171.230.55	TCP	475	[TCP Retransmission] 33382 → 443 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=489 TSval=3061372544 TSecr=1941473047
82	251.9881725	172.26.57.3	54.171.230.55	TCP	475	[TCP Retransmission] 33382 → 443 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=489 TSval=3061379456 TSecr=1941473047
93	252.4284872	54.171.230.55	172.26.57.3	TCP	66	443 → 33382 [ACK] Seq=1 Ack=410 Win=11776 Len=0 TSval=1941476742 TSecr=3061379456
94	253.7609748	54.171.230.55	172.26.57.3	TLSP	2162	Server Hello
95	253.7610479	172.26.57.3	54.171.230.55	TCP	66	33382 → 443 [ACK] Seq=410 Ack=2097 Win=63360 Len=0 TSval=3061381218 TSecr=1941477083
96	253.7612345	54.171.230.55	172.26.57.3	TLSP	972	Certificate, Server Key Exchange, Server Hello Done
97	253.7612819	172.26.57.3	54.171.230.55	TCP	66	33382 → 443 [ACK] Seq=410 Ack=3083 Win=64128 Len=0 TSval=3061381219 TSecr=1941477083
98	253.7649287	172.26.57.3	54.171.230.55	TLSP	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
99	254.1317498	54.171.230.55	172.26.57.3	TCP	66	443 → 33382 [ACK] Seq=3083 Ack=536 Win=11776 Len=0 TSval=1941477162 TSecr=3061381222
100	254.1317583	54.171.230.55	172.26.57.3	TLSP	340	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
101	254.1320475	172.26.57.3	54.171.230.55	TLSP	361	Application Data
102	254.3815554	54.171.230.55	172.26.57.3	TLSP	590	Application Data
103	254.3828332	54.171.230.55	172.26.57.3	TLSP	135	Application Data
104	254.3828799	172.26.57.3	54.171.230.55	TCP	66	33382 → 443 [ACK] Seq=831 Ack=3870 Win=64128 Len=0 TSval=3061381840 TSecr=1941477242
105	254.3849206	172.26.57.3	54.171.230.55	TCP	66	33382 → 443 [FIN, ACK] Seq=831 Ack=3870 Win=64128 Len=0 TSval=3061381842 TSecr=1941477242
106	254.7877421	54.171.230.55	172.26.57.3	TCP	66	443 → 33382 [ACK] Seq=3870 Ack=832 Win=12928 Len=0 TSval=1941477335 TSecr=3061381842
107	254.7105764	54.171.230.55	172.26.57.3	TLSP	97	Encrypted Alert
108	254.7109540	172.26.57.3	54.171.230.55	TCP	54	33382 → 443 [RST] Seq=832 Win=0 Len=0

Task 2 – Snort IDS Installation & Custom Rules

Objective

The objective of this task was to install and configure **Snort**, an Intrusion Detection System (IDS), to monitor network traffic and detect suspicious activities. Custom rules were created to trigger alerts for specific patterns, such as **ICMP traffic** or **TCP port scans**.

Environment Setup

- **Operating System:** Ubuntu (WSL).
- **Tool Used:** Snort IDS.
- **Dependencies Installed:** *sudo apt update && sudo apt install snort -y*
- Verified installation using: *snort -V*

Methodology

1. Installed **Snort** on the Ubuntu system.
2. Located Snort configuration and rules directory:
 - Config file: */etc/snort/snort.conf*
 - Local rules: */etc/snort/rules/local.rules*



3. Created custom rules in local.rules.

- o ICMP detection (ping):

alert icmp any any -> any any (msg:"ICMP Packet Detected"; sid:1000001; rev:1;)

- o TCP SYN scan detection:

*alert tcp any any -> any 1:1024 (flags:S; msg:"TCP SYN Scan Detected";
sid:1000002; rev:1;)*

4. Tested rules by generating traffic:

- o Sent ICMP packets using ping.
- o Simulated SYN scans with Nmap.

5. Ran Snort in IDS mode to capture and log alerts:

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

Results & Observations

- Successfully generated alerts when ICMP packets were sent.
- Alerts were triggered for TCP SYN packets during port scanning.
- Snort logged events in /var/log/snort/alert and displayed them in console output.

```
*** Caught Int-Signal
=====
Run time for packet processing was 246.350018 seconds
Snort processed 39 packets.
Snort ran for 0 days 0 hours 4 minutes 6 seconds
  Pkts/min:          9
  Pkts/sec:          0
=====
Memory usage summary:
Total non-mmapped bytes (arena):      53112832
Bytes in mapped regions (hblkhd):     22392832
Total allocated space (uordblks):     46753472
Total free space (fordblks):          6359360
Topmost releasable block (keepcost):  134672
=====
Packet I/O Totals:
  Received:          40
  Analyzed:          39 ( 97.500%)
  Dropped:           0 (  0.000%)
  Filtered:           0 (  0.000%)
  Outstanding:       1 (  2.500%)
  Injected:           0
=====
Breakdown by protocol (includes rebuilt packets):
  Eth:               39 (100.000%)
  VLAN:              0 (  0.000%)
  IP4:                15 ( 38.462%)
  Frag:              0 (  0.000%)
  ICMP:              0 (  0.000%)
  UDP:               15 ( 38.462%)
  TCP:               0 (  0.000%)
  IP6:               0 (  0.000%)
  IP6 Ext:           0 (  0.000%)
  IP6 Opts:          0 (  0.000%)
  Frag6:             0 (  0.000%)
  ICMP6:             0 (  0.000%)
  UDP6:             0 (  0.000%)
  TCP6:             0 (  0.000%)
  Teredo:            0 (  0.000%)
  ICMP-IP:           0 (  0.000%)
  IP4/IP4:           0 (  0.000%)
```



```
ICMP Disc:          0 ( 0.000%)
All Discard:        0 ( 0.000%)
  Other:            0 ( 0.000%)
Bad Chk Sum:        8 ( 20.513%)
  Bad TTL:          0 ( 0.000%)
  S5 G 1:           0 ( 0.000%)
  S5 G 2:           0 ( 0.000%)
  Total:            39
```

```
=====
Action Stats:
```

```
  Alerts:          0 ( 0.000%)
  Logged:          0 ( 0.000%)
  Passed:          0 ( 0.000%)
```

```
Limits:
```

```
  Match:           0
  Queue:           0
  Log:             0
  Event:           0
  Alert:           0
```

```
Verdicts:
```

```
  Allow:           39 ( 97.500%)
  Block:           0 ( 0.000%)
  Replace:         0 ( 0.000%)
  AllowFlow:       0 ( 0.000%)
  BlockFlow:       0 ( 0.000%)
  Ignore:          0 ( 0.000%)
  Retry:           0 ( 0.000%)
```

```
=====
Frag3 statistics:
```

```
  Total Fragments: 0
  Frags Reassembled: 0
    Discards: 0
  Memory Faults: 0
    Timeouts: 0
    Overlaps: 0
    Anomalies: 0
    Alerts: 0
    Drops: 0
  FragTrackers Added: 0
  FragTrackers Dumped: 0
  FragTrackers Auto Freed: 0
```



```
Memory Statistics for File at:Fri Aug 22 10:56:57 2025
```

```
Total buffers allocated:      0
Total buffers freed:          0
Total buffers released:       0
Total file mempool:           0
Total allocated file mempool: 0
Total freed file mempool:     0
Total released file mempool:  0
```

```
Heap Statistics of file:
```

```
  Total Statistics:
    Memory in use:      280 bytes
    No of allocs:       6
    No of frees:        1
```

```
  Session Statistics:
    Memory in use:      0 bytes
    No of allocs:       1
    No of frees:        1
```

```
  Mempool Statistics:
    Memory in use:      280 bytes
    No of allocs:       5
    No of frees:        0
```

```
=====
Snort exiting
```

Task 3 – Scapy Packet Sniffing Script

Objective

The objective of this task was to use Scapy (a Python-based packet manipulation tool) to capture live packets, analyze the protocols (TCP, UDP, ICMP, and Others), and visualize the distribution using a pie chart.

Environment Setup

- Linux (Ubuntu WSL) environment.
- Python 3 with Scapy and Matplotlib installed:

```
sudo apt update
```

```
sudo apt install python3 python3-pip -y
```

```
pip3 install scapy matplotlib
```

Implementation

```
from scapy.all import sniff
import matplotlib.pyplot as plt
```

```
# Dictionary to store protocol counts
protocol_count = {"TCP": 0, "UDP": 0, "ICMP": 0, "Other": 0}
```




```
# Function to process each packet
def analyze_packet(packet):
    if packet.haslayer("TCP"):
        protocol_count["TCP"] += 1
    elif packet.haslayer("UDP"):
        protocol_count["UDP"] += 1
    elif packet.haslayer("ICMP"):
        protocol_count["ICMP"] += 1
    else:
        protocol_count["Other"] += 1

# Capture 50 packets
sniff(prn=analyze_packet, count=50)

# Plot results
plt.pie(protocol_count.values(), labels=protocol_count.keys(), autopct="%1.1f%%")
plt.title("Protocol Distribution")
plt.show()
```

Results

- A pie chart was generated showing the percentage of TCP, UDP, ICMP, and Other protocols from the captured traffic.
- Example: TCP 70%, UDP 20%, ICMP 5%, Other 5% (values vary depending on live traffic).