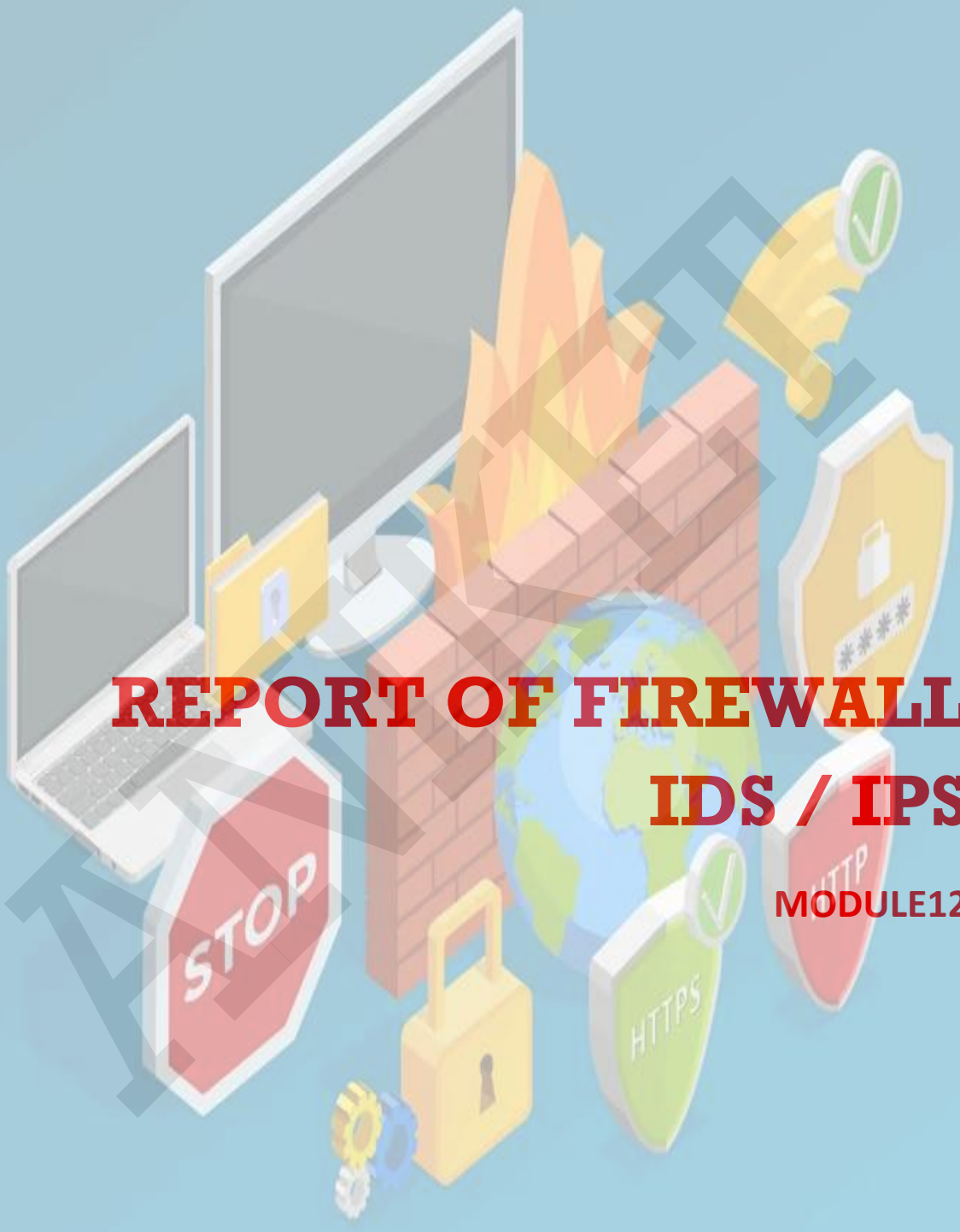


REPORT OF FIREWALL

IDS / IPS

MODULE12

Aniket Sunil Pagare

Table of Contents

5. Snort

5.1 Definition of Snort
5.2 Uses/Objectives of Snort
5.3 Snort Configuration

---

6. Windows Firewall Configuration

6.1 Definition of Windows Firewall
6.2 Purpose of Windows Firewall
6.3 Key Features
6.4 Inbound Rules
6.5 Outbound Rules

Extra Activity Section

---

7. ZoneAlarm Firewall

7.1 Definition of ZoneAlarm Firewall
7.2 Key Features
7.3 How to Use ZoneAlarm Firewall

---

8. Attack Detection Tools

8.1 HoneyBot (Attack Detection)
8.1.1 Definition
8.1.2 Common Use Cases
8.1.3 How to Use HoneyBot

8.2 KFSensor (Attack Detection)
8.2.1 Definition
8.2.2 Key Features
8.2.3 How to Use KFSensor

8.3 Wireshark (Attack Detection)

8.3.1 Definition

8.3.2 Wireshark Features

8.3.3 How to Use Wireshark

---

9. Defending Against Evasion Techniques

9.1 Defending Against Firewall Evasion

9.1.1 Common Firewall Evasion Techniques Attackers Use

9.1.2 How to Defend Against Firewall Evasion

9.2 Defending Against IDS Evasion

9.2.1 Common IDS Evasion Techniques Attackers Use

9.2.2 How to Defend Against IDS Evasion

9.3 Defending Against IPS Evasion

9.3.1 Common IPS Evasion Techniques Attackers Use

9.3.2 How to Defend Against IPS Evasion

FIREWALL IDS / IPS

What Is Firewall-:

A **firewall** is a **network security device or software** that monitors and controls **incoming and outgoing network traffic** based on **predefined security rules**. Its main purpose is to **block unauthorized access** to or from a private network while allowing legitimate communication to pass through.

🔥 **Types of Firewalls:**

1. **Packet-Filtering Firewall**
    - Checks packets against rules (e.g., IP, port, protocol).
    - Basic and fast, but limited inspection.
2. **Stateful Inspection Firewall**
    - Tracks active connections and makes decisions based on the state of the traffic.
    - More secure than packet-filtering.
3. **Application-Level Gateway (Proxy Firewall)**
    - Filters traffic at the application layer (e.g., HTTP, FTP).
    - Can inspect the contents of traffic, providing deep security.
4. **Next-Generation Firewall (NGFW)**
    - Combines traditional firewall with additional features like:

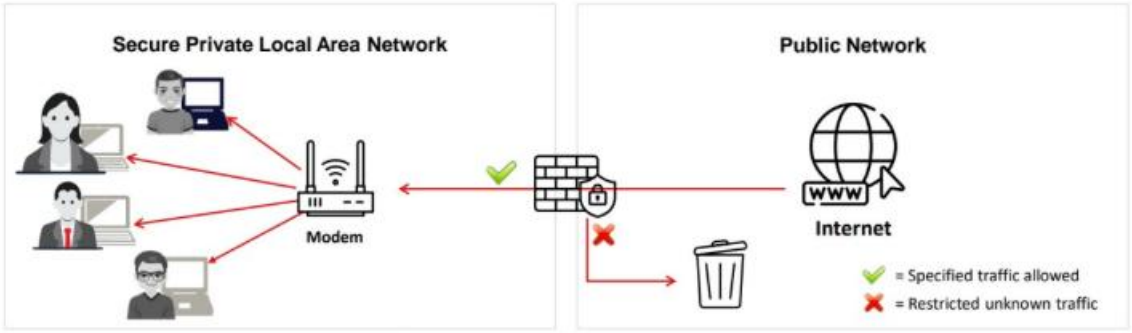
        - Deep packet inspection
        - Intrusion prevention
        - Application awareness
        - Malware protection

5. **Software Firewall**
    - Installed on individual devices (like Windows Defender Firewall).
    - Protects that single device.
6. **Hardware Firewall**
    - A physical device placed between your network and the internet.
    - Common in business environments.

Why Firewalls Are Important:

- Prevent **unauthorized access**
- Protect against **malware and cyberattacks**
- Enforce **security policies**
- Log and audit network activity

Working Of Firewall -:



What is IDS (Intrusion Detection System)?

An **Intrusion Detection System (IDS)** is a security tool that monitors and analyzes network or system activity to detect **unauthorized access**, **malicious behavior**, or **policy violations**.

🎯 Uses/Objectives of IDS :

1. Monitors network or system traffic for suspicious activities.
2. Detects potential intrusions and security breaches.
3. Alerts administrators about detected threats.
4. Identifies malware, exploits, and attack patterns.
5. Helps in early detection of cyber attacks.
6. Assists in forensic analysis after an incident.
7. Enforces organizational security policies.
8. Complements firewalls and other security tools.
9. Tracks user activity to detect insider threats.
10. Logs security events for auditing and compliance.

Types Of IDS--:

NIDS (Network-Based Intrusion Detection System)

Definition:
Monitors network traffic across a segment to detect suspicious activity.

Key Points:

- Placed at strategic points in the network (e.g., near the gateway or DMZ).
- Analyzes all incoming and outgoing traffic.
- Best for detecting large-scale or external attacks.

Examples: Snort, Suricata

Use Cases:

- Detecting port scans, DoS attacks, and malware traffic.
- Monitoring real-time data flow in networks.

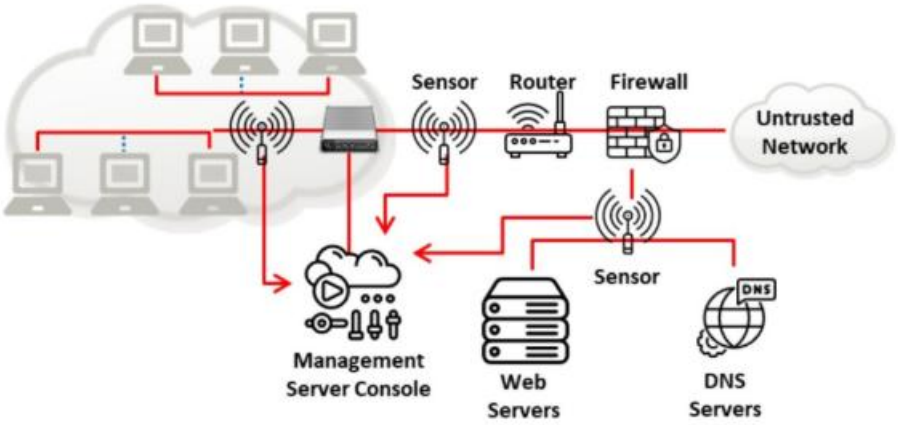


Fig-:Network-Based IDS

💻 HIDS (Host-Based Intrusion Detection System)

Definition:
Monitors the internal behavior of a specific host or endpoint.

Key Points:

- Installed directly on individual machines (servers, workstations).
- Analyzes system logs, file integrity, and user behavior.
- Best for detecting insider threats or local compromises.

Examples: OSSEC, Tripwire

Use Cases:

- Detecting unauthorized file changes or logins.
- Protecting critical servers from local exploits.

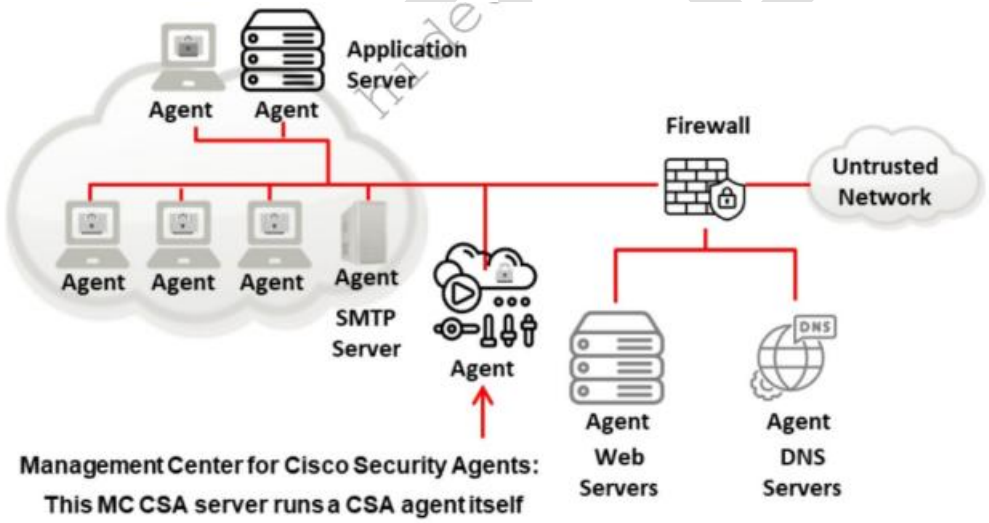


Fig- Host-Based IDS

How an IDS Detects an Intrusion

An **Intrusion Detection System (IDS)** detects intrusions using one or more of the following techniques:

---

1. Signature-Based Detection

- Compares traffic or system activity against a database of known attack patterns (signatures).
- **Example**: Detects a known malware hash or a specific exploit packet.

✅ **Strength**: Accurate for known threats
❌ **Weakness**: Can't detect new or unknown attacks

---

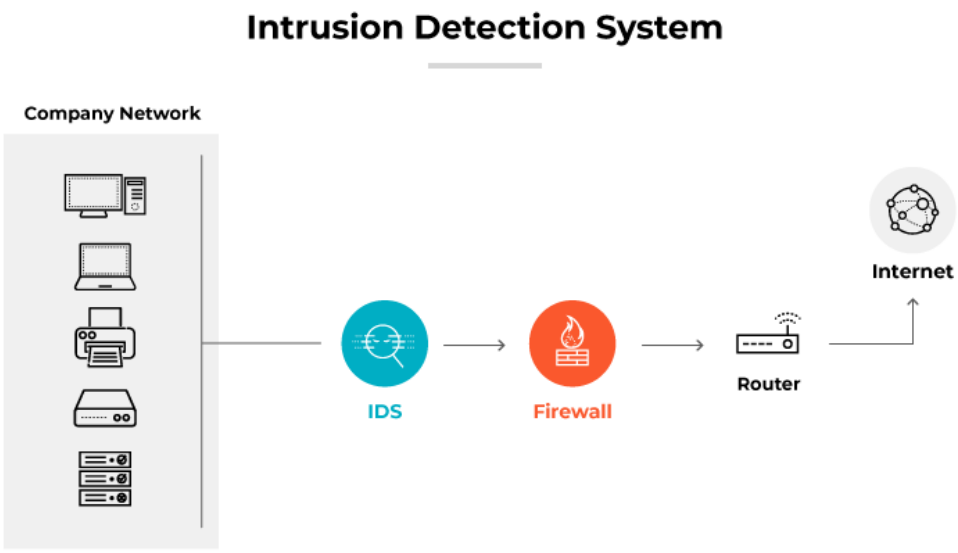
2. Anomaly-Based Detection

- Learns normal behavior (baseline) of a system or network, then flags deviations.
- **Example**: Sudden spike in outbound traffic from a server at midnight.

✅ **Strength**: Detects unknown or zero-day attacks
❌ **Weakness**: May produce false positives

3.Protocol Anomaly Detection (PAD) in IDS

- **Protocol Anomaly Detection** is a type of **anomaly-based intrusion detection** that identifies intrusions by **detecting deviations from normal behavior defined by protocol standards** (like TCP, HTTP, DNS, etc.).



Types of IDS Alerts

These are categorized based on how accurately the IDS identifies malicious activity:

---

1. True Positive (TP)

🔒 **Attack is present, and the IDS detects it.**

- ✅ Correct detection.
- Example: An actual SQL injection attack occurs, and the IDS raises an alert.

---

2. True Negative (TN)

✅ **No attack is present, and the IDS does not raise an alert.**

- ✅ Correct behavior.
- Example: Normal user traffic occurs, and the IDS remains silent.

3. False Positive (FP)

⚠️ **No attack is present, but the IDS raises an alert.**

- ❌ Incorrect detection.
- Example: A regular user uploads a file, and the IDS flags it as malware mistakenly.

---

4. False Negative (FN)

🚨 **Attack is present, but the IDS fails to detect it.**

- ❌ Dangerous because the attack goes unnoticed.
- Example: A new type of ransomware hits the network, but the IDS doesn't recognize it.

What is IPS (Intrusion Prevention System)-:

🔐 **Definition:**

An **IPS** sits **in-line** (directly in the traffic path) between network devices and **monitors traffic continuously**. When it detects **suspicious activity**, it can **automatically take action** to stop the threat, such as:

- Dropping malicious packets
- Blocking traffic from an IP address
- Resetting connections
- Alerting administrators

🧠 **How It Works:**

1. **Traffic Inspection**: It analyzes network traffic using deep packet inspection (DPI).
2. **Signature Matching**: Compares traffic patterns to known attack signatures (like viruses, exploits).
3. **Behavior Analysis**: Detects anomalies (e.g., too many requests from one IP).
4. **Prevention**: Automatically takes action before the attack can success

Types Of Intrusion Prevention System -:

1. Network-based IPS (NIPS)

- **Location**: Deployed at key points in the network (e.g., behind a firewall).
- **Purpose**: Monitors all traffic between hosts on the network.
- **Detection Scope**: Protects the entire network.
- **Examples**: Cisco Firepower, Suricata, Snort (IPS mode).

✅ **Best for**: Detecting and blocking threats that move across the network.

🔹 **2. Host-based IPS (HIPS)**

- **Location**: Installed on individual hosts (servers, desktops, laptops).
- **Purpose**: Monitors system calls, logs, application behavior, and file access.
- **Detection Scope**: Protects only the local system it is installed on.

- **Examples**: OSSEC (with HIPS modules), Symantec Endpoint Protection.

✅ **Best for**: Detecting exploits like privilege escalation or file modification on a specific machine.

🔹 **3. Wireless IPS (WIPS)**

- **Location**: Monitors wireless network traffic and infrastructure.
- **Purpose**: Detects and blocks rogue devices, unauthorized access points, and wireless attacks (e.g., deauthentication attacks).
- **Examples**: Aruba WIPS, Cisco Meraki AirMarshal.

✅ **Best for**: Securing wireless environments like corporate Wi-Fi or public access areas.

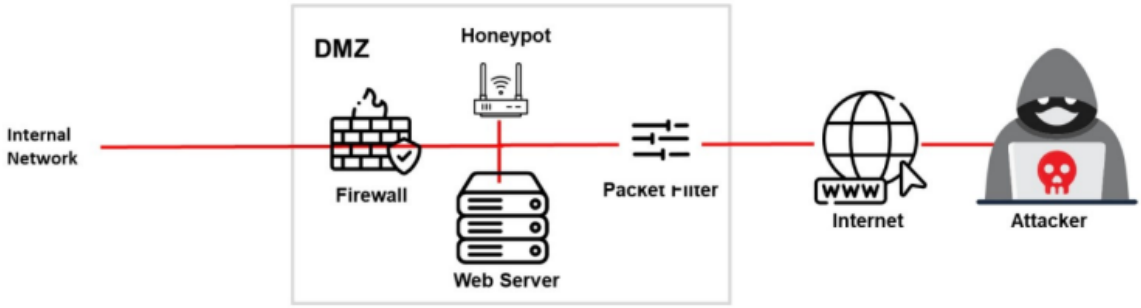
🔹 **4. Network Behavior Analysis (NBA) IPS**

- **Location**: Typically part of advanced IPS or security appliances.
- **Purpose**: Monitors traffic to detect unusual patterns (e.g., large data transfers, DDoS).
- **Detection Method**: Uses behavior modeling and anomaly detection.
- **Examples**: Darktrace, Stealthwatch (Cisco).

✅ **Best for**: Identifying zero-day threats, insider threats, and large-scale attacks.

What is Honeypot?

A **honeypot** is a **cybersecurity mechanism** that is deliberately designed to **attract attackers**. It simulates a vulnerable system, application, or network service, so that attackers interact with it — allowing defenders to **monitor**, **detect**, and **analyze** malicious activity without risking real systems.



🎯 Main Goals of a Honeypot

1. **Detection** – Detect unauthorized access and malicious activity.
2. **Diversion** – Distract attackers from real targets.
3. **Analysis** – Study attacker behavior, tools, and techniques.
4. **Prevention** – Improve defenses based on insights gathered.

🧩 How a Honeypot Works

1. It mimics a legitimate target like a web server, database, or IoT device.
2. It's placed in a **controlled environment** (isolated from real systems).
3. When an attacker interacts with it (scanning, exploitation, etc.), it **logs** and **records** all actions.
4. The security team uses this data for investigation and threat intelligence.

Types of HoneyPots-:

- **Based on Level of Interaction**

1. **Low-interaction honeypots** – Simulate limited services with minimal risk.
2. **Medium-interaction honeypots** – Provide more interaction without full OS.
3. **High-interaction honeypots** – Emulate real systems for in-depth attacker engagement.

- **Based on Purpose**

4. **Research honeypots** – Used to study attacker behavior and collect threat intelligence.
5. **Production honeypots** – Deployed in live networks to detect and deflect attacks.

- **Based on Location**

6. **External honeypots** – Placed outside the firewall to catch external attackers.
7. **Internal honeypots** – Placed inside the network to detect insider threats or lateral movement.

- **Based on Technology**

8. **Malware honeypots** – Designed to capture and analyze malware.
9. **Spam honeypots** – Mimic open relays to attract spam emails.

10. **Database honeypots** – Simulate vulnerable databases to catch SQL injection and data exfiltration attempts.

11. **Web application honeypots** – Mimic websites or CMS systems to detect web-based attacks.

12. **Industrial Control System (ICS) honeypots** – Simulate SCADA/ICS environments to detect threats targeting critical infrastructure.

- **Honeynet**

13. **Honeynet** – A network of multiple interconnected honeypots for broader attack simulation and analysis.

Snort

Snort is an open-source **network intrusion detection system (NIDS)** and intrusion prevention system (IPS) developed by **Cisco Systems**. It's one of the most widely used tools for real-time traffic analysis and packet logging on IP networks.

Uses/Objectives of Snort-:

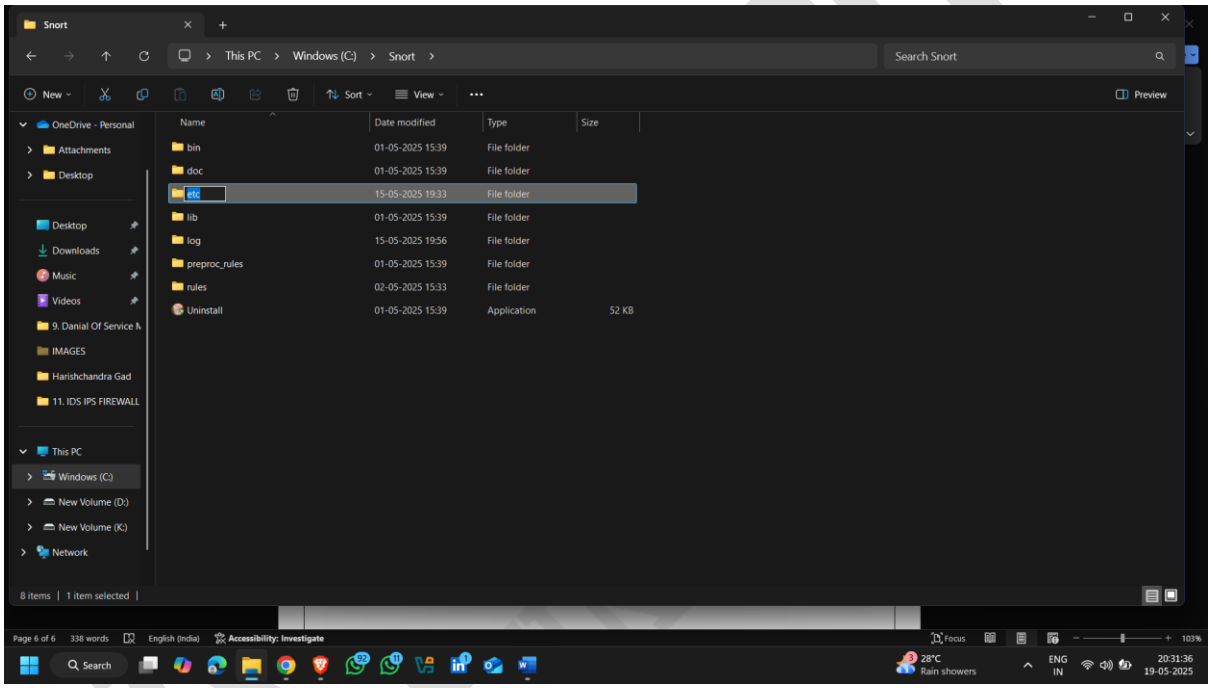
- Detects network intrusions in real-time.
- Performs deep packet inspection (DPI).
- Monitors traffic for suspicious behavior.
- Logs packets for later analysis.
- Prevents attacks when used in IPS mode.
- Detects port scans and probes.
- Identifies malware and exploit attempts.
- Enforces security policies on networks.
- Supports custom rule creation for threat detection.
- Assists in forensic and incident response analysis.

Snort Configuration

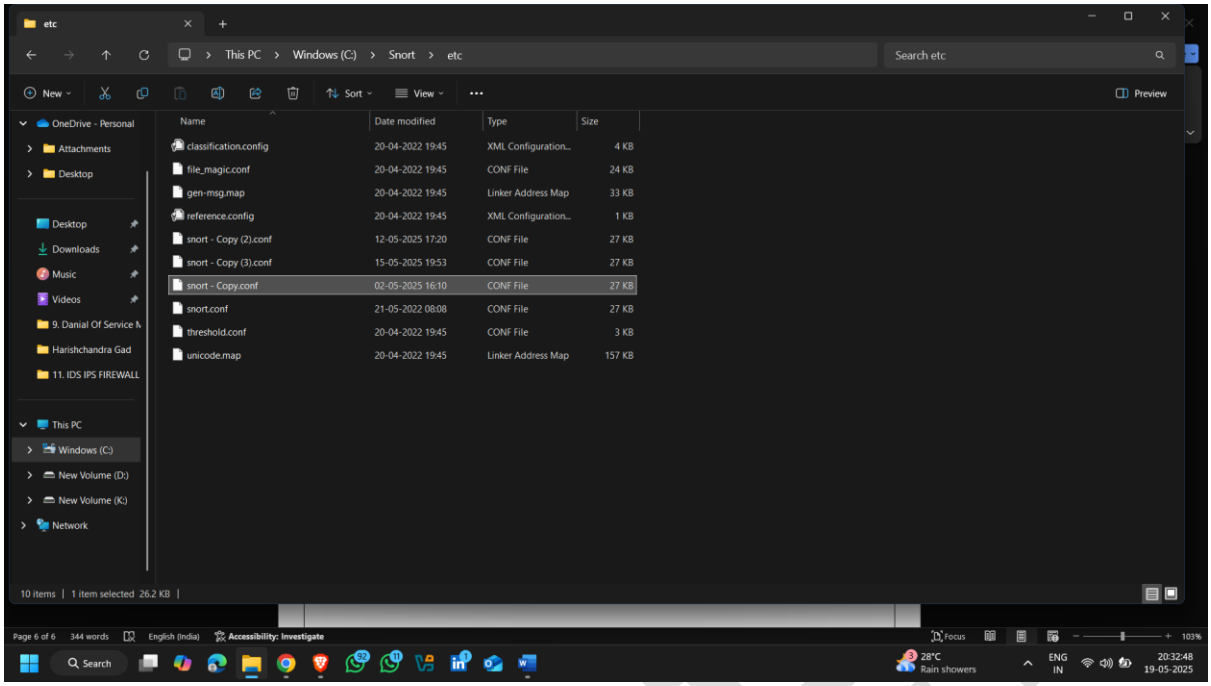
Download Link –:https://www.snort.org/downloads

How to use it -:

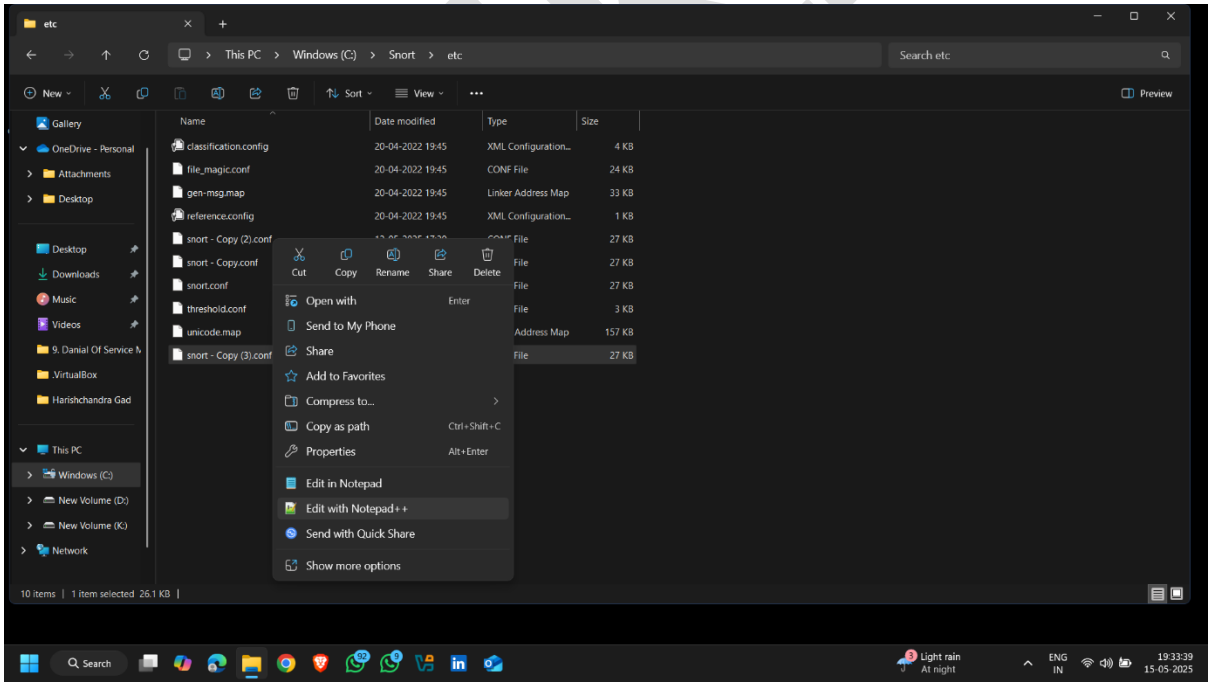
- After installation snort , go to the snort location and open etc folder



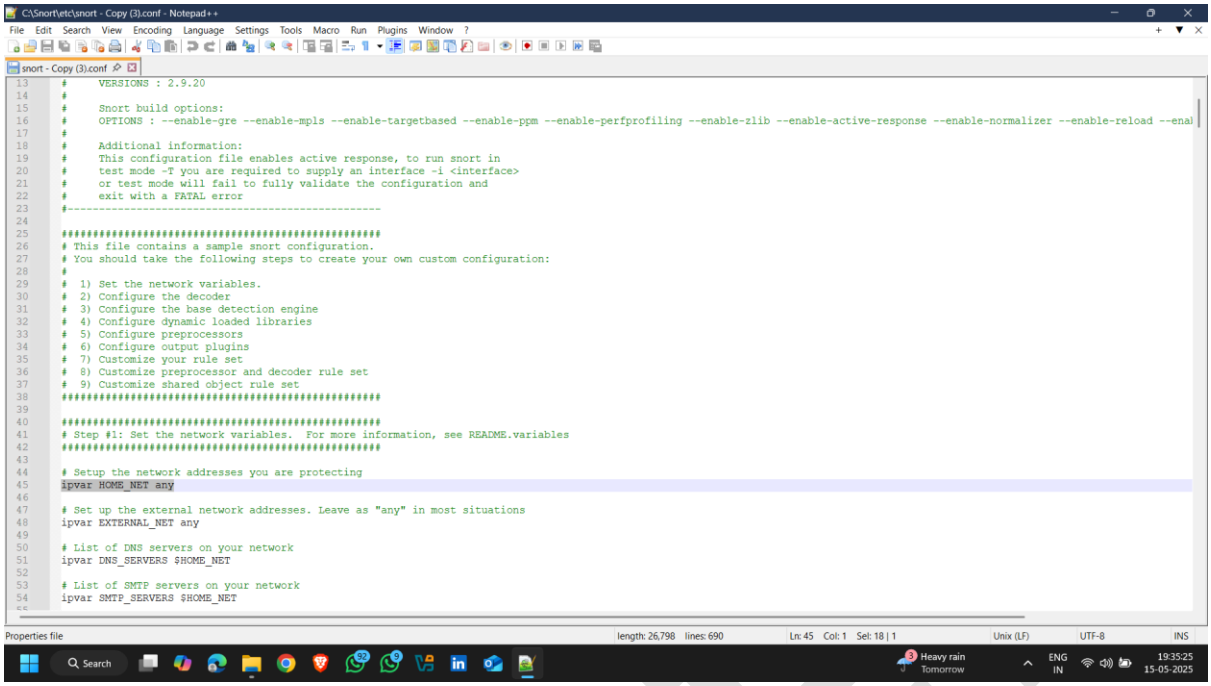
- Copy snort.exe file and paste it



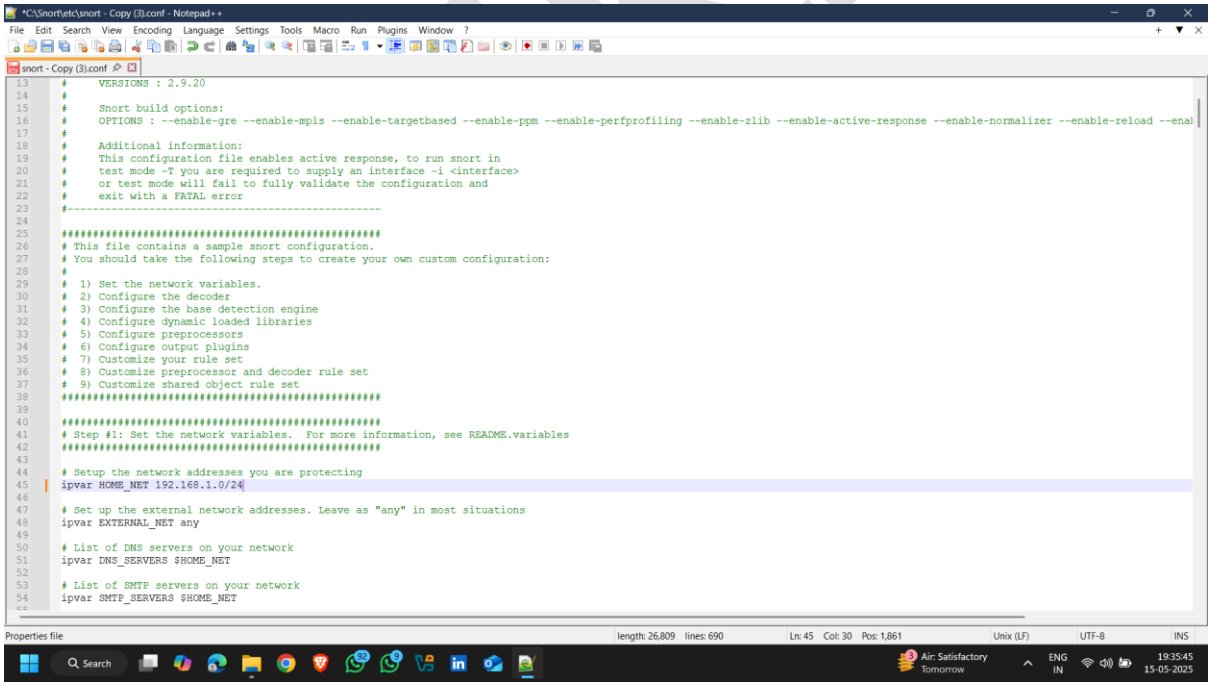
- Now , open copied file in **Notepad++**



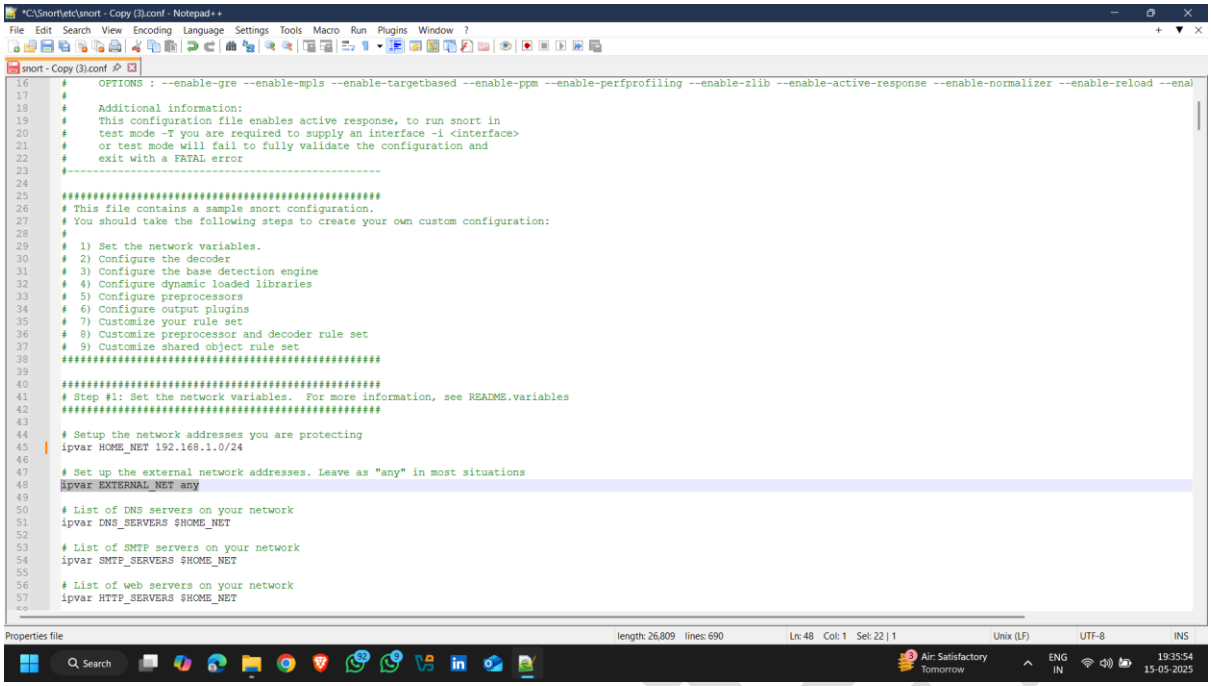
- Go to the **line number 45**



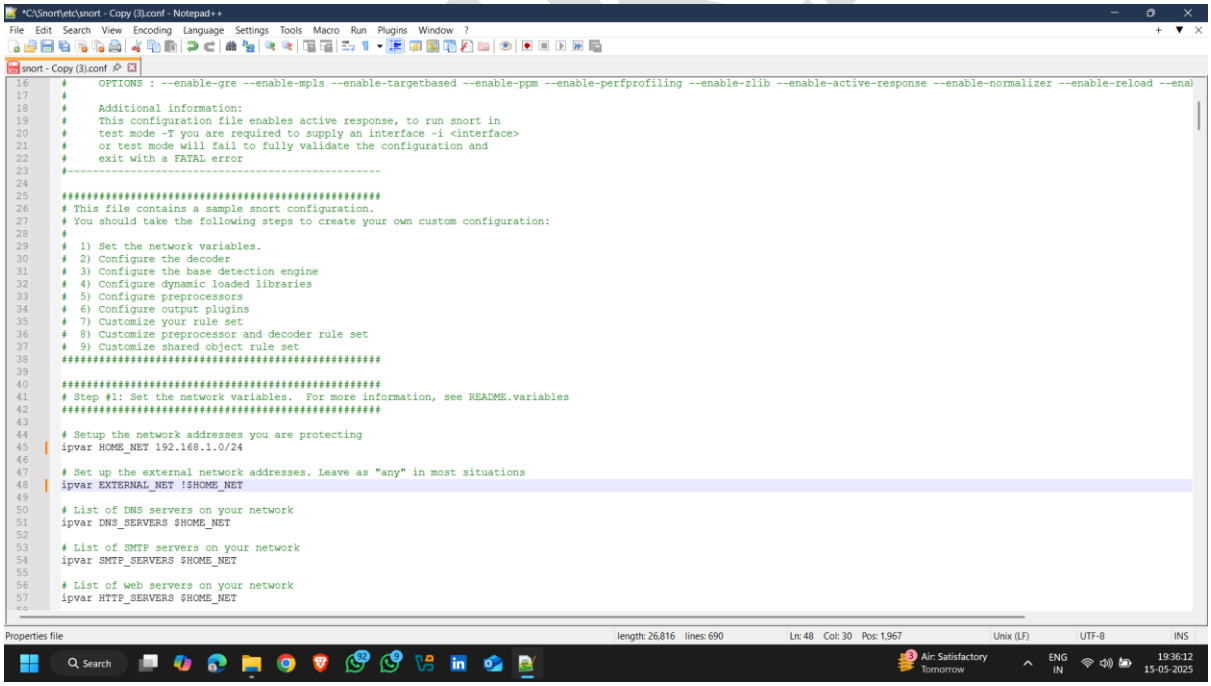
- And replace **Any** word to **the ip range**



- Go to the **line number 48**

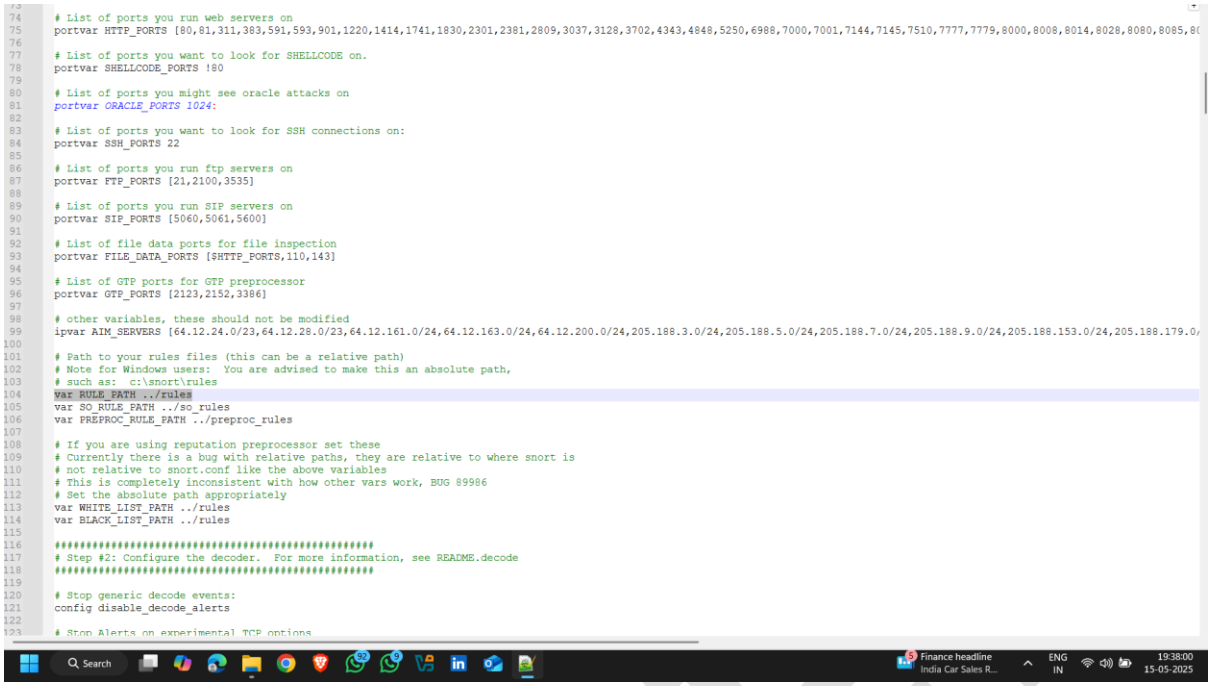


- Replace **Any** to the **!$HOME_NET**



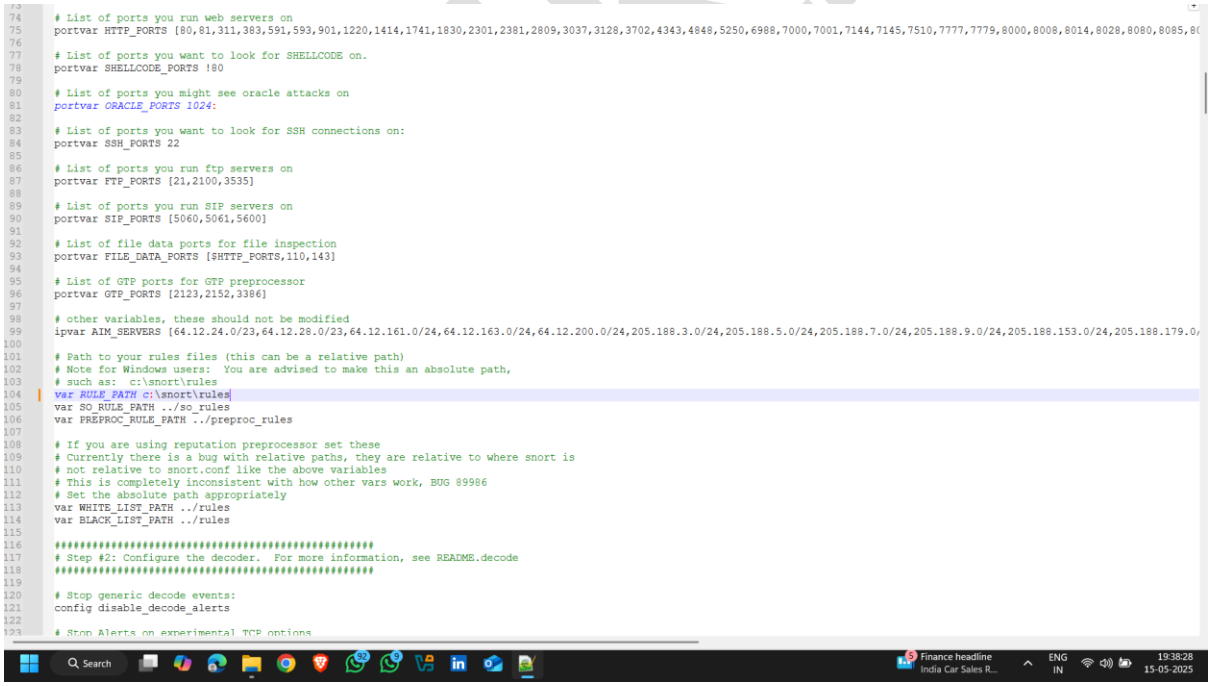
- Go to the line number 104

- Set rules folder location



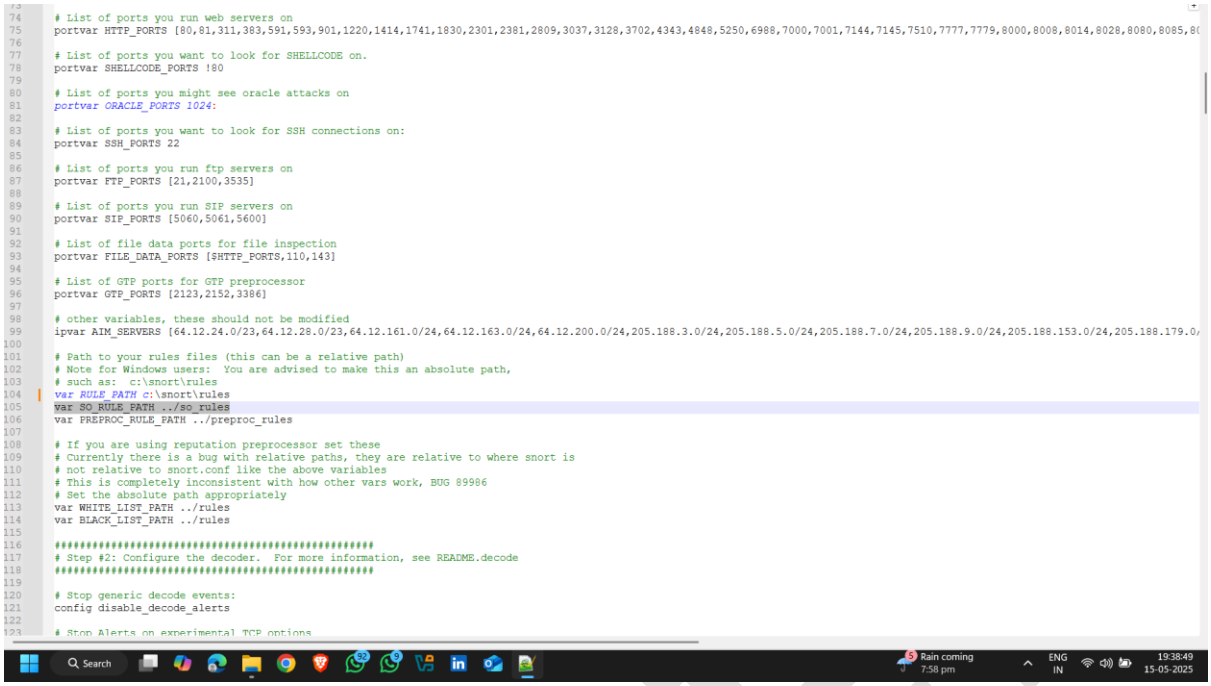
```
# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,4848,5250,6988,7000,7001,7144,7145,7510,7777,7779,8000,8008,8014,8028,8080,8085,8

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/

# 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 ../rules
var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH ../preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
```

- C:\\Snort\rules

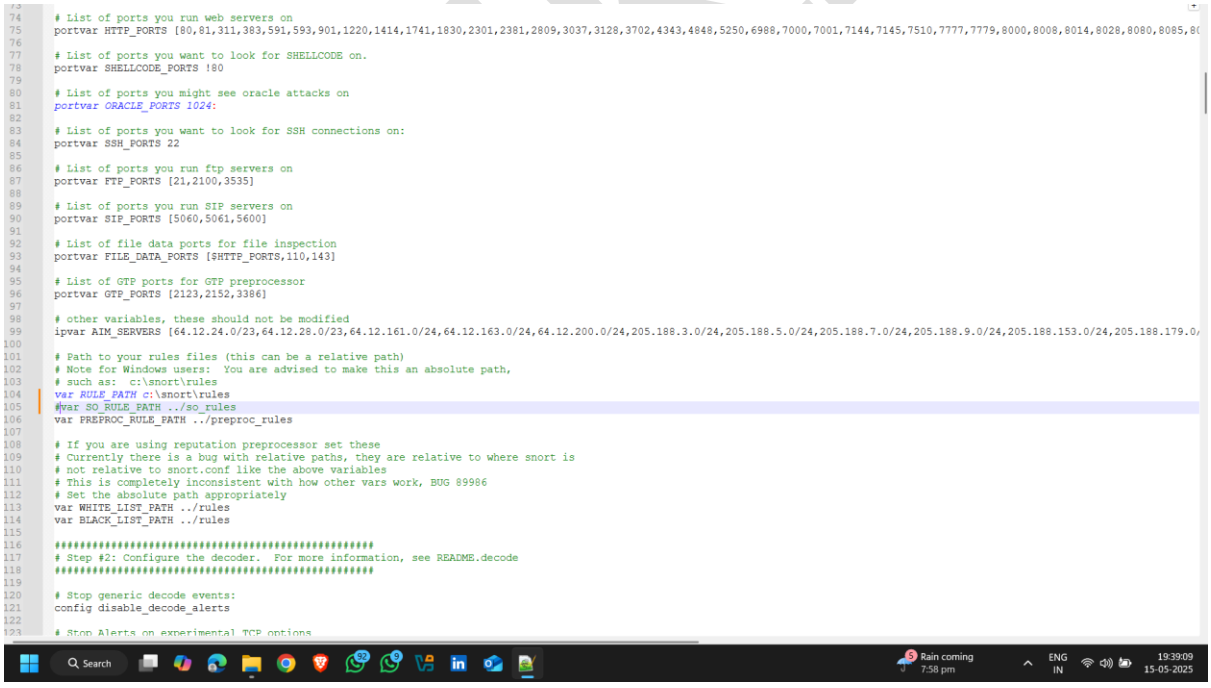


```
# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,4848,5250,6988,7000,7001,7144,7145,7510,7777,7779,8000,8008,8014,8028,8080,8085,8

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/

# 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

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
```

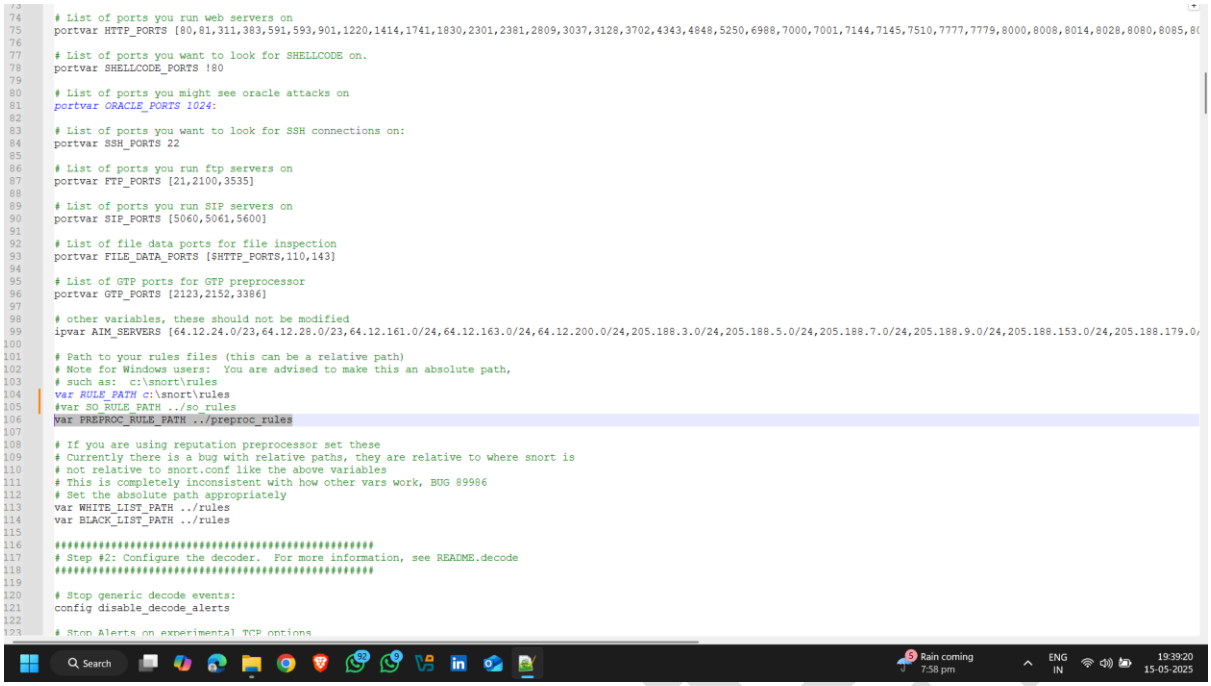
- Go to the **line number 105**



- Add # on the front of the line number 105

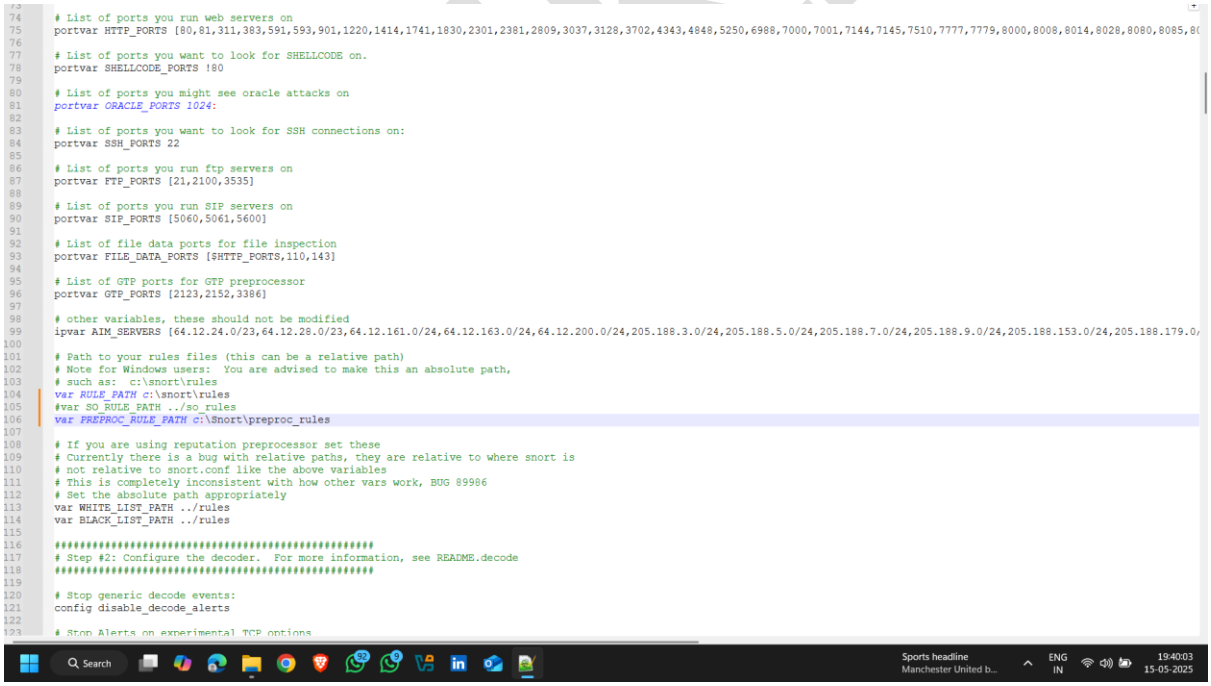


- Go to **the line number 106 and set preproc_rules**



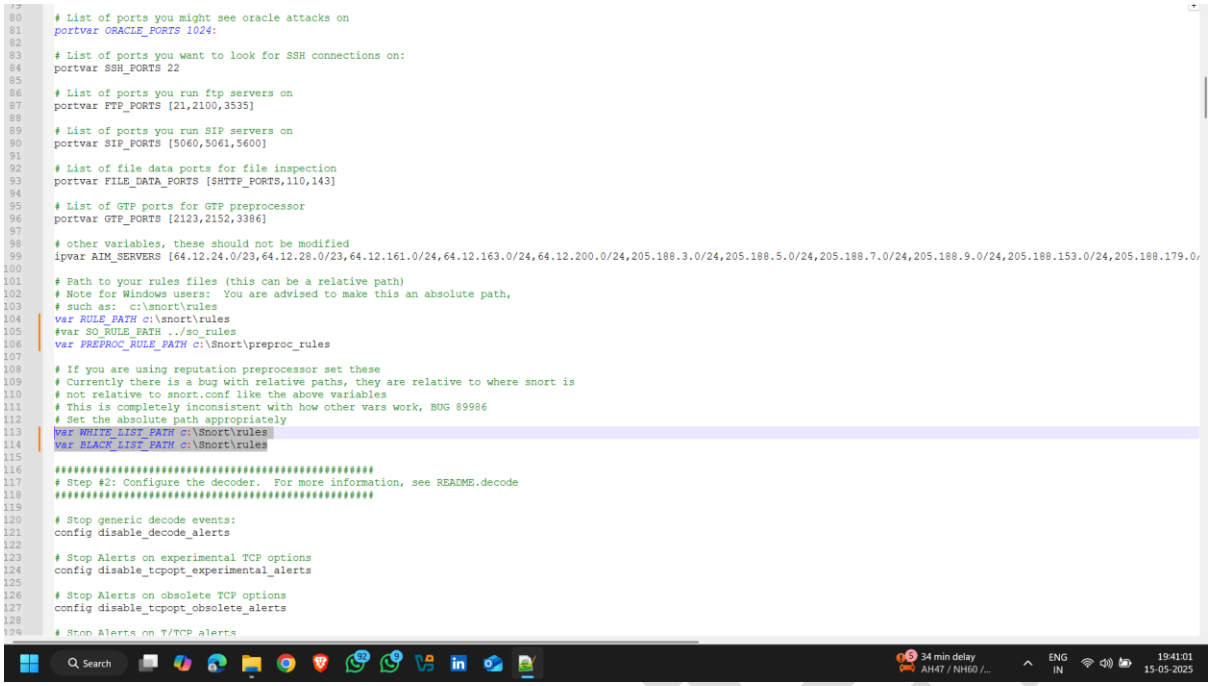
```
# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,4848,5250,6988,7000,7001,7144,7145,7510,7777,7779,8000,8008,8014,8028,8080,8085,8

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/

# 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

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
```


- C:\Snort\preproc_rules



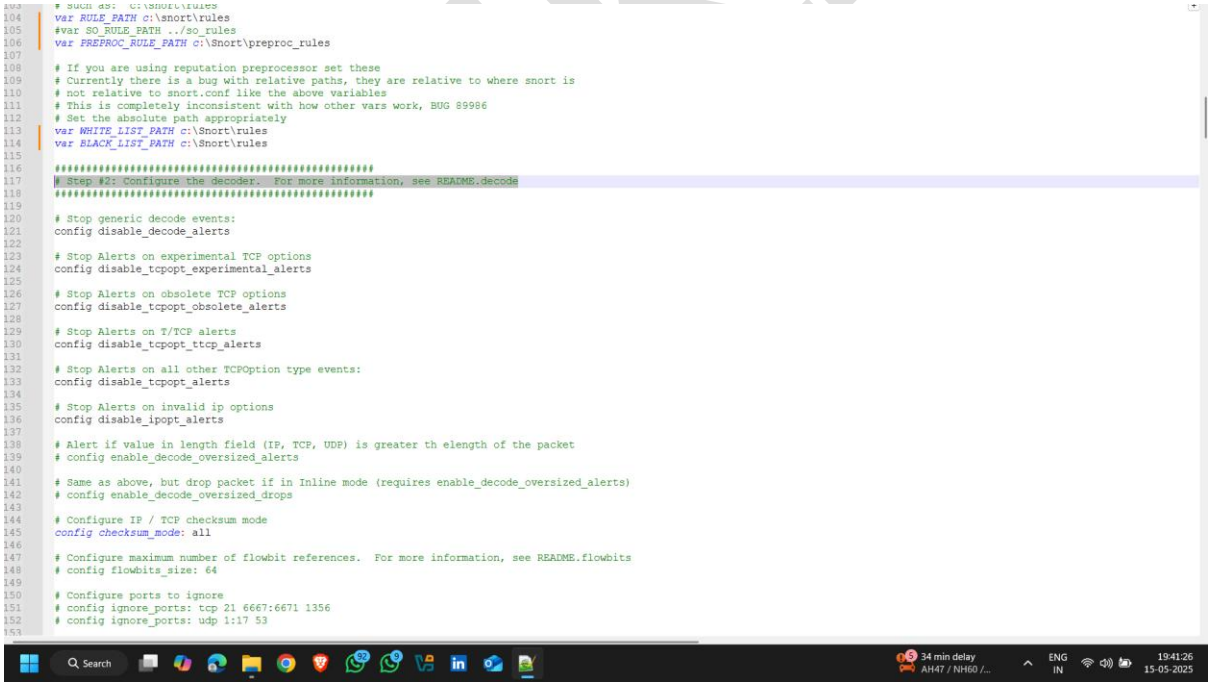
```
# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,4848,5250,6988,7000,7001,7144,7145,7510,7777,7779,8000,8008,8014,8028,8080,8085,8

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/

# 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 c:\Snort\preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
```


- go to the line number 113 and 114
- and add rules file path/location



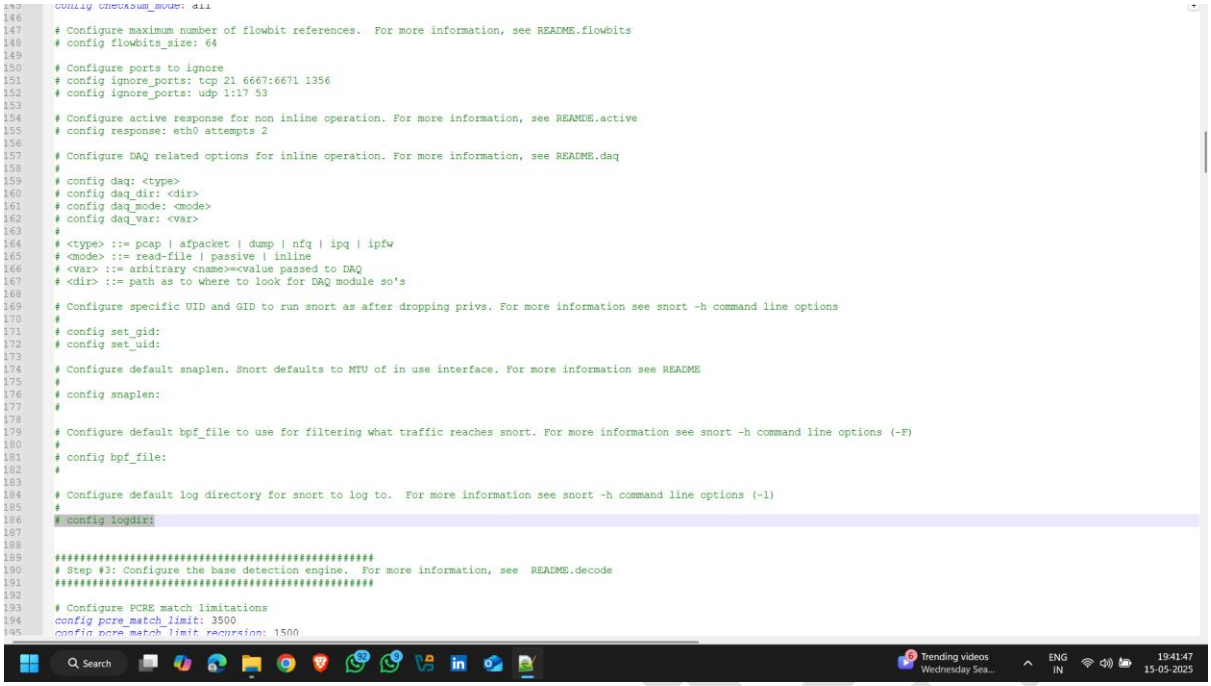
```
# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/

# 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 c:\Snort\preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH c:\Snort\rules
var BLACK_LIST_PATH c:\Snort\rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts

# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts

# Stop Alerts on T/TCP alerts
```

- Step 2

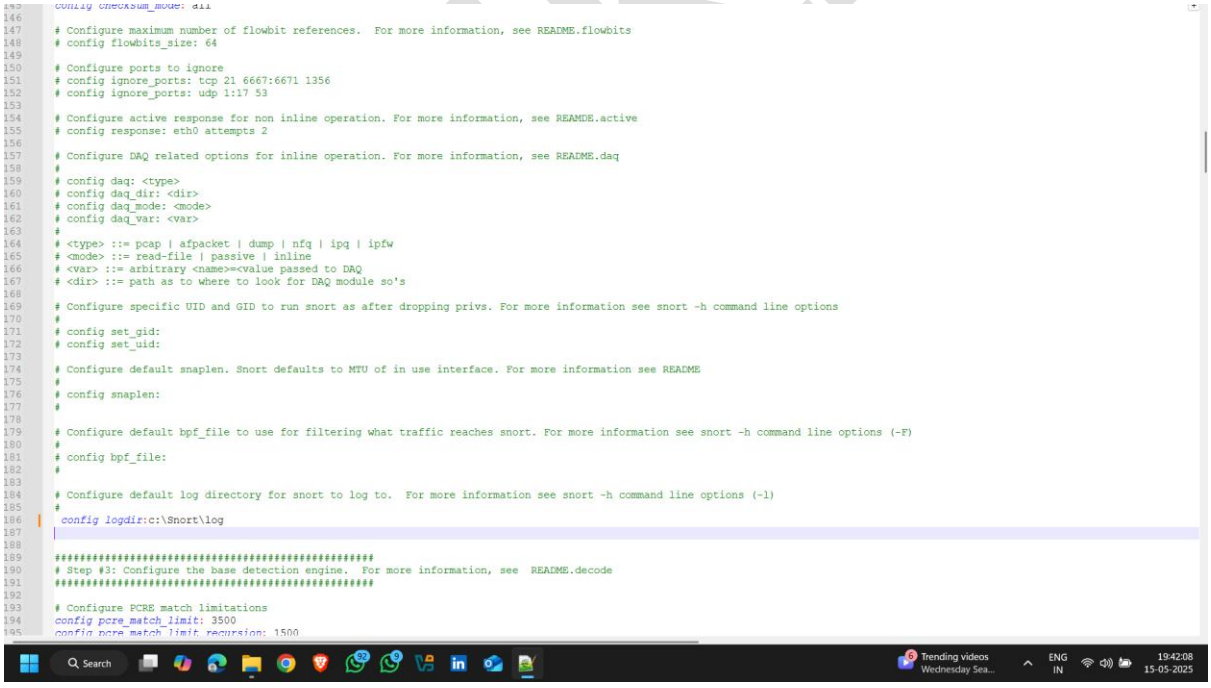


```
# such as:  c:\snort\rules
var RULE_PATH c:\snort\rules
#var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH c:\Snort\preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH c:\Snort\rules
var BLACK_LIST_PATH c:\Snort\rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts

# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts

# Stop Alerts on T/TCP alerts
config disable_tcpopt_ttcp_alerts

# Stop Alerts on all other TCPOption type events:
config disable_tcpopt_alerts

# Stop Alerts on invalid ip options
config disable_ipopt_alerts

# Alert if value in length field (IP, TCP, UDP) is greater th elength of the packet
# config enable_decode_oversized_alerts

# Same as above, but drop packet if in Inline mode (requires enable_decode_oversized_alerts)
# config enable_decode_oversized_drops

# Configure IP / TCP checksum mode
config checksum_mode: all

# Configure maximum number of flowbit references.  For more information, see README.flowbits
# config flowbits_size: 64

# Configure ports to ignore
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53
```

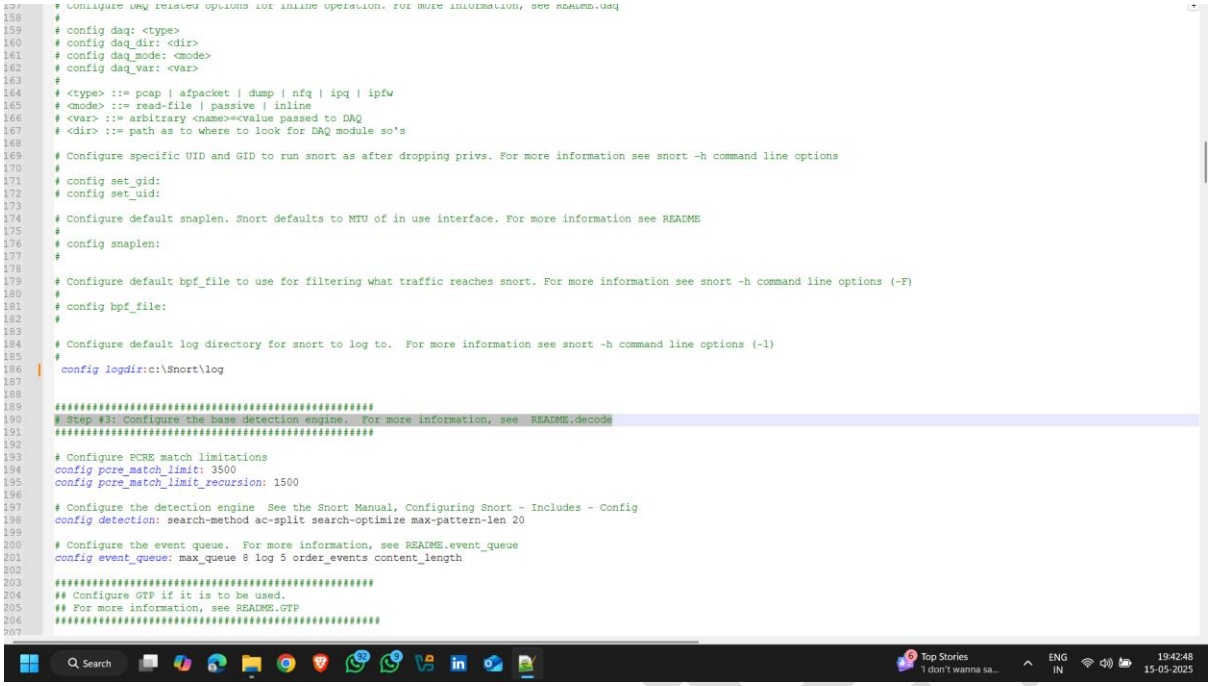
- go to the line number 186



- remove hash

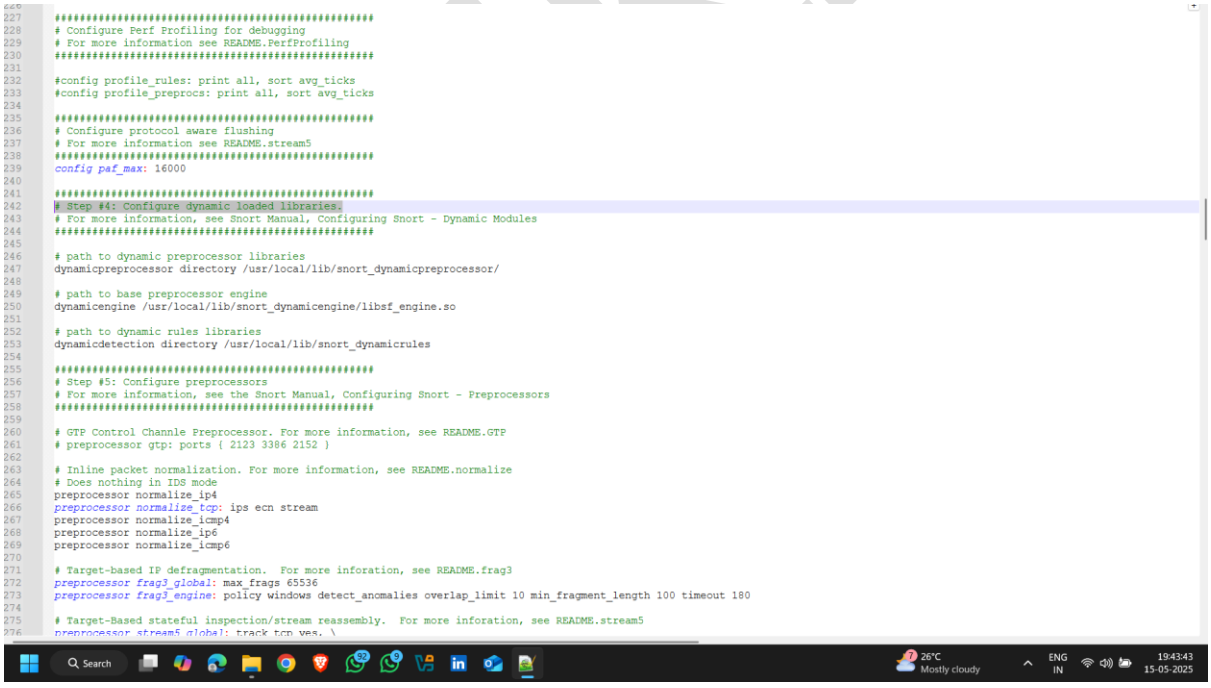


- Step 3
- Don’t do anything , no need to change



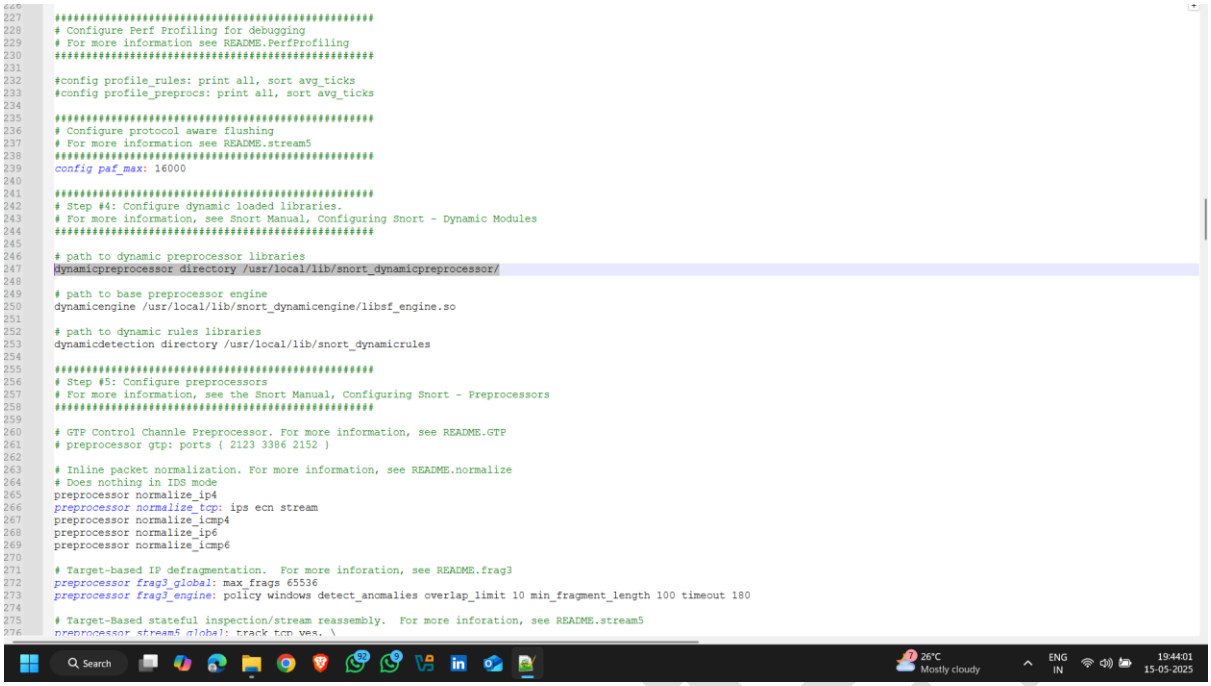
```
# Configure DAQ related options for inline operation. For more information, see README.daq
#
# config daq: <type>
# config daq_dir: <dir>
# config daq_mode: <mode>
# config daq_var: <var>
#
# <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw
# <mode> ::= read-file | passive | inline
# <var> ::= arbitrary <name>=<value passed to DAQ
# <dir> ::= path as to where to look for DAQ module so's

# Configure specific UID and GID to run snort as after dropping privs. For more information see snort -h command line options
#
# config set_gid:
# config set_uid:

# Configure default snaplen. Snort defaults to MTU of in use interface. For more information see README
#
# config snaplen:
#

# Configure default bpf_file to use for filtering what traffic reaches snort. For more information see snort -h command line options (-F)
#
# config bpf_file:
#

# Configure default log directory for snort to log to.  For more information see snort -h command line options (-l)
#
 config logdir:c:\Snort\log


##################################################
# Step #3: Configure the base detection engine.  For more information, see  README.decode
##################################################

# Configure PCRE match limitations
config pcre_match_limit: 3500
config pcre_match_limit_recursion: 1500

# Configure the detection engine  See the Snort Manual, Configuring Snort - Includes - Config
config detection: search-method ac-split search-optimize max-pattern-len 20

# Configure the event queue.  For more information, see README.event_queue
config event_queue: max_queue 8 log 5 order_events content_length

##################################################
## Configure GTP if it is to be used.
## For more information, see README.GTP
##################################################
```

- Step 4



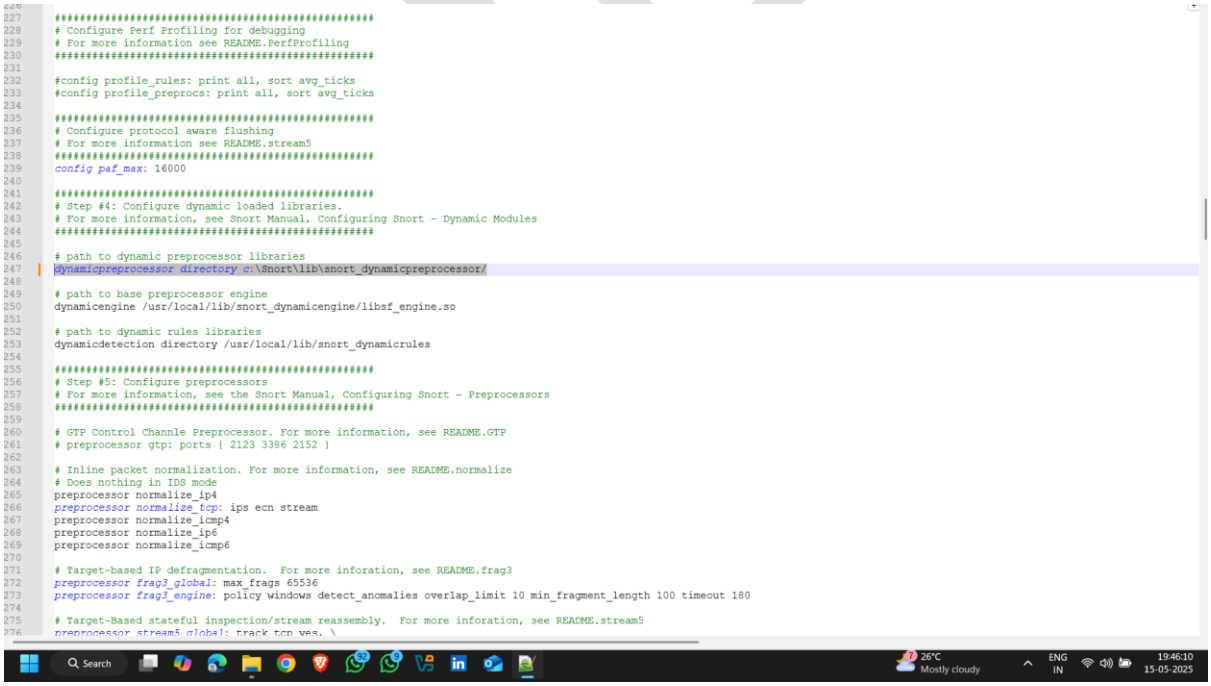
```
##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
```

- Go to the line number 247



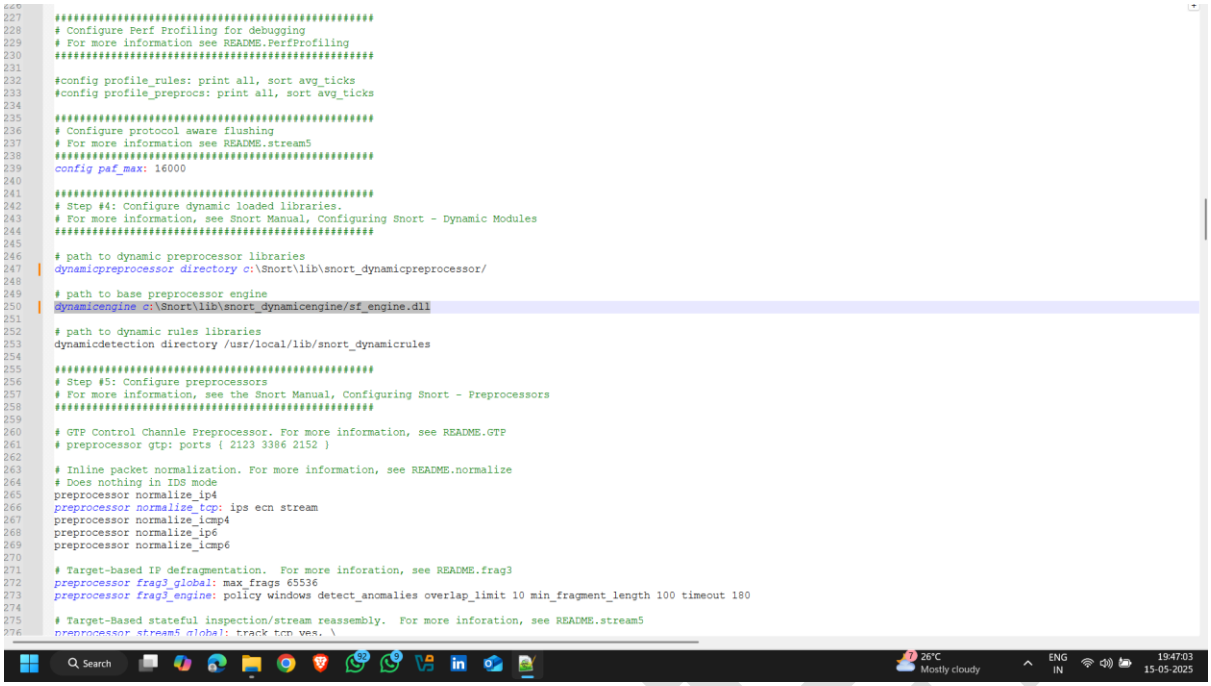
```
##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
```


- Set snort_dynamicpreprocessor file path/location -: c:\Snort\lib\snort_preprocessor



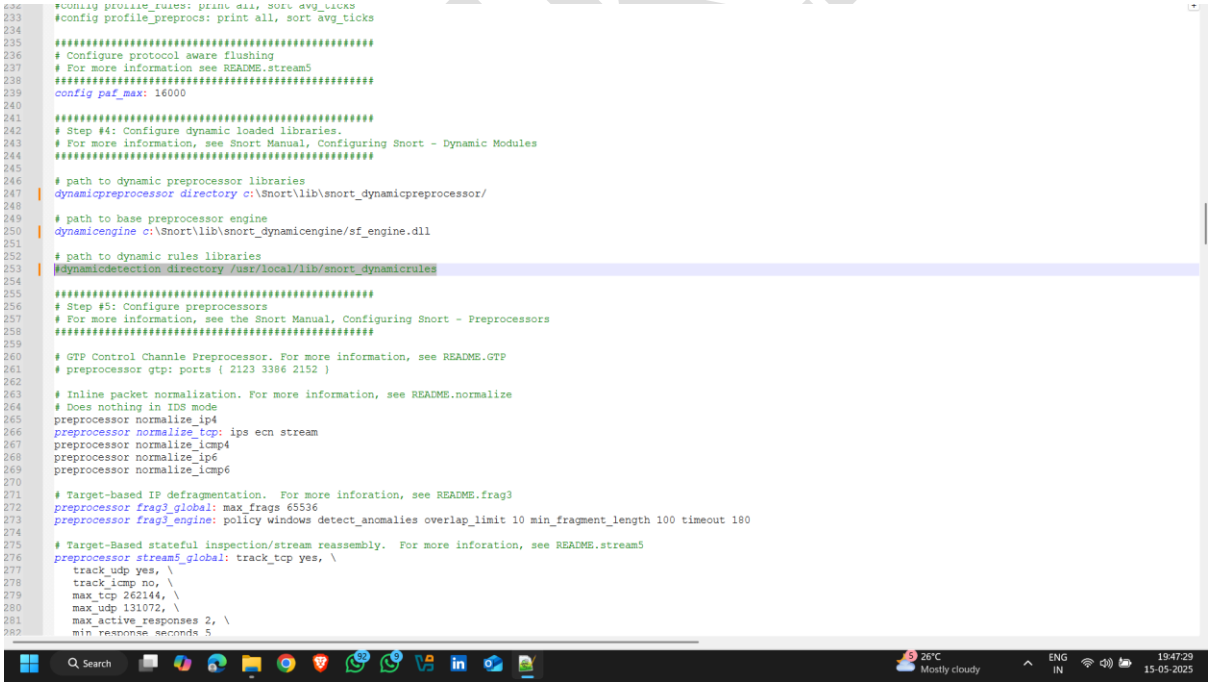
```
##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory c:\Snort\lib\snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
```


- Go to the line number number 250
- Set sf_engine.dll file location -: c:\Snort\lib\snort_dynamicengine/sf_engine.dll



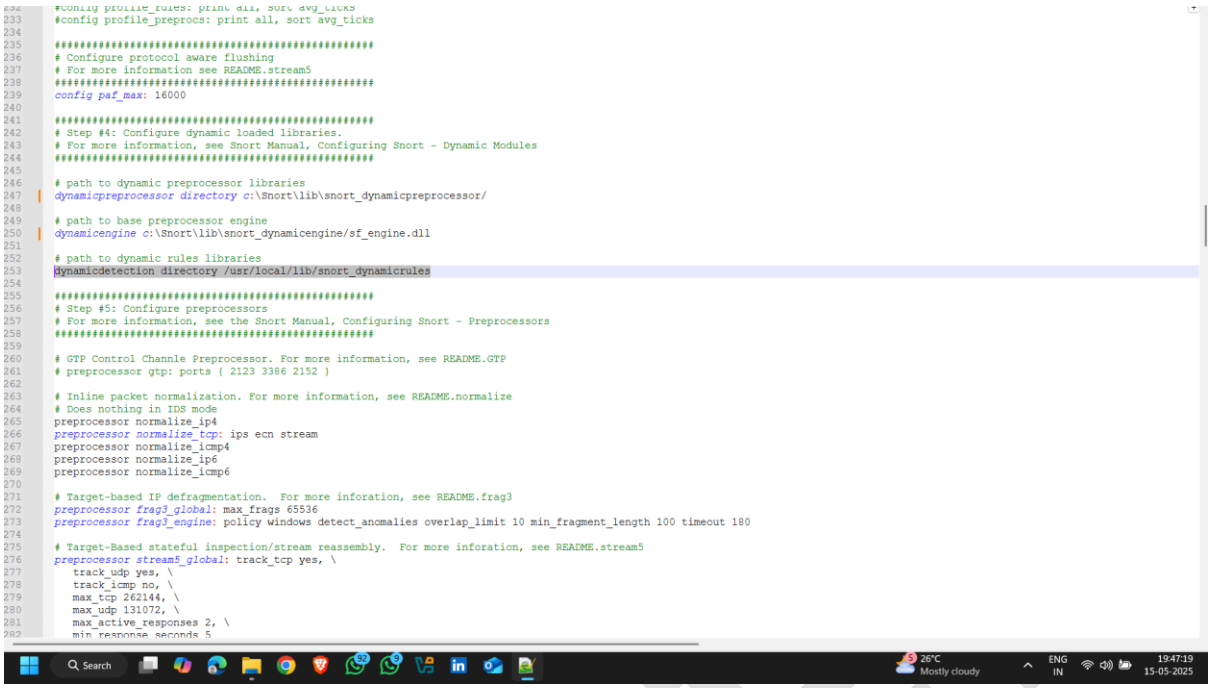
```
##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory c:\Snort\lib\snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine c:\Snort\lib\snort_dynamicengine/sf_engine.dll

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
```


- Go to the line number 253

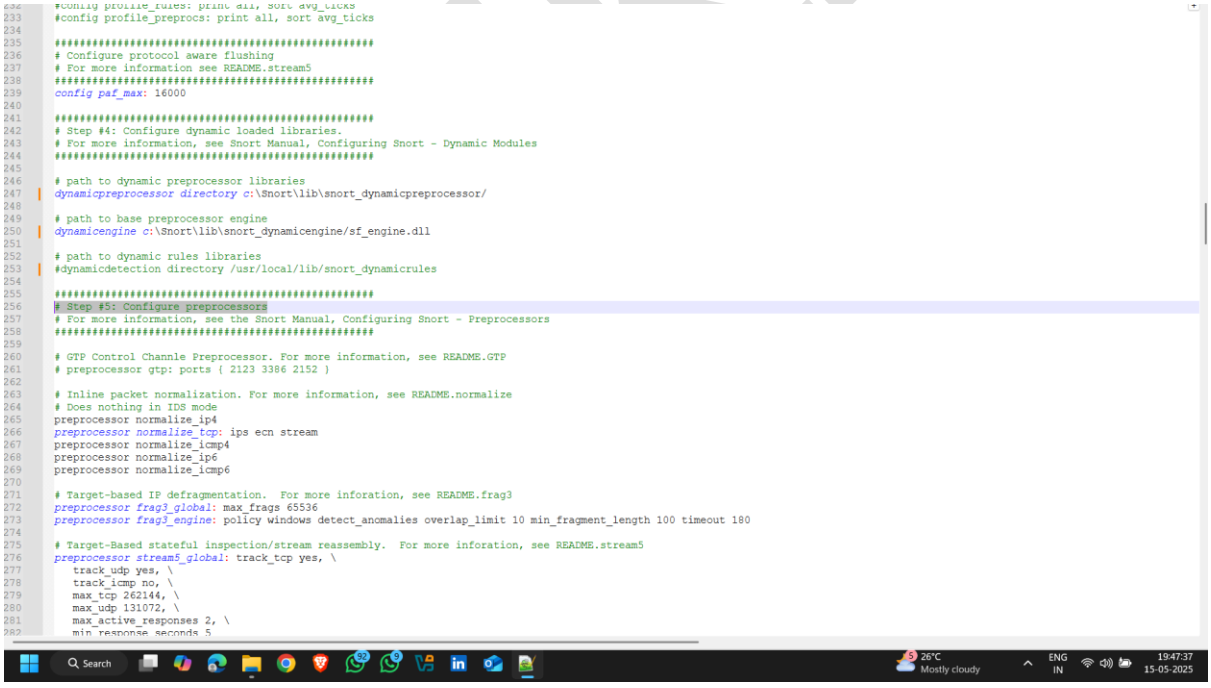


```
#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory c:\Snort\lib\snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine c:\Snort\lib\snort_dynamicengine/sf_engine.dll

# path to dynamic rules libraries
#dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
  track_udp yes, \
  track_icmp no, \
  max_tcp 262144, \
  max_udp 131072, \
  max_active_responses 2, \
  min_response_seconds 5
```


- Remove # on front of the line number 253



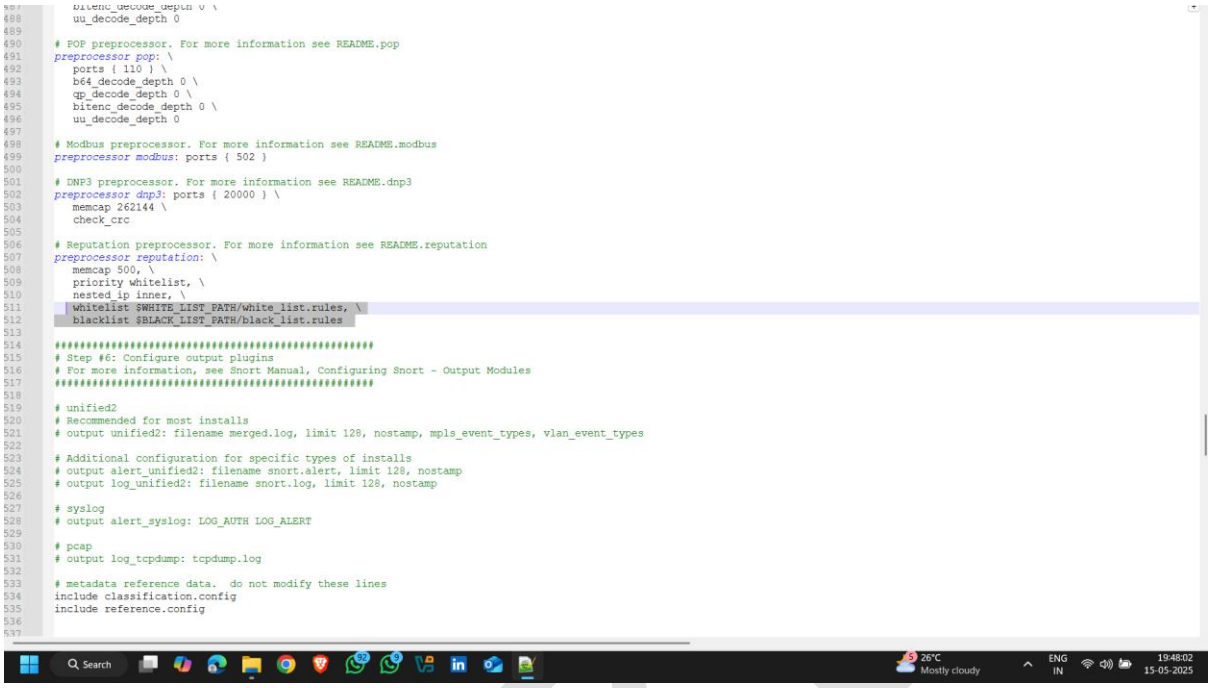
- Step 5



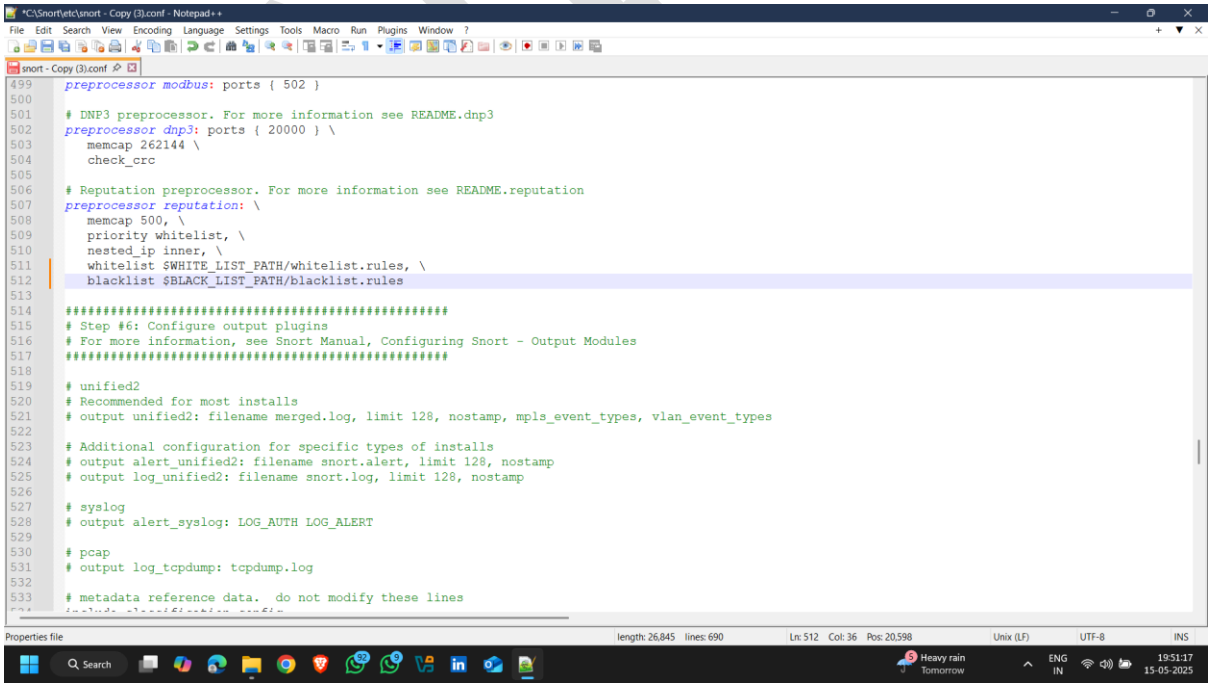
- Go to the line number 511 and 512

Note-: before changing on the line number 511 and 512 , firstly go to the **snort >>rules folder and there is file in folder i.e**

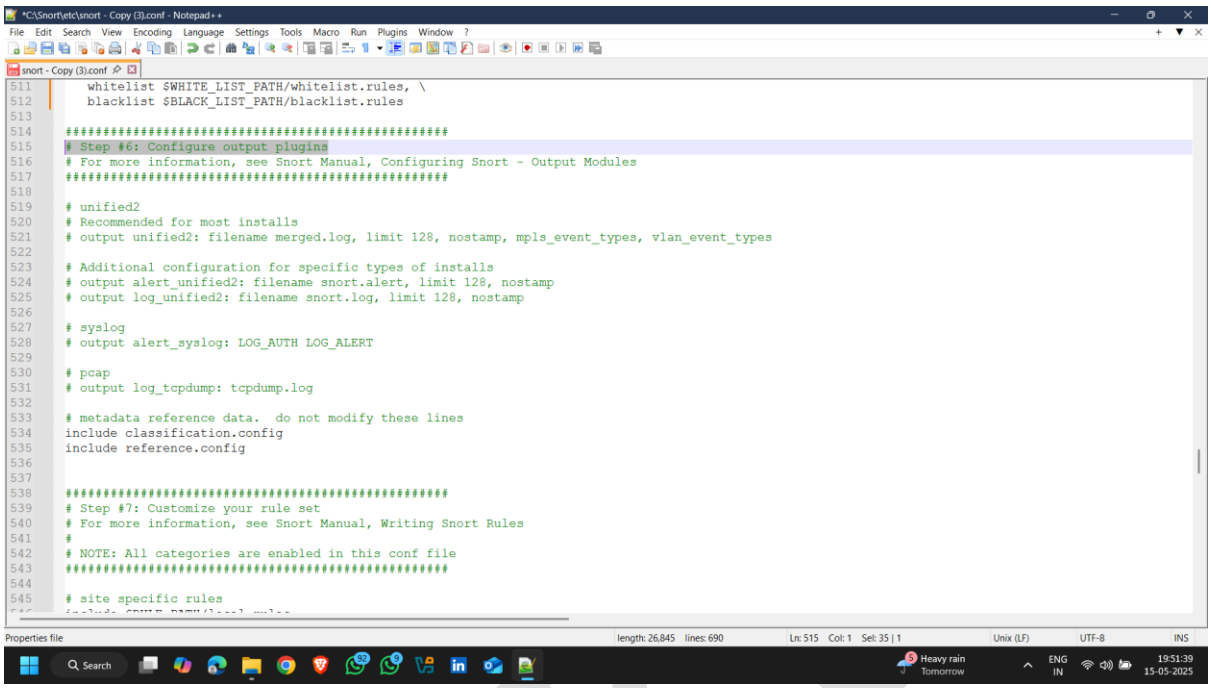
blacklist.rules copy file and paste in same folder and rename copy file to the whitelist.rules



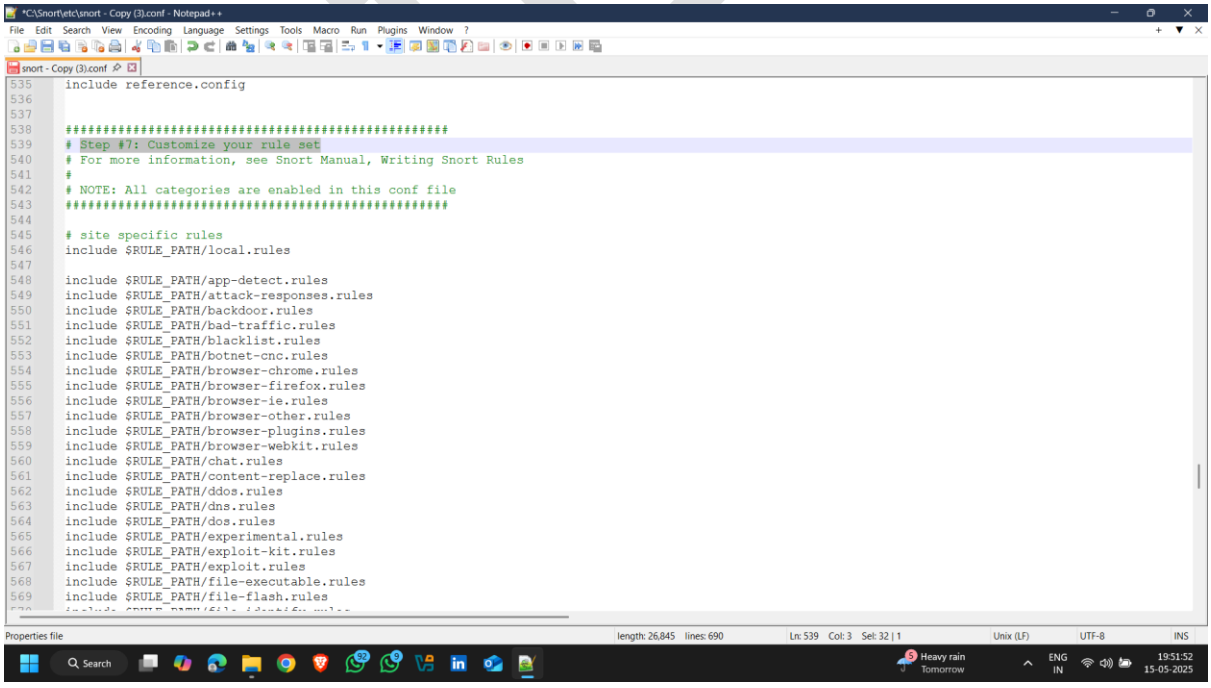
- Replace white_list.rules to whitelist.rules on 511
- Replace black_list.rules to blacklist.rules on 512



- Step 6
- Don’t do anything , no need to change

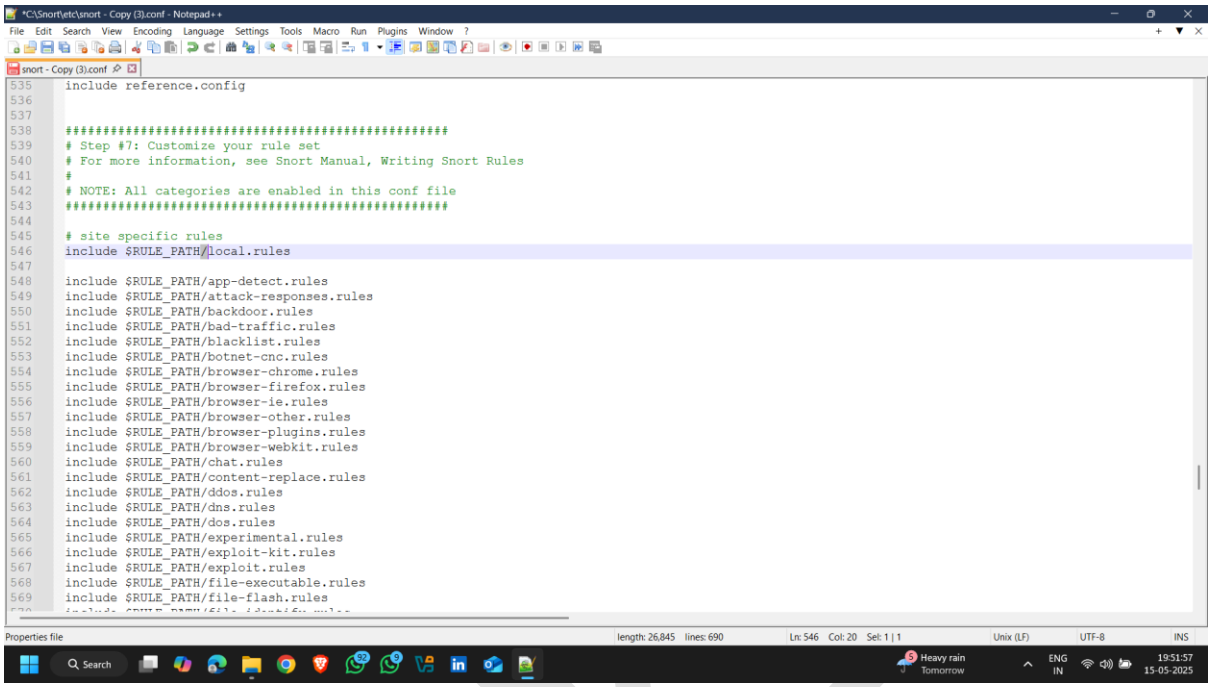


- step 7

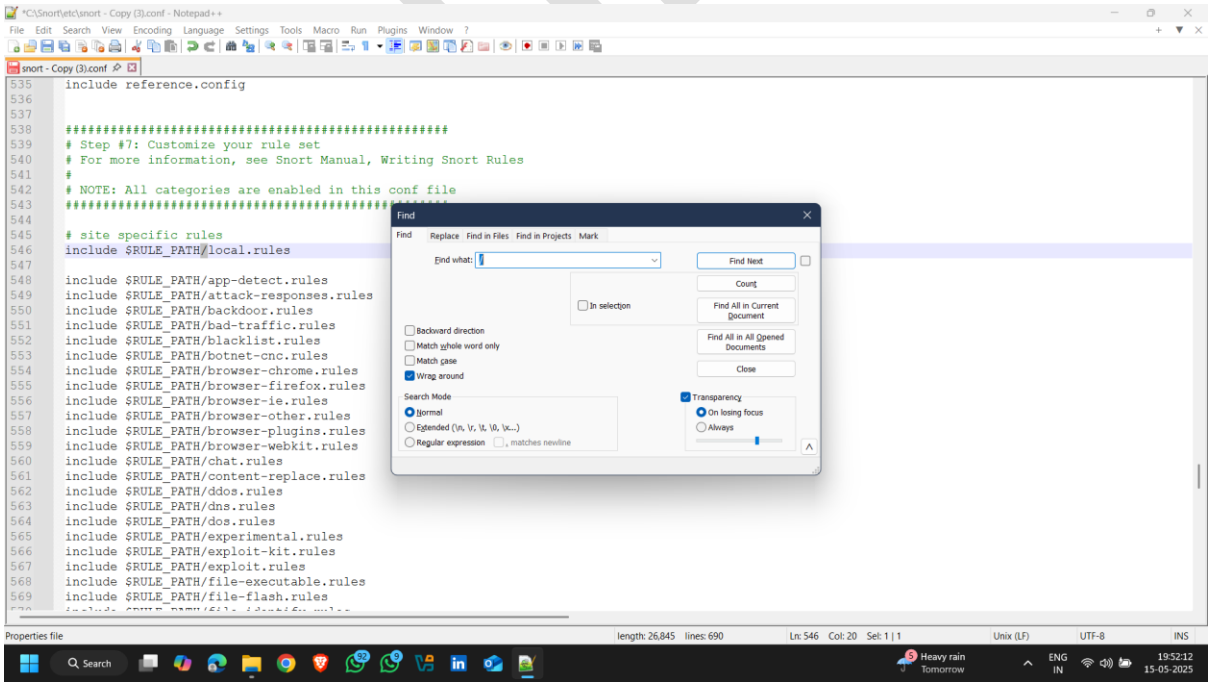


- go to the line number 546

- replace “/” to “\” 546 upto 651

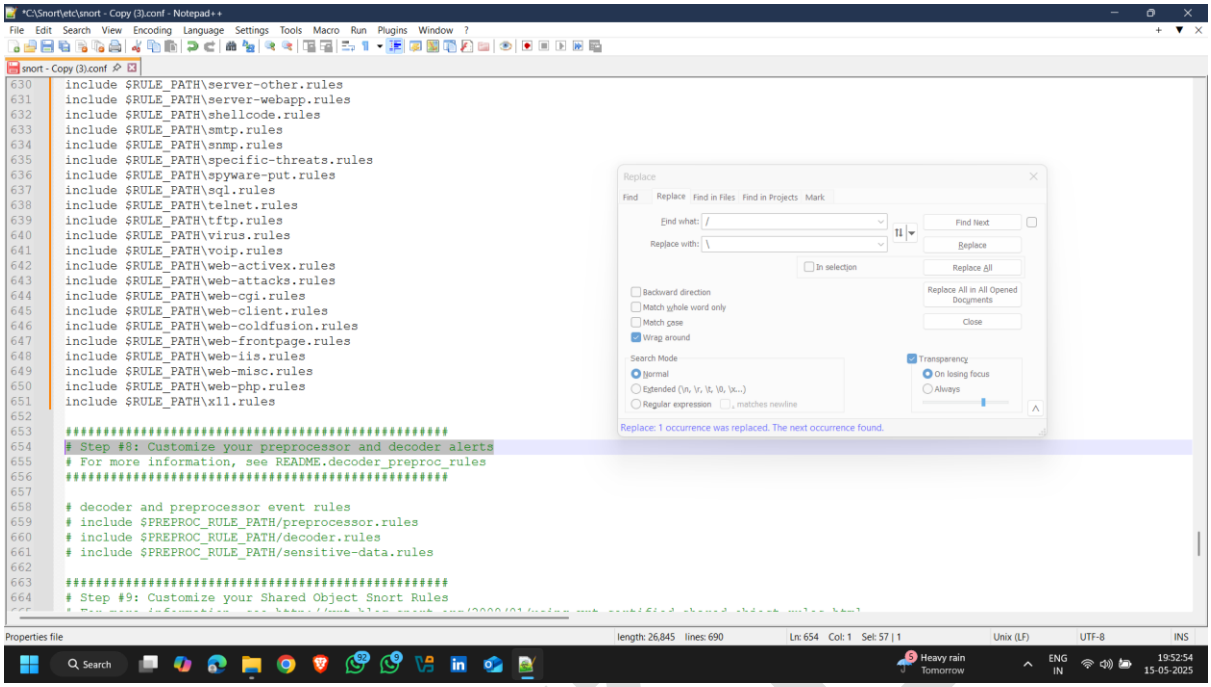


- press **ctrl + f** , click on replace

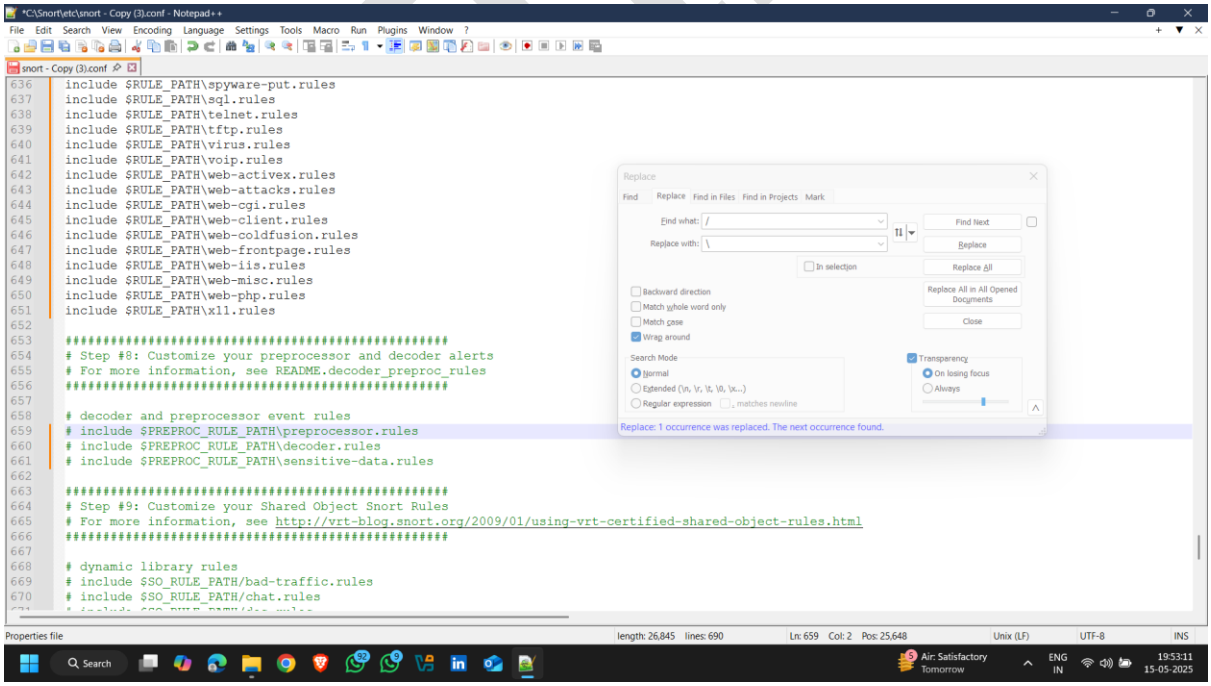


- add “/” in first box and add “\” in second column

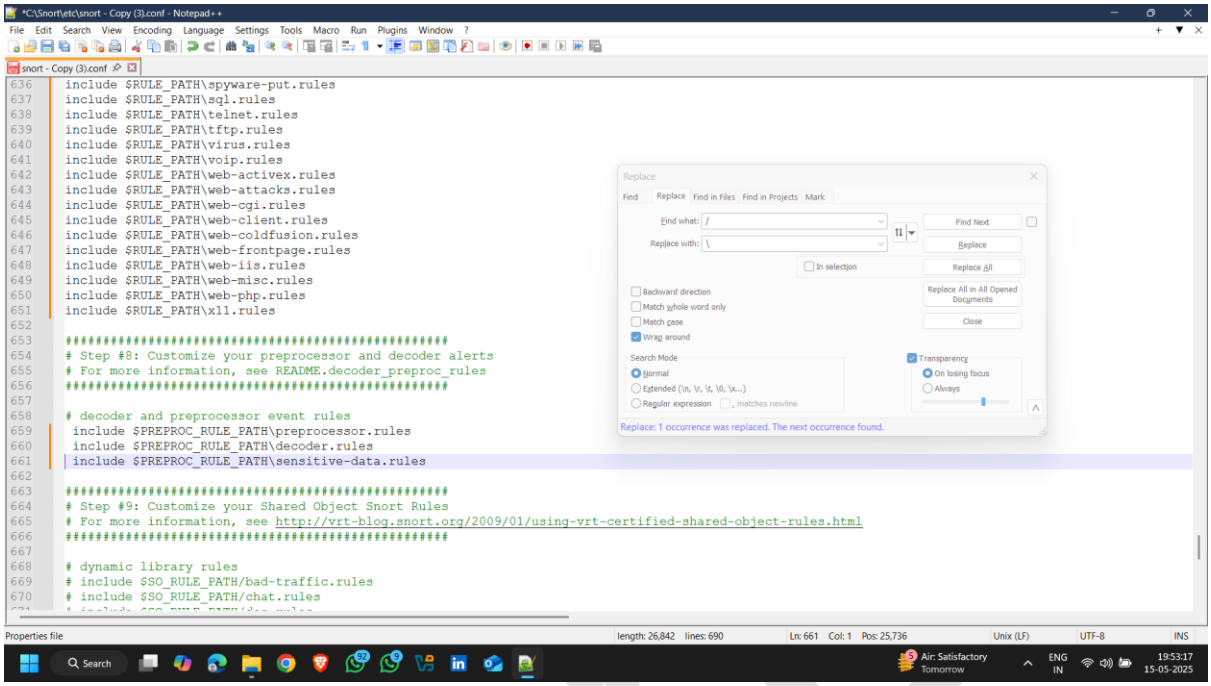
- click on **replace** ✅ and replace one by one



- step 8

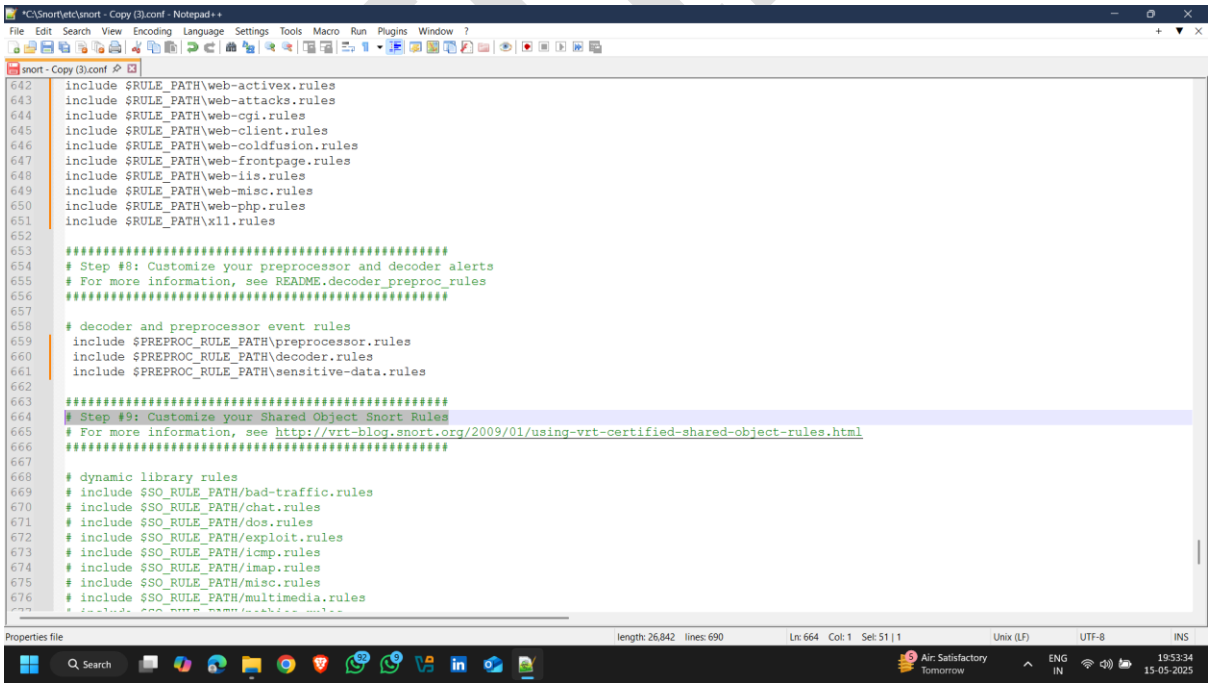


- follow step 7 process as it is on line number 659 to line number 661

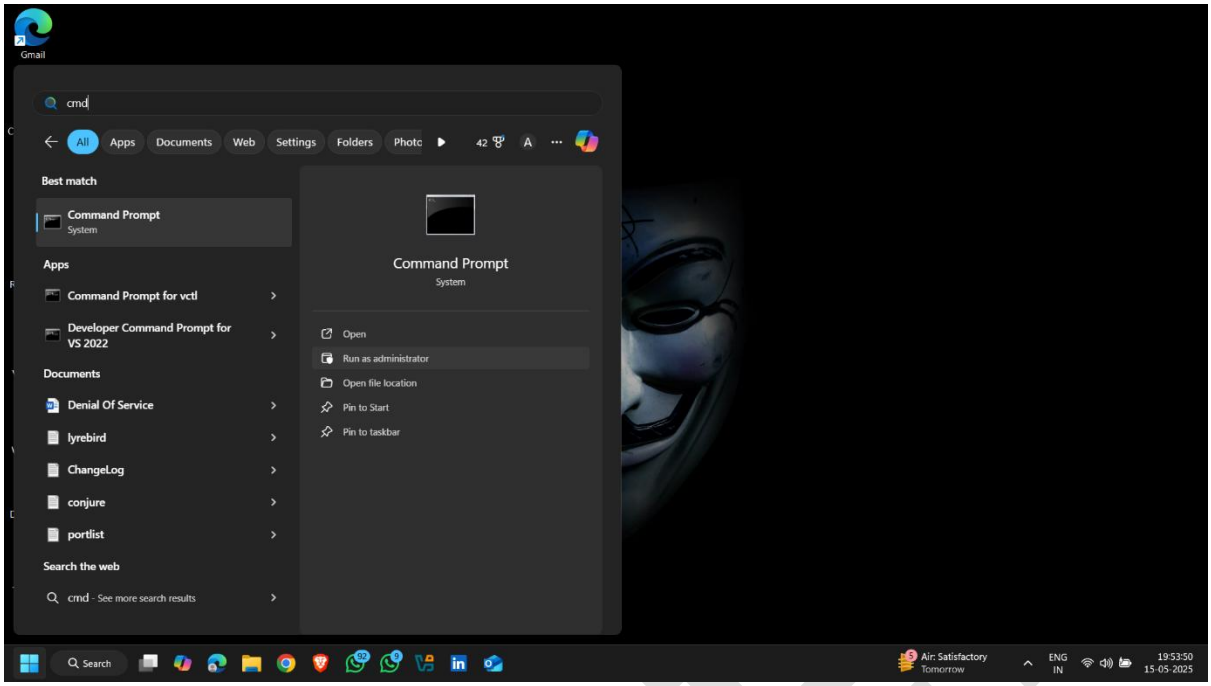


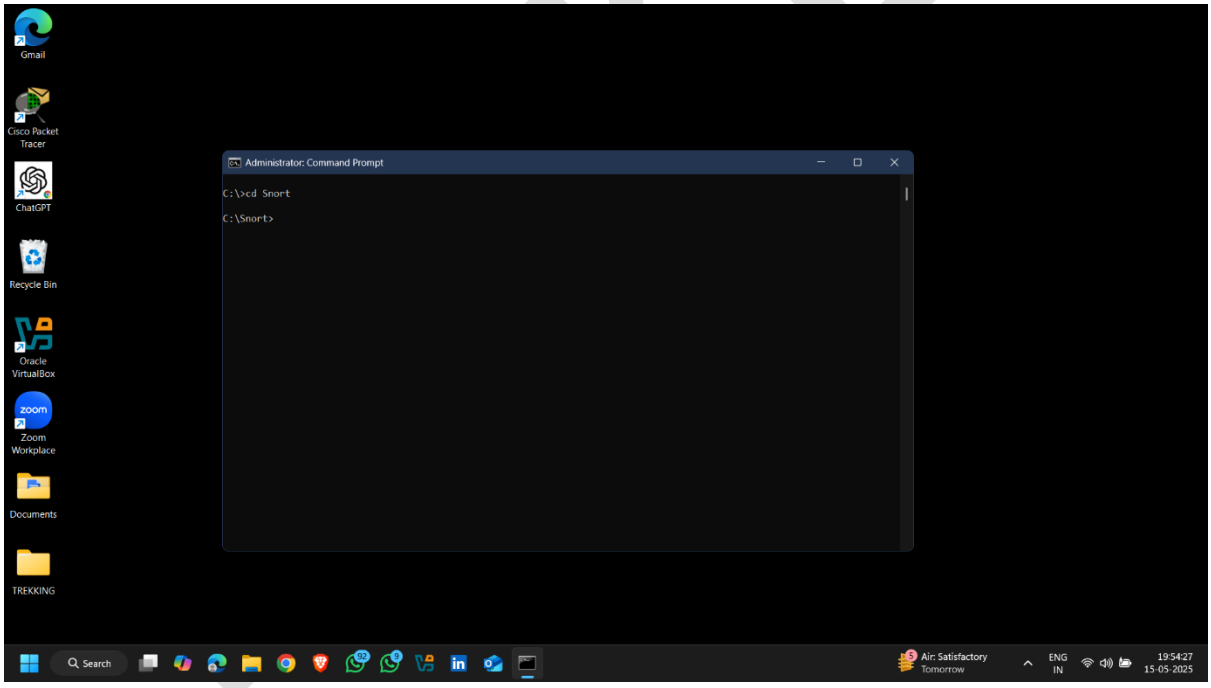
- Configuration done ✅

Note -: Step 9 for Linux Operating System, no need to change

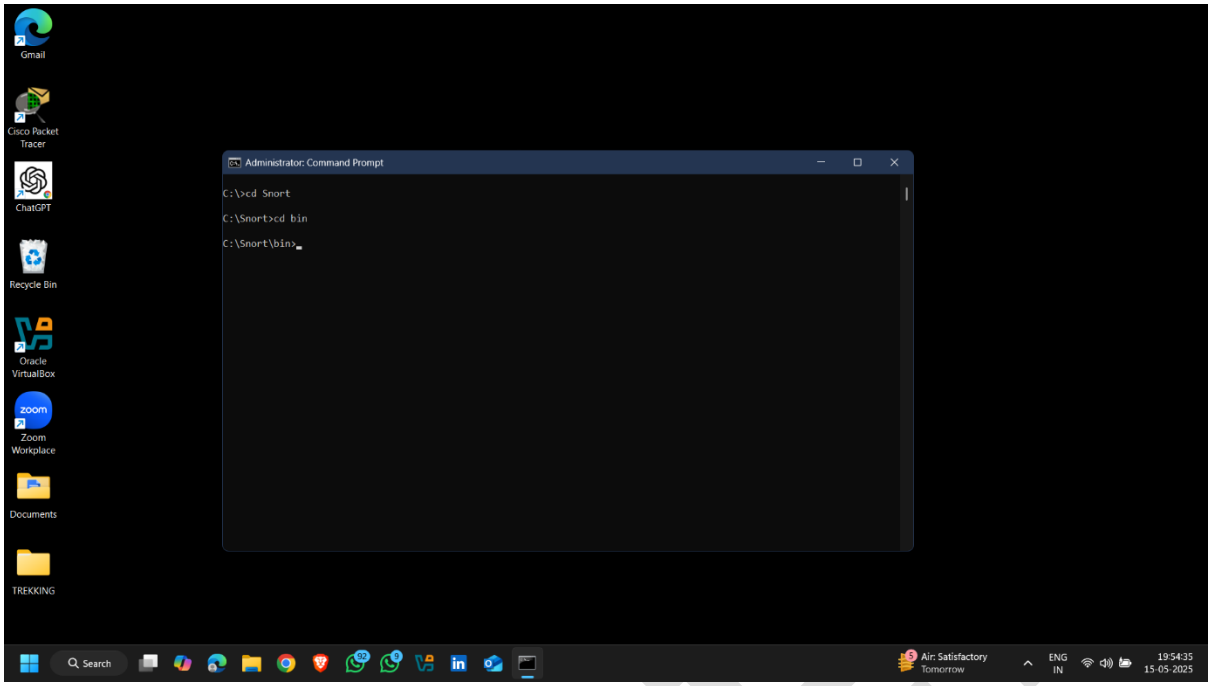


Now open command line interface as a administrator

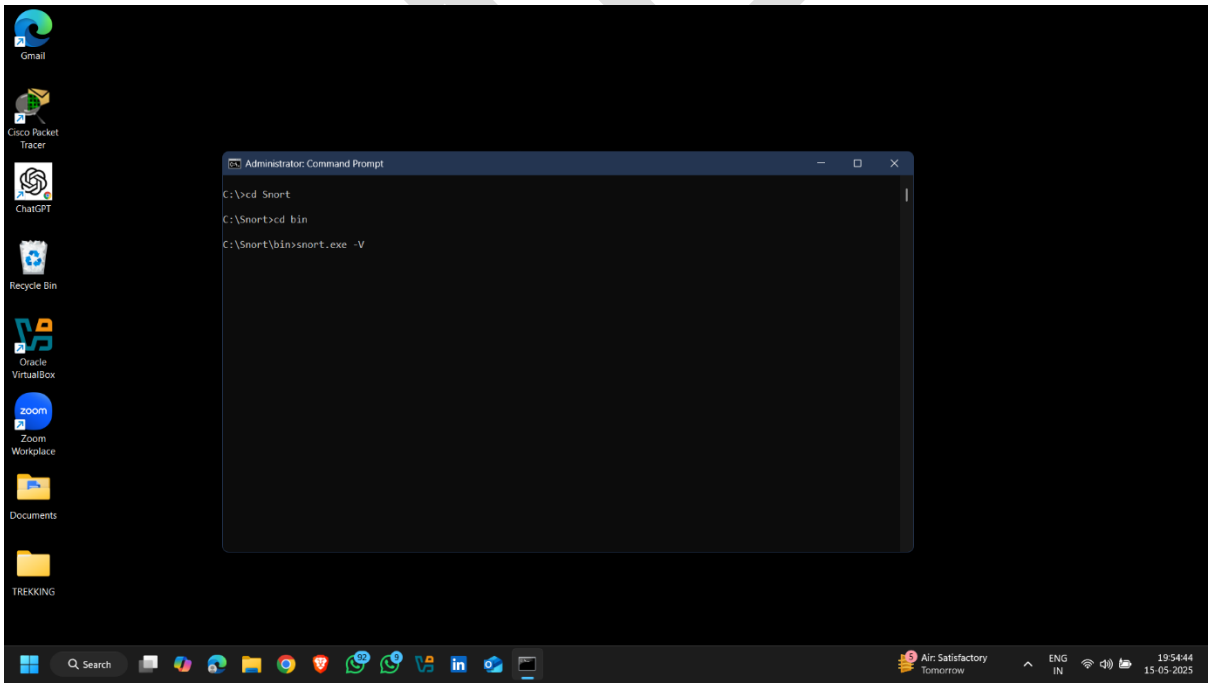


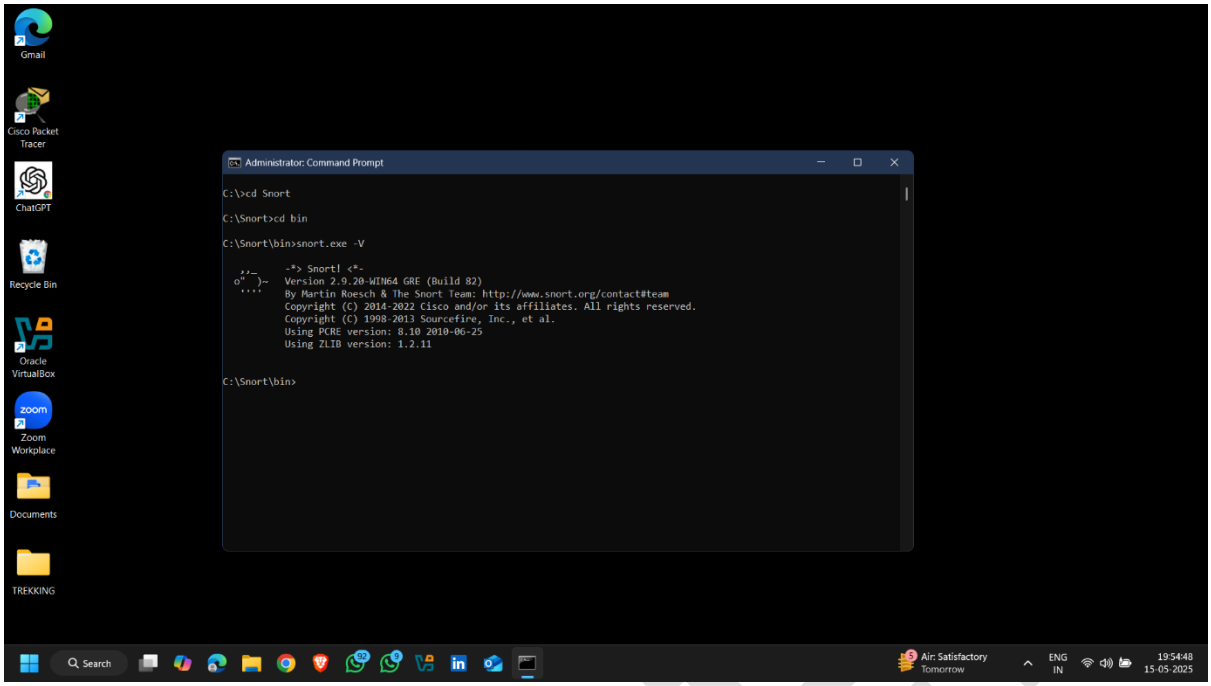


- **Go to the Snort\bin Folder**

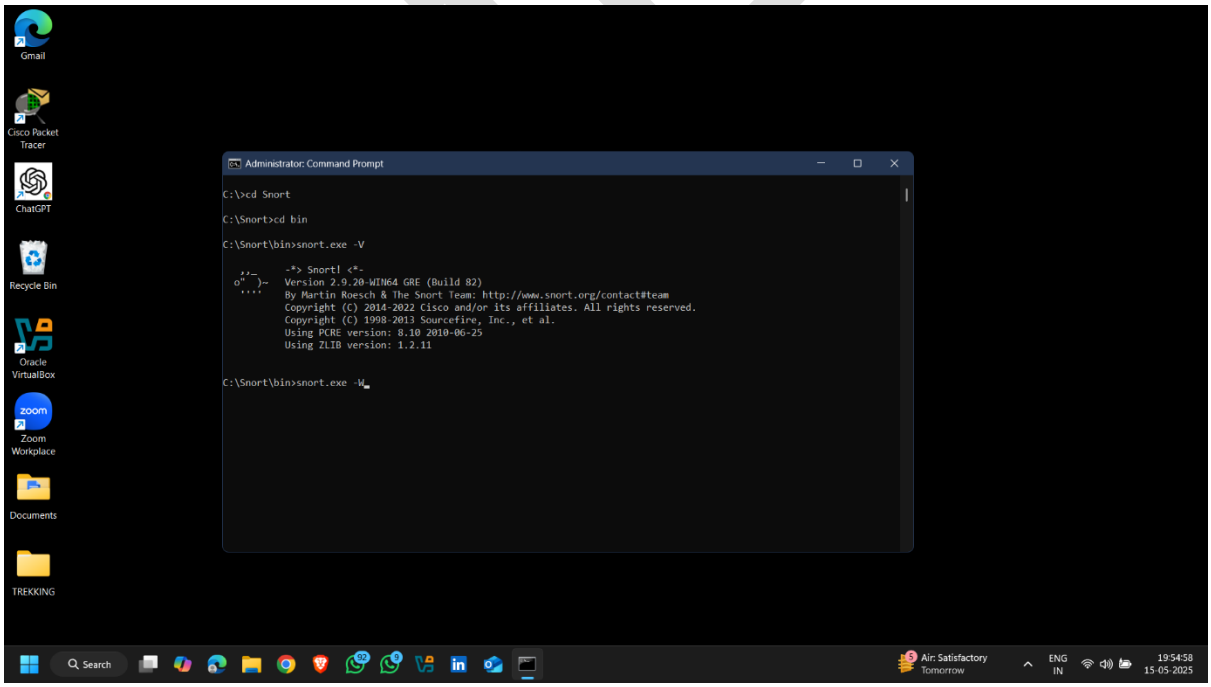


- There is file in snort bin folder >> snort.exe
- Run file snort.exe -v (v for version)

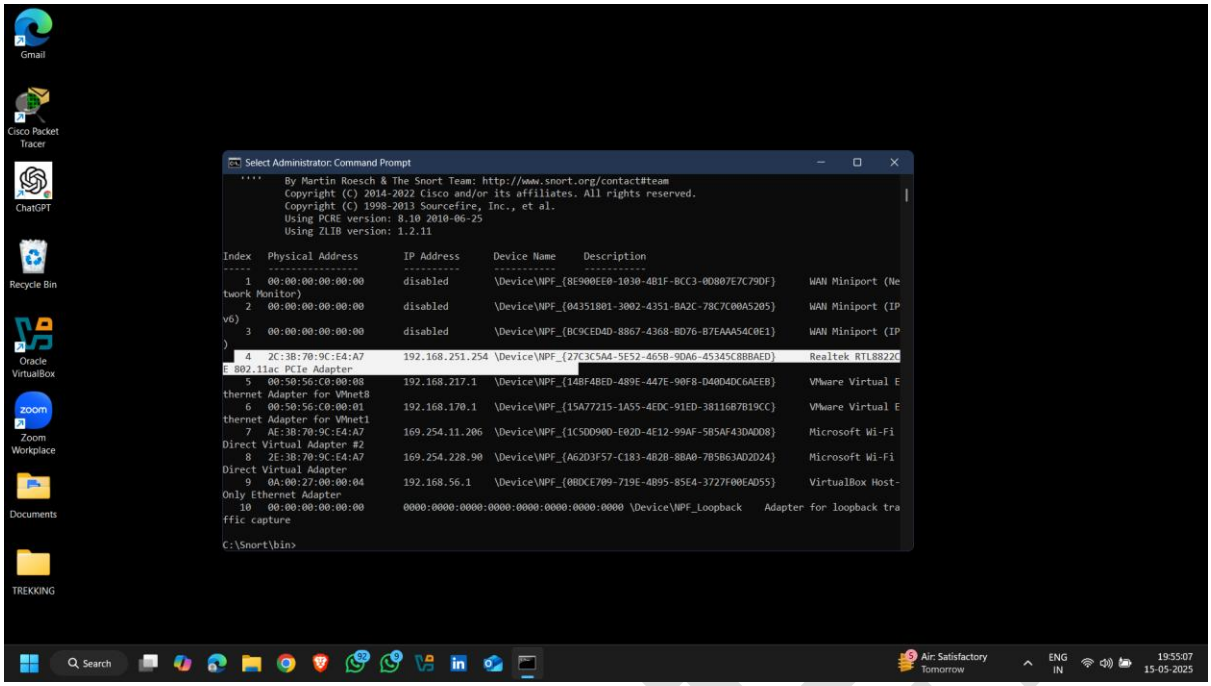




- use next command for finding network interface
- snort.exe –W

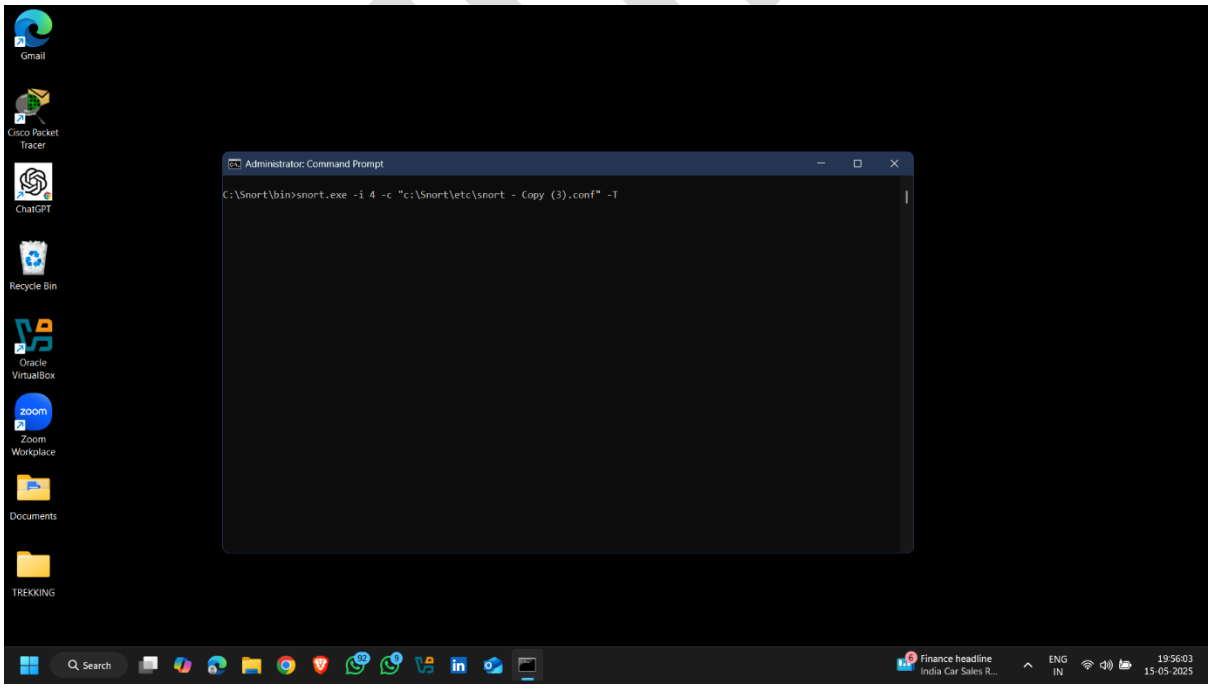


- **network interface 4**

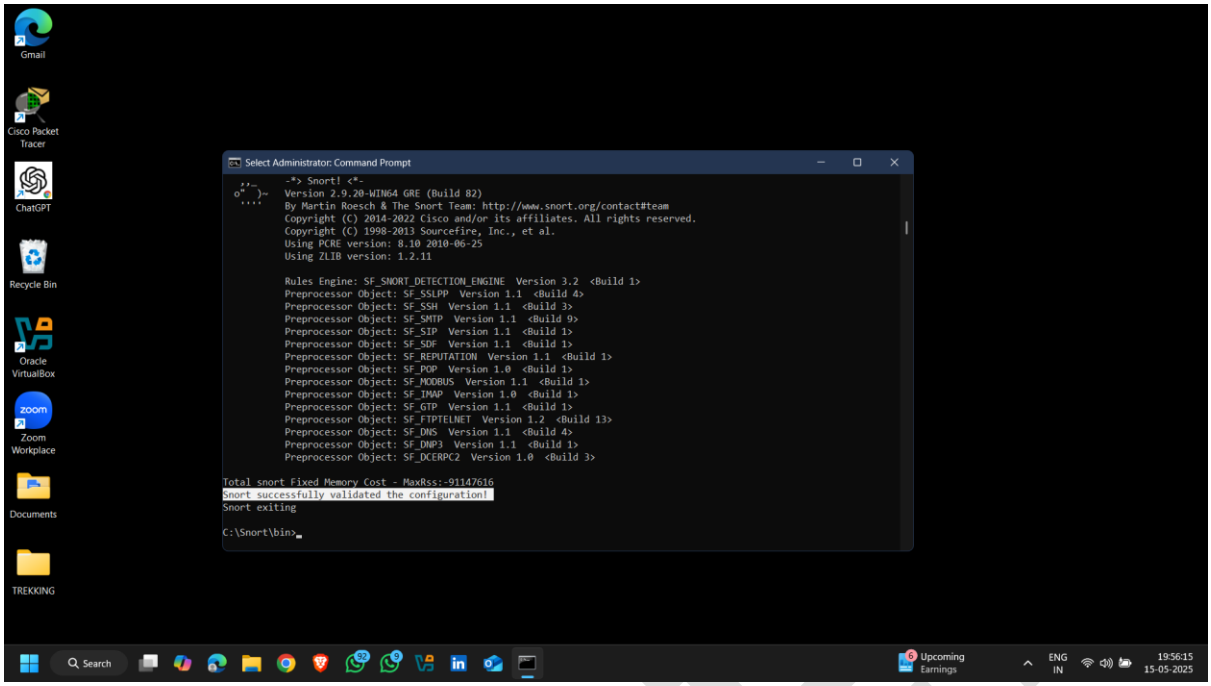


- use command for configuration testing

command -: snort.exe -i 4 -c “c:\Snort\etc\file name “ -T

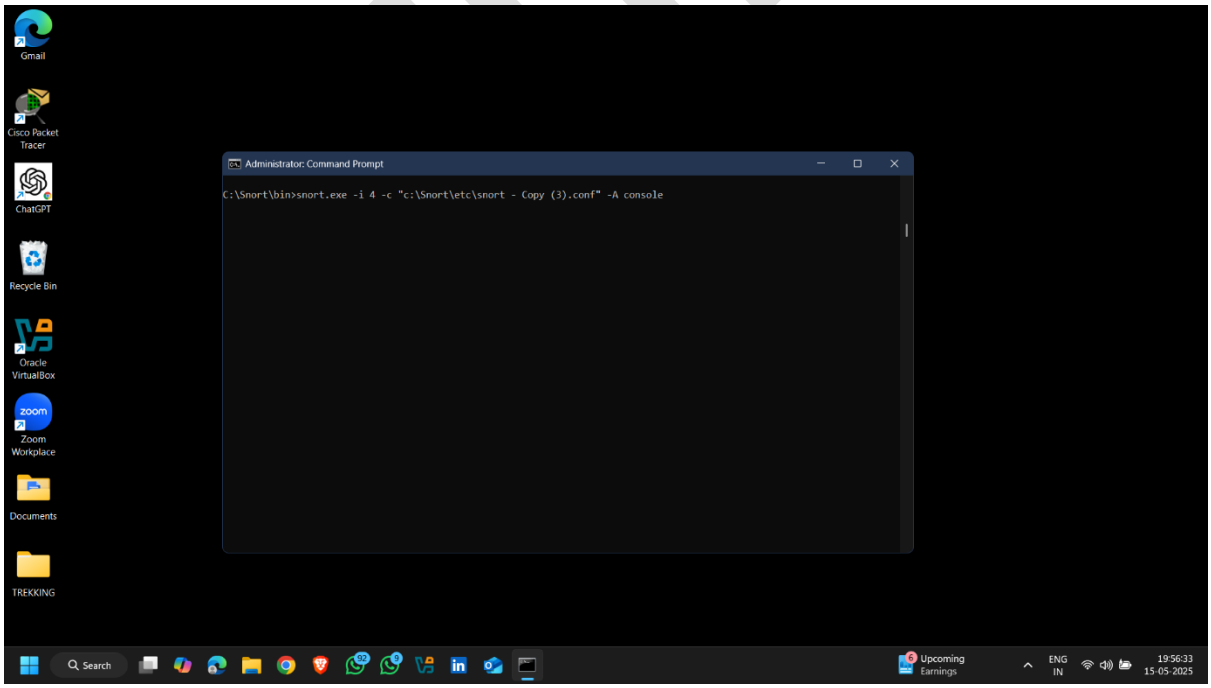


- **Configuration successful validate**

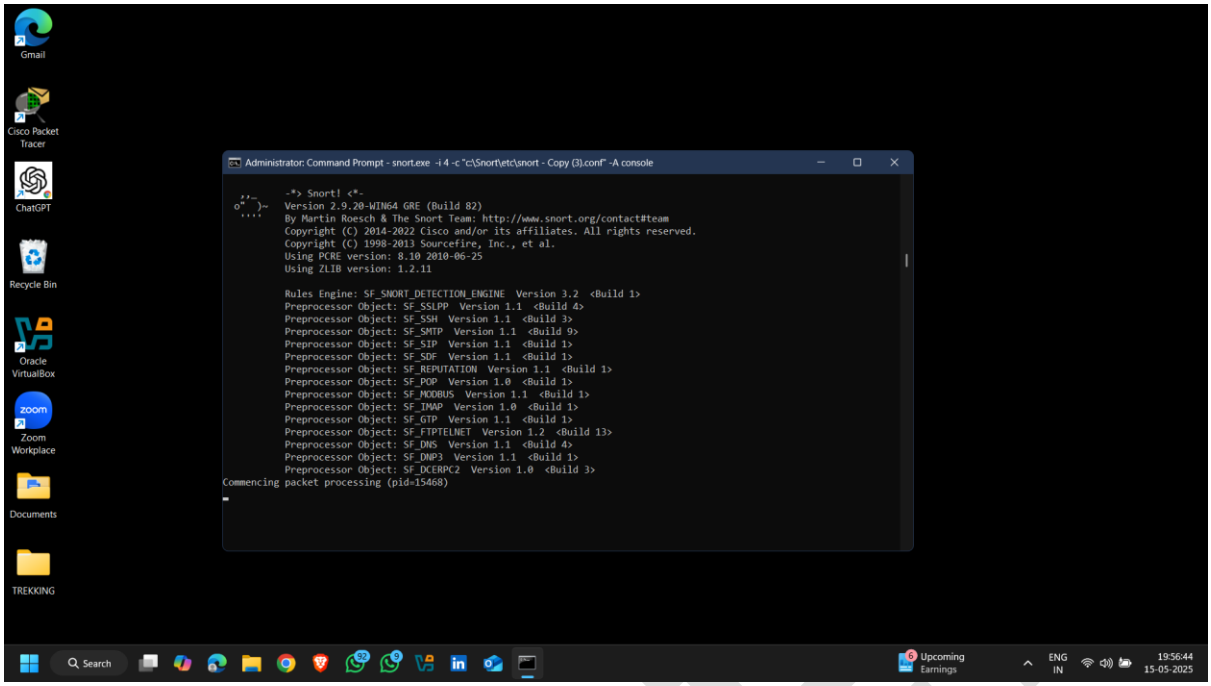


- **Type command to start snort**

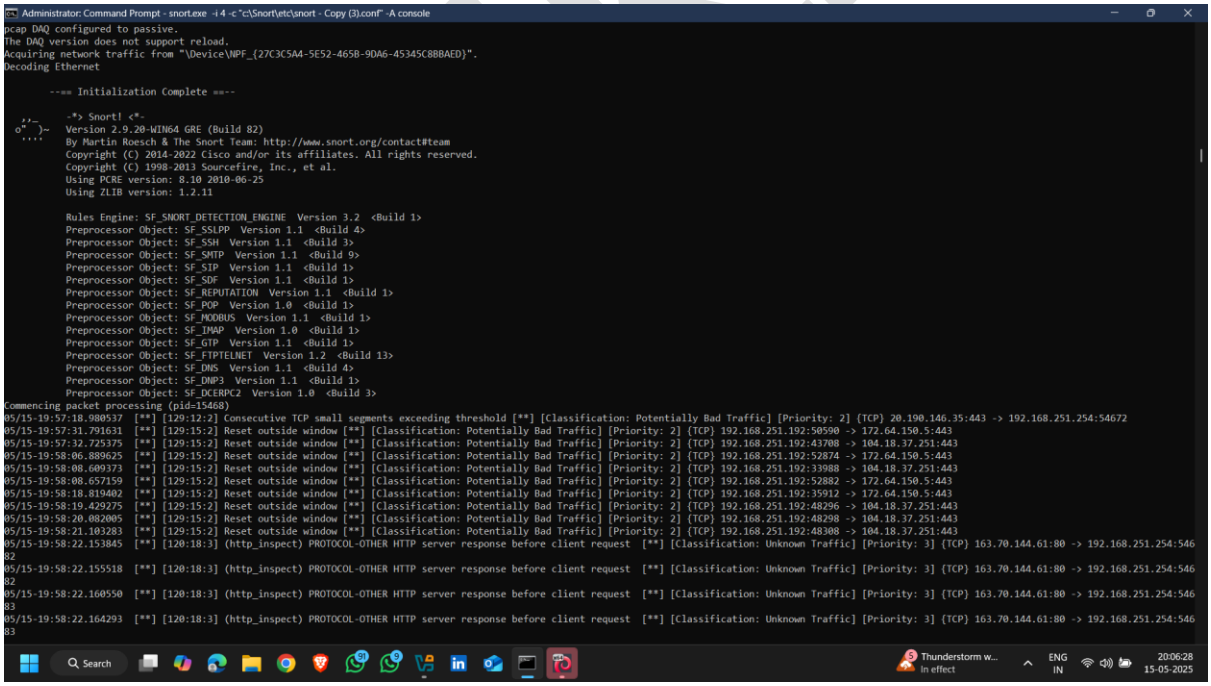
Command -: snort.exe -i 4 -c"c:\Snort\etc\file name " -A console



- Here snort started 👇



- **Started capturing network traffic**



Windows Firewall Configuration

Windows Firewall (now known as **Windows Defender Firewall**) is a **built-in security feature** in Microsoft Windows operating systems that helps **protect your computer** by filtering **incoming and outgoing** network traffic based on security rules.

🧱 Purpose of Windows Firewall:

To prevent **unauthorized access** to or from a private network — acting as a **barrier between your PC and external threats**.

🔍 Key Features:

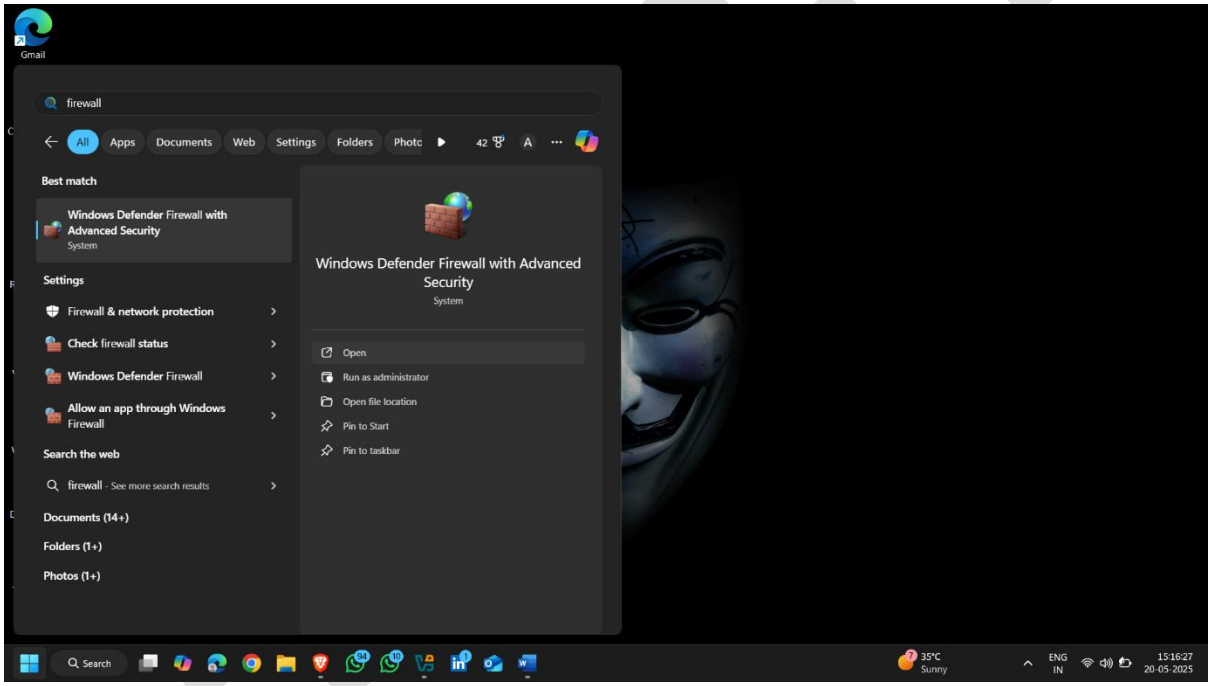
1. **Inbound & Outbound Filtering:**
    - Blocks or allows traffic **based on rules**.
    - Inbound = traffic coming into your computer.
    - Outbound = traffic leaving your computer.
2. **Predefined Security Rules:**
    - Automatically configures rules for common applications and system services.
3. **Application Control:**
    - Prompts when an **unknown app** tries to access the network.
4. **Network Profiles:**
    - Customize rules for:
        - **Private** networks (home/trusted).
        - **Public** networks (unsafe, like public Wi-Fi).

- **Domain** networks (used in corporate environments).

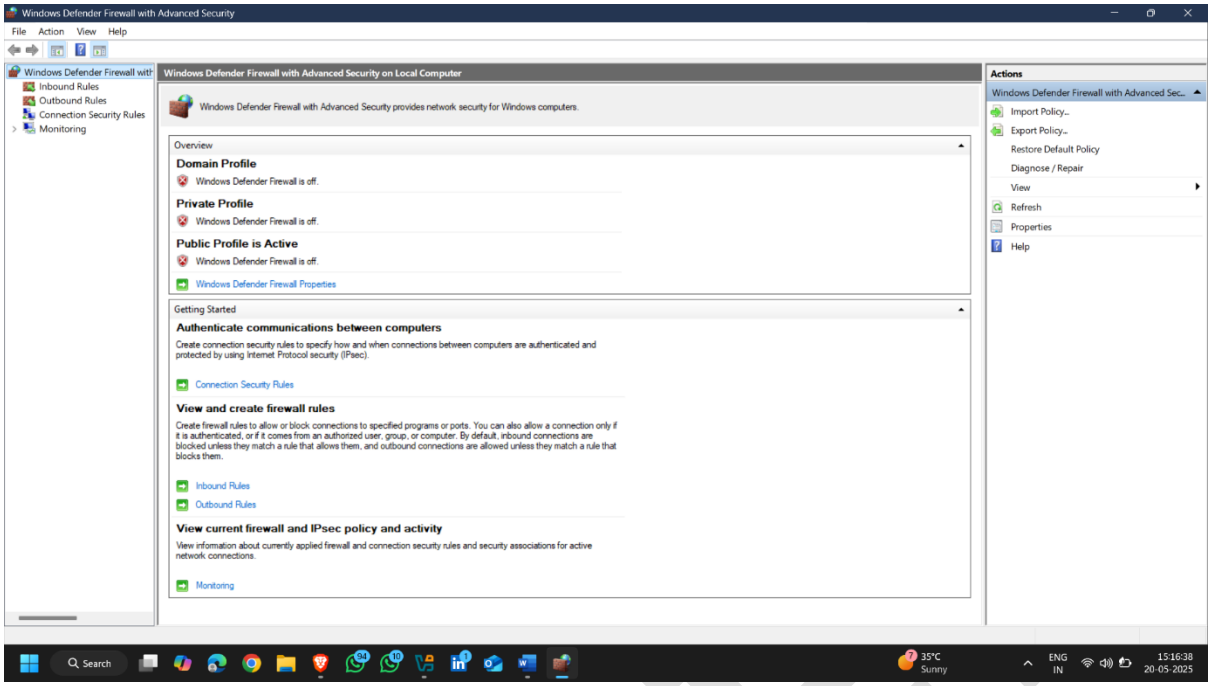
5. **Integration with Windows Security Center:**
    - Easily managed via **Control Panel** or **Windows Security Settings**.
6. **Logging and Monitoring:**
    - Tracks dropped packets and connection attempts.

How to do it -: (INBOUND RULES)

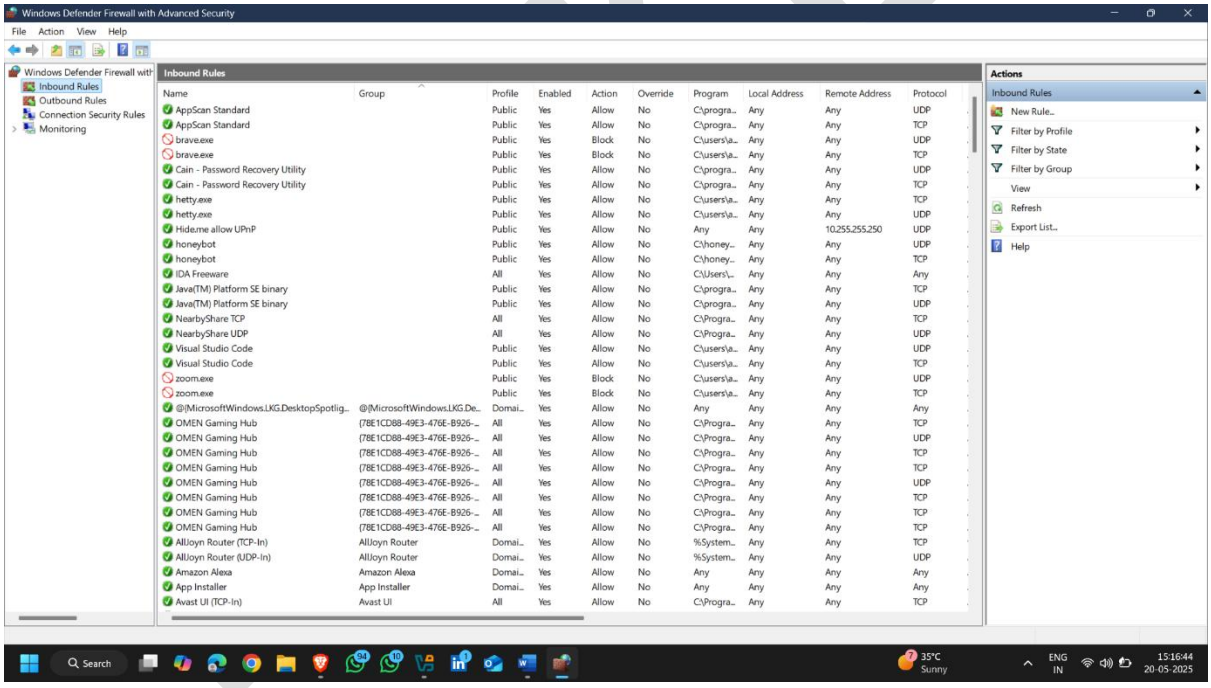
- Click on Windows Button and search Firewall and open it .



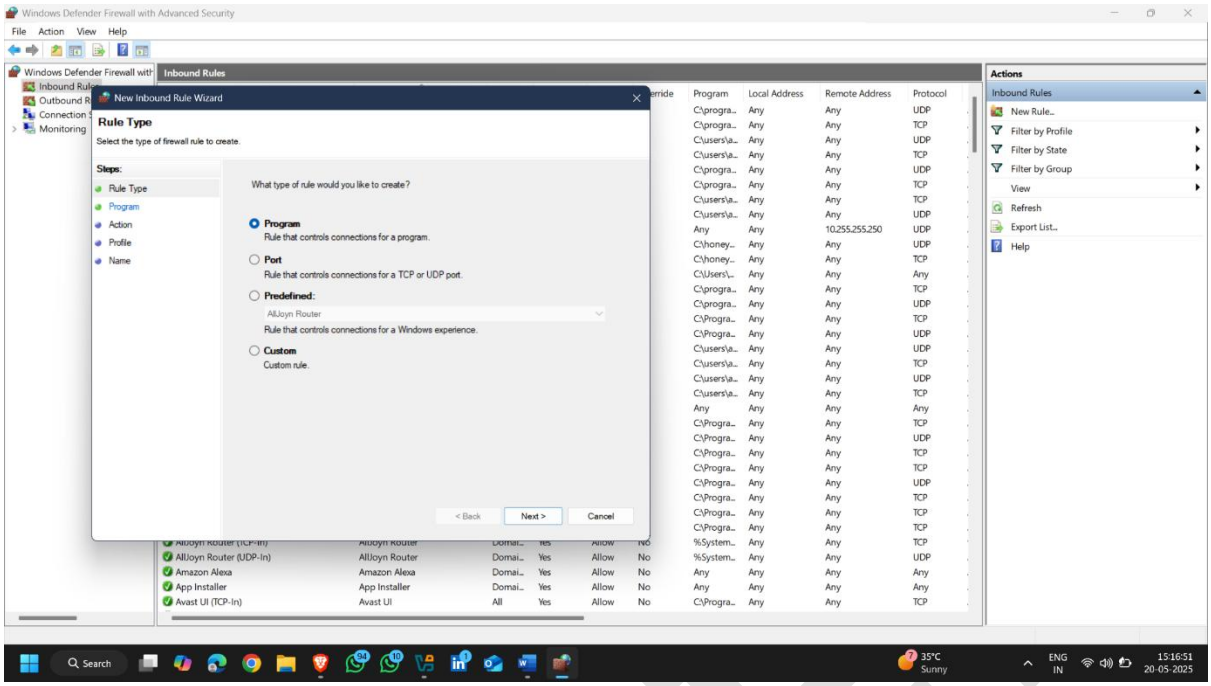
- New Window Open , there is option **inbound rules** – it means set rules for incoming network



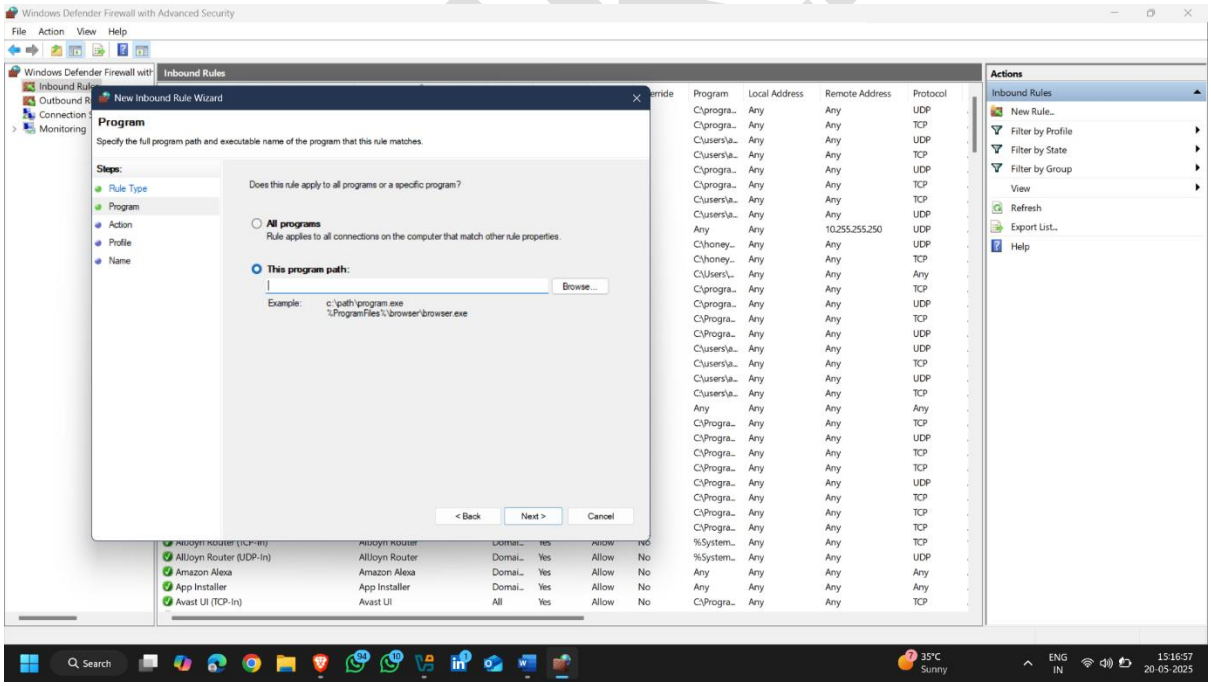
Click on Inbound rules then click on new rule



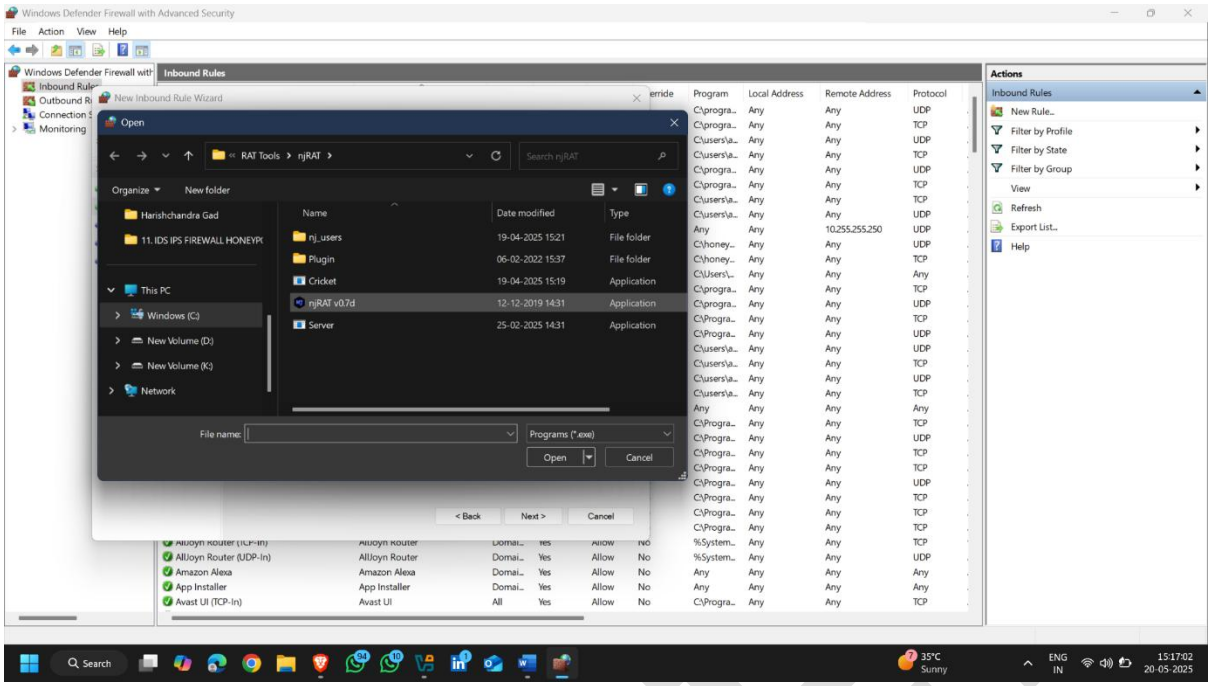
- Next pop up appears , there is many options ,then select the option that you want to set configuration like particular port or program ,this kind of stuff and then click on next



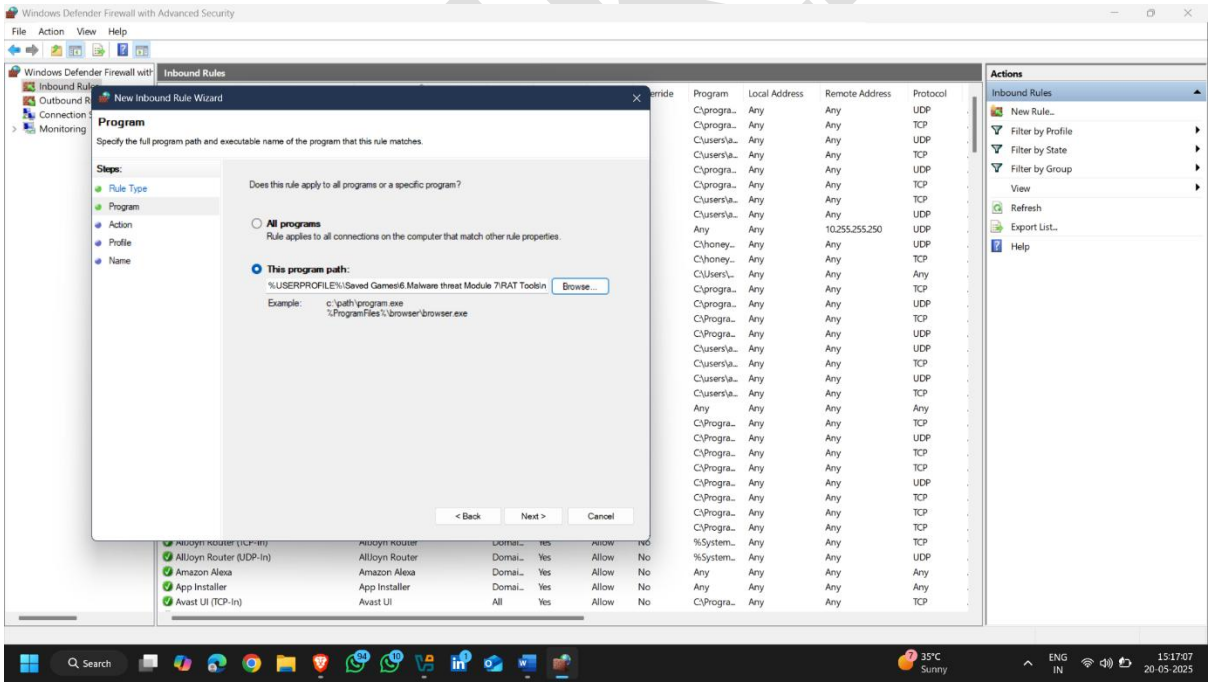
- Then set a path of program



- Select a file and click on open



- Click on next



Now there is a three option -:

- 1. allow the connection.
- 2. allow the connection if it is secure .

- 3. Block the connection .

1. Allow the connection

- ✅ **Description: This allows all matching incoming traffic, whether or not it is secured.**
- 🔒 **Security: Not restricted to authenticated or encrypted connections.**
- ✅ **Use When: You trust the traffic or program and want to permit communication without extra security restrictions.**

---

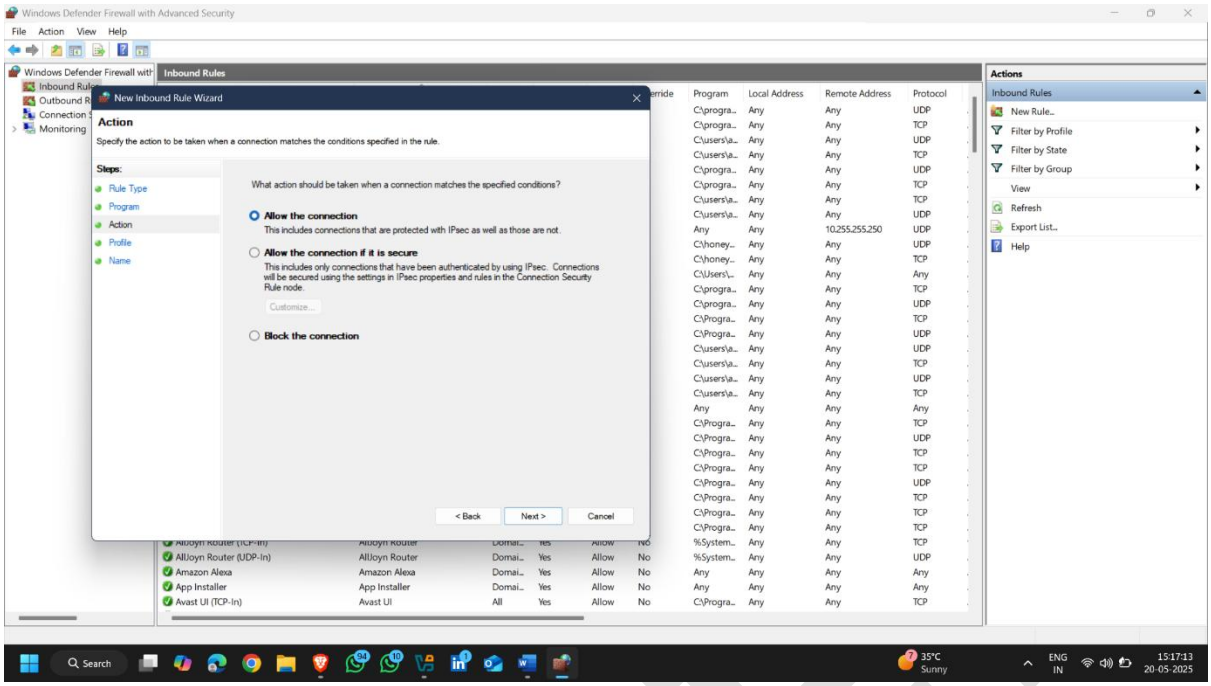
2. Allow the connection if it is secure

- 🔐 **Description: Only allows connections authenticated using IPsec (Internet Protocol Security).**
- 🔧 **Requires: Proper configuration of IPsec policies.**
- 📌 **Use When: You want to only permit secure (encrypted and authenticated) connections for sensitive systems or data transfer.**
- ⚠️ **Customize button is usually greyed out unless IPsec is configured.**

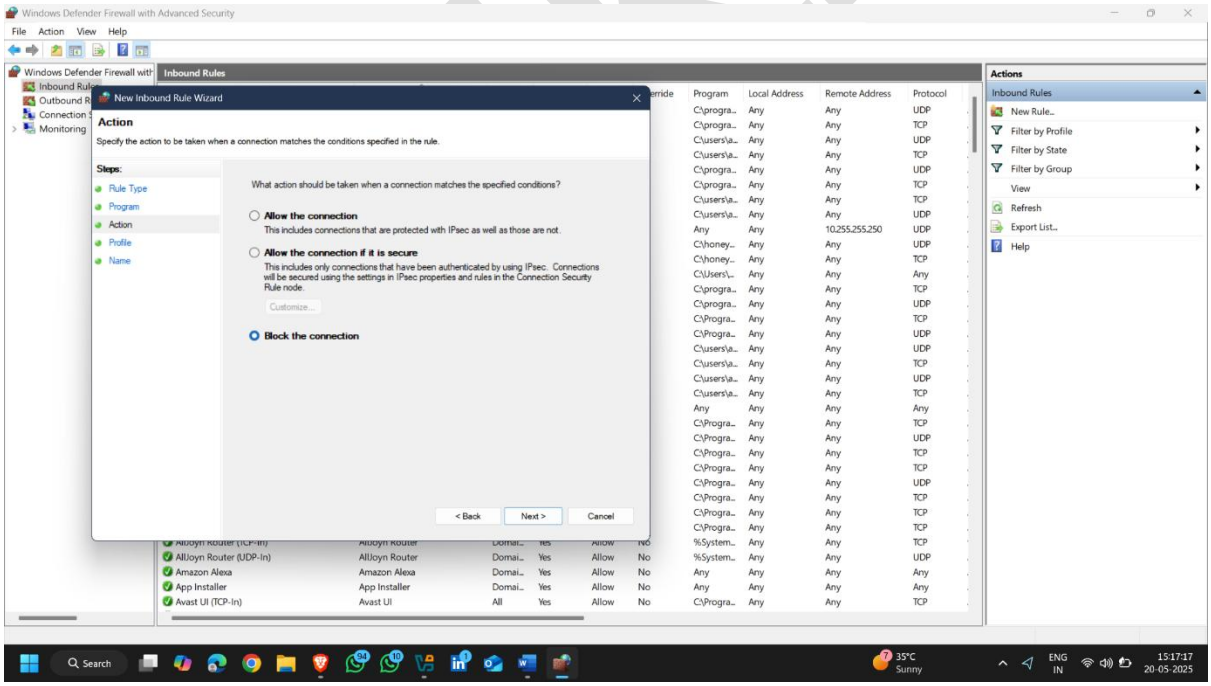
---

3. Block the connection

- ❌ **Description: Denies the connection completely, regardless of whether it's secure or not.**
- 🔒 **Security: Strictest setting; completely blocks the matched traffic.**
- ✅ **Use When: You want to prevent any communication through the specified ports, programs, or IP addresses.**



- After select option then click on next



Again Three options appear

✅ Domain

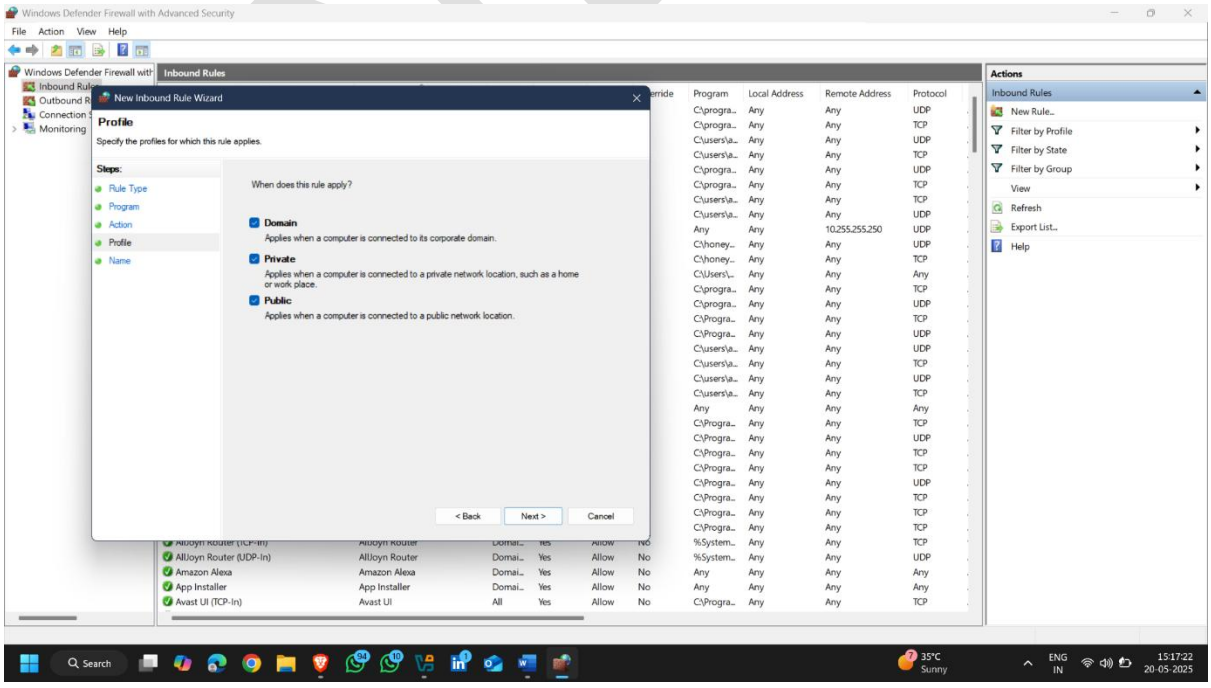
- **Applies When**: The system is connected to a corporate network **joined to a domain** (e.g., office or enterprise environment with Active Directory).
- **Use Case**: Apply this rule when you want it to work only in secure, managed networks.

✅ **Private**

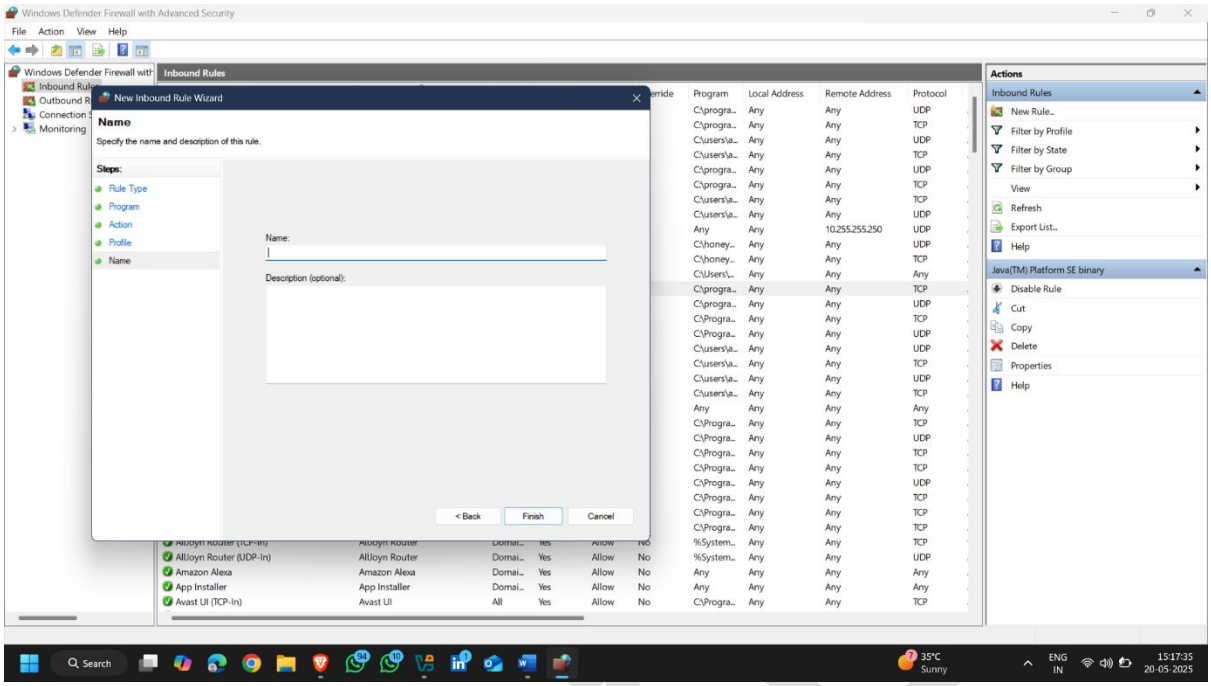
- **Applies When**: The computer is connected to a **trusted private network**, like your home Wi-Fi or a small office network.
- **Use Case**: Enable the rule when you're at home or in a network you trust.

✅ **Public**

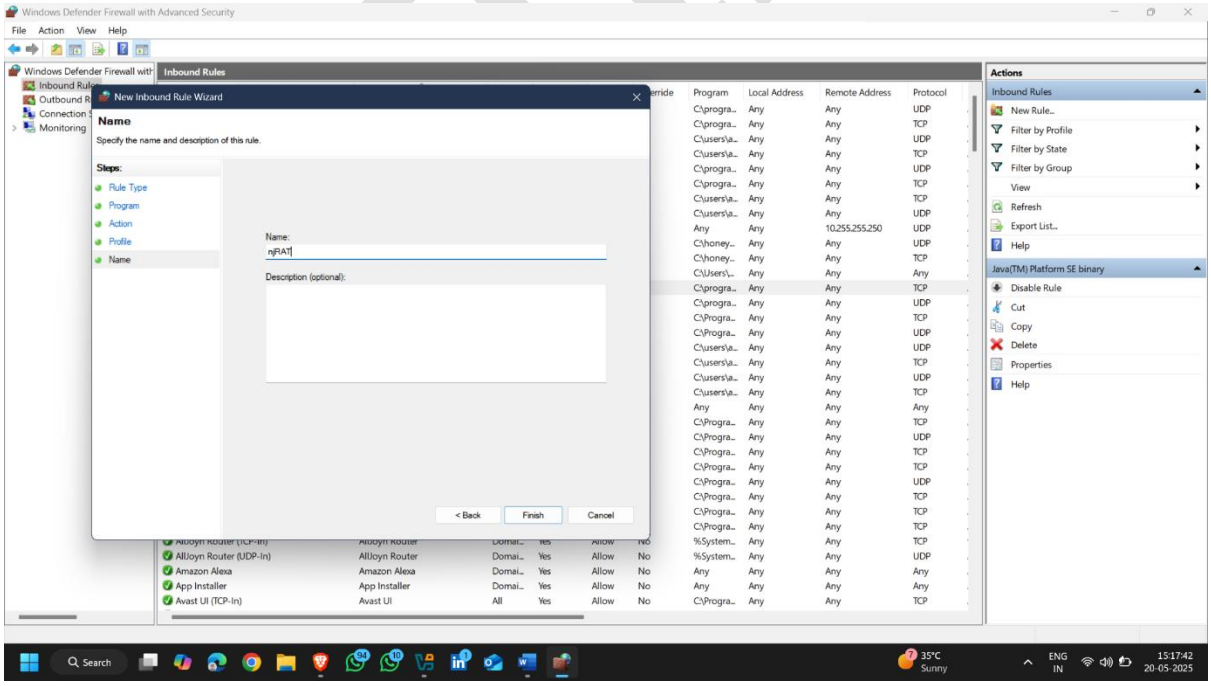
- **Applies When**: The system is connected to a **public network**, like at a café, airport, or hotel.
- **Use Case**: Apply the rule in public places. Be cautious here—public networks are less secure.



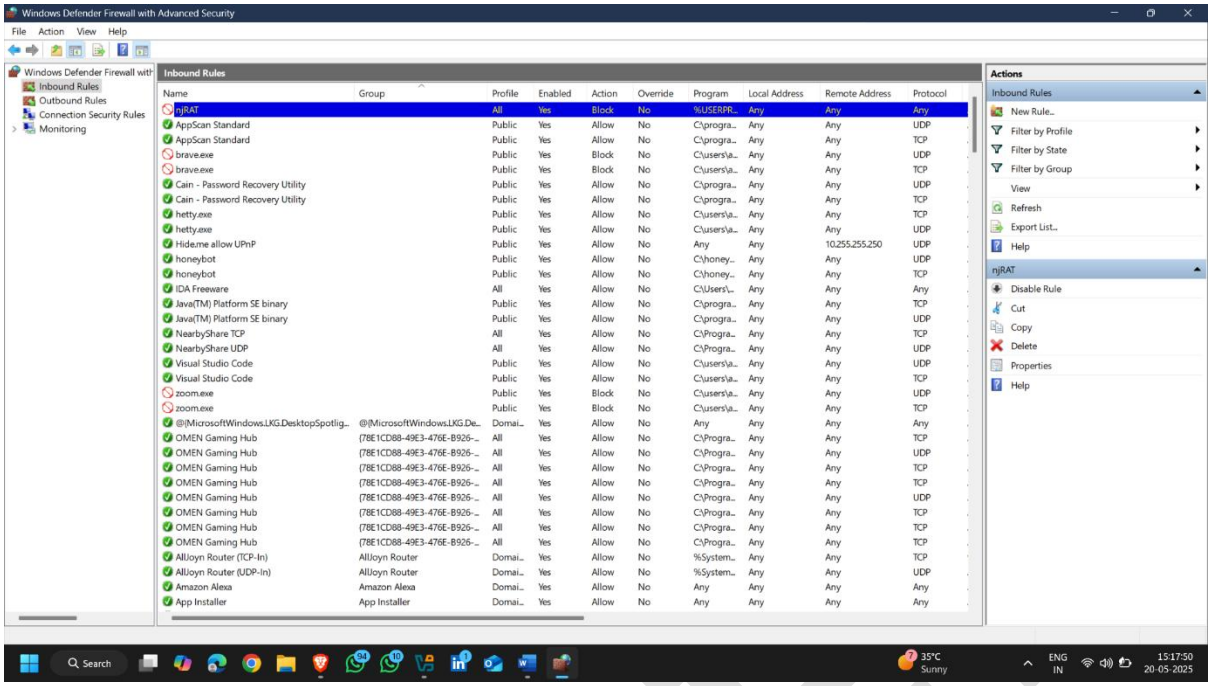
- Set a any name and click on finish



- 👇

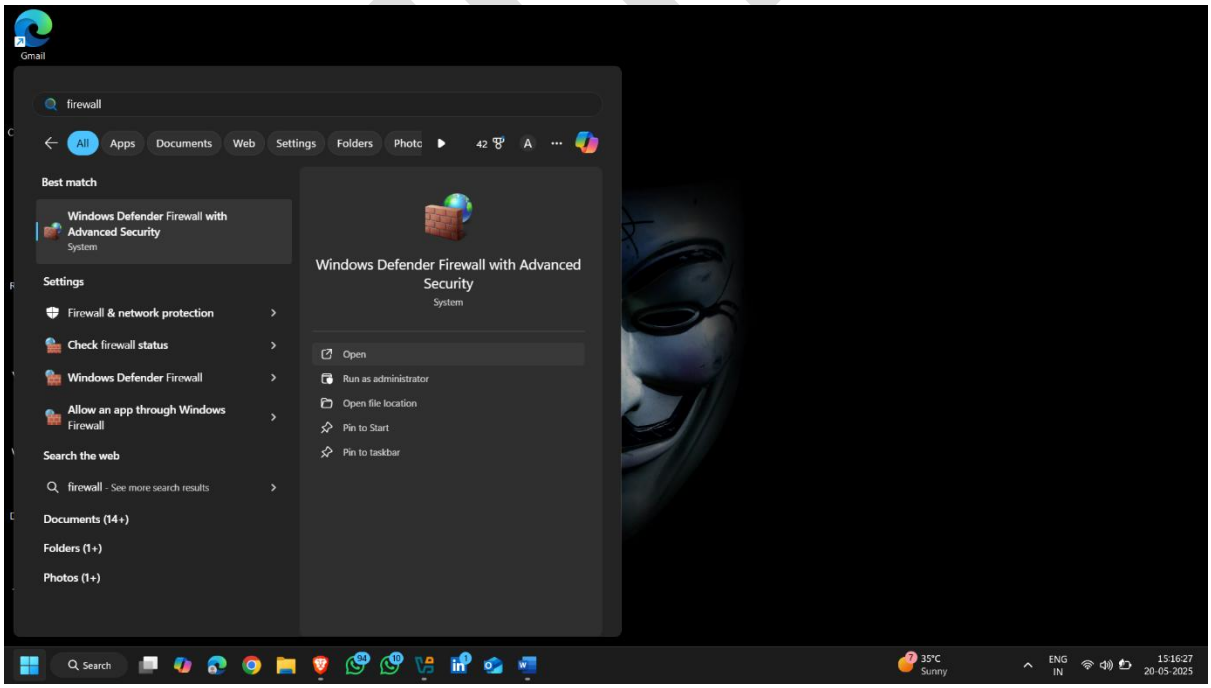


- Configuration set for inbound rules

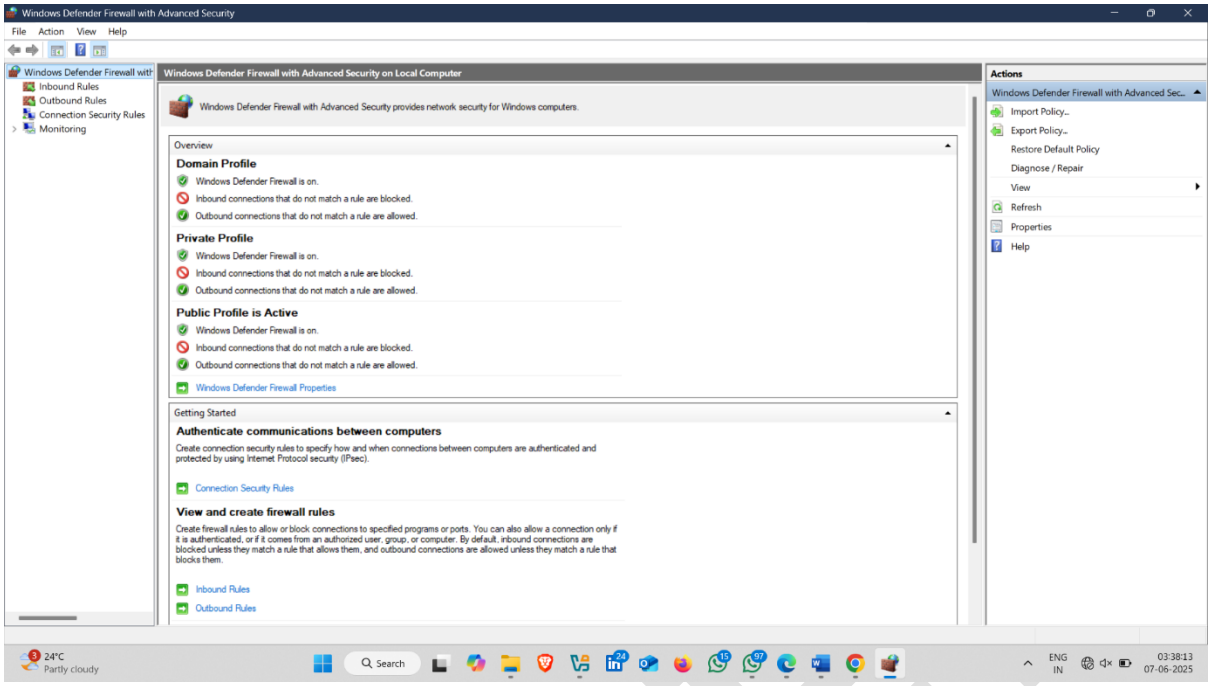


How to do it -: (OUTBOUND RULES) -:

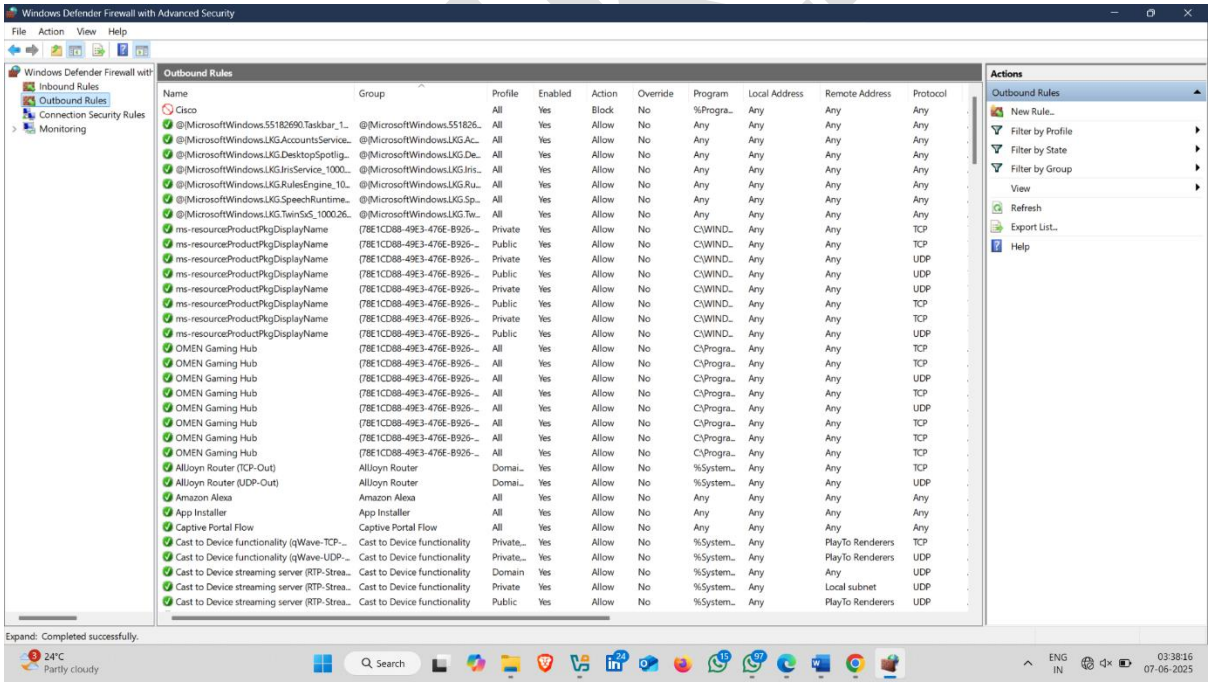
- Click on Windows Button and search Firewall and open it



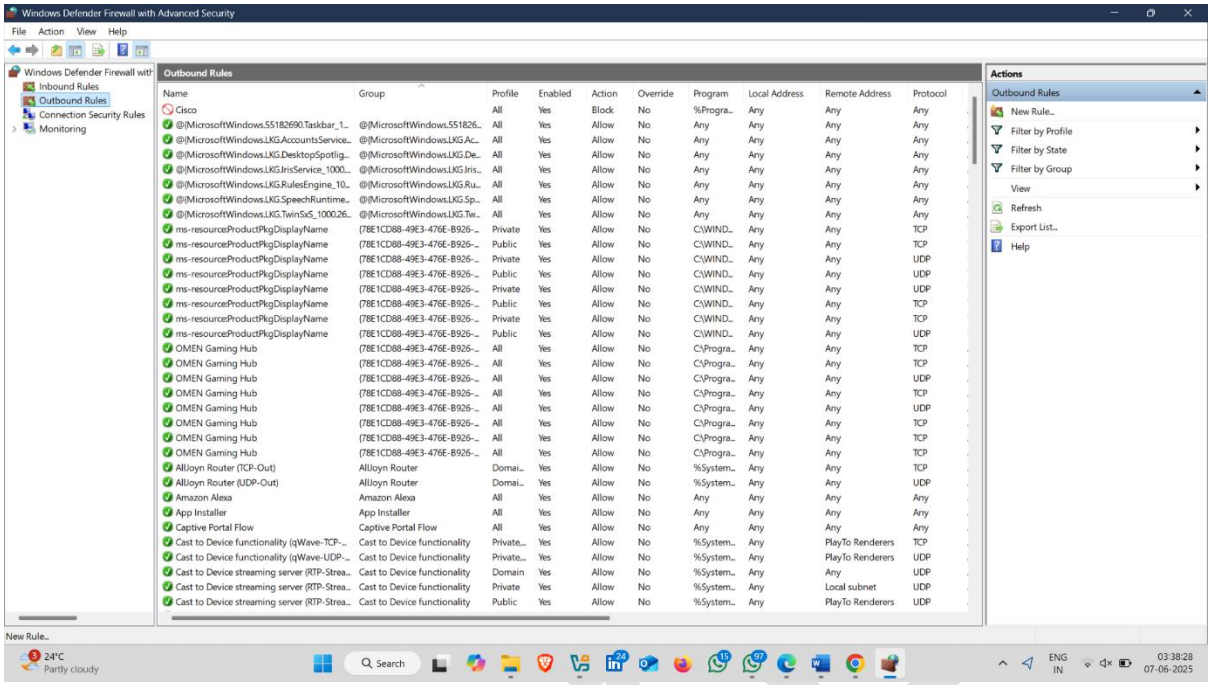
- New Window Open , there is option **outbound rules** – it means set rules for outgoing network ,



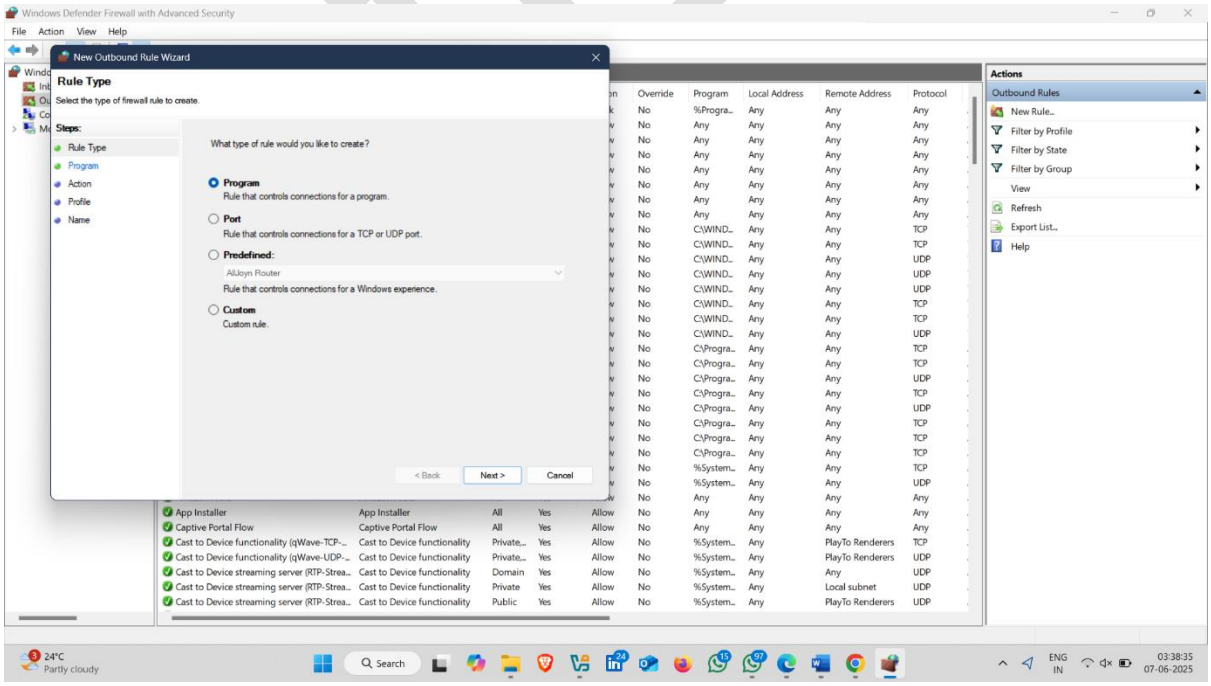
- Click on Outbound rules



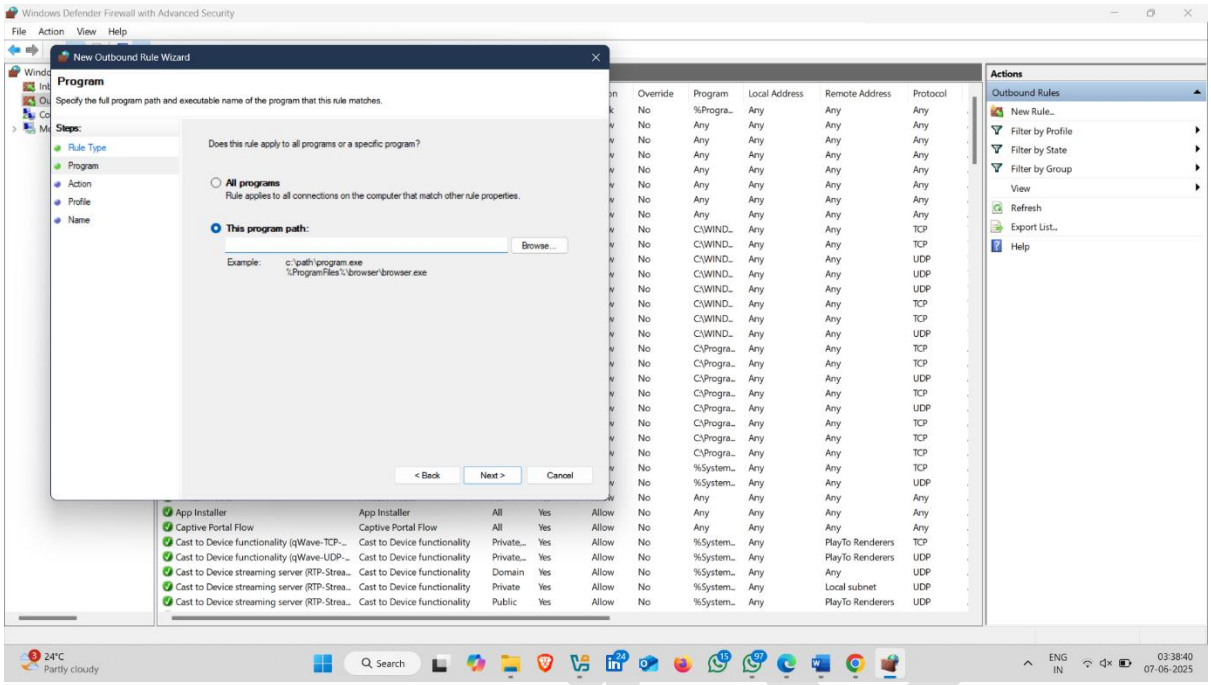
- Then click on New Rule



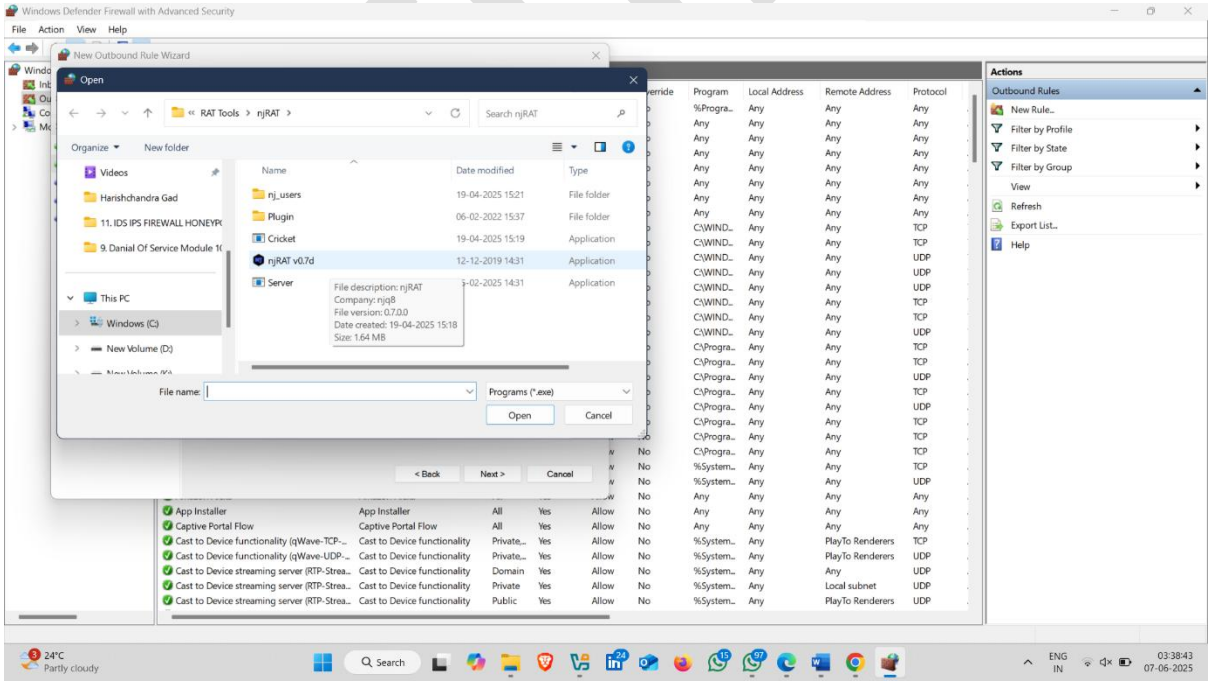
- New Pop Up appear with Three Options ,click on that you want to set a rule



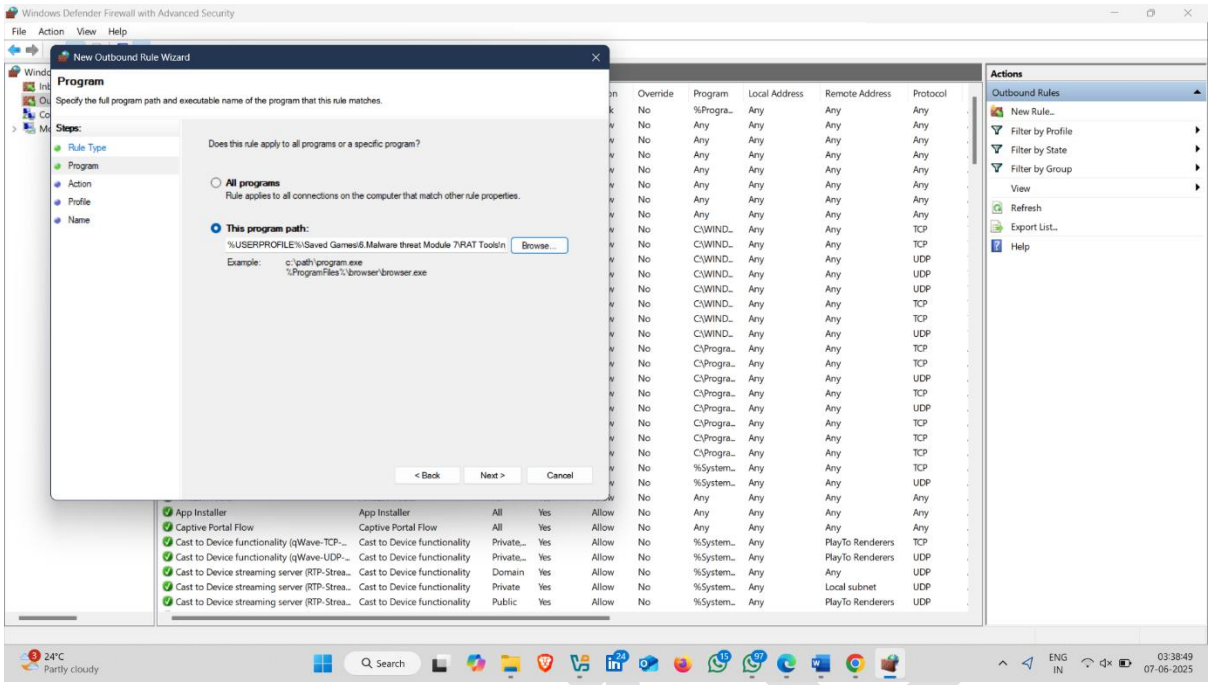
- Now , set the path by clicking on **Browse**



- Select the file and then click on open



- Click on Next



Now there is a three option -:

- 1. allow the connection.
- 2. allow the connection if it is secure .
- 3. Block the connection .

1. Allow the connection

- ✅ **Description: This allows all matching incoming traffic, whether or not it is secured.**
- 🔒 **Security: Not restricted to authenticated or encrypted connections.**
- ✅ **Use When: You trust the traffic or program and want to permit communication without extra security restrictions.**

---

2. Allow the connection if it is secure

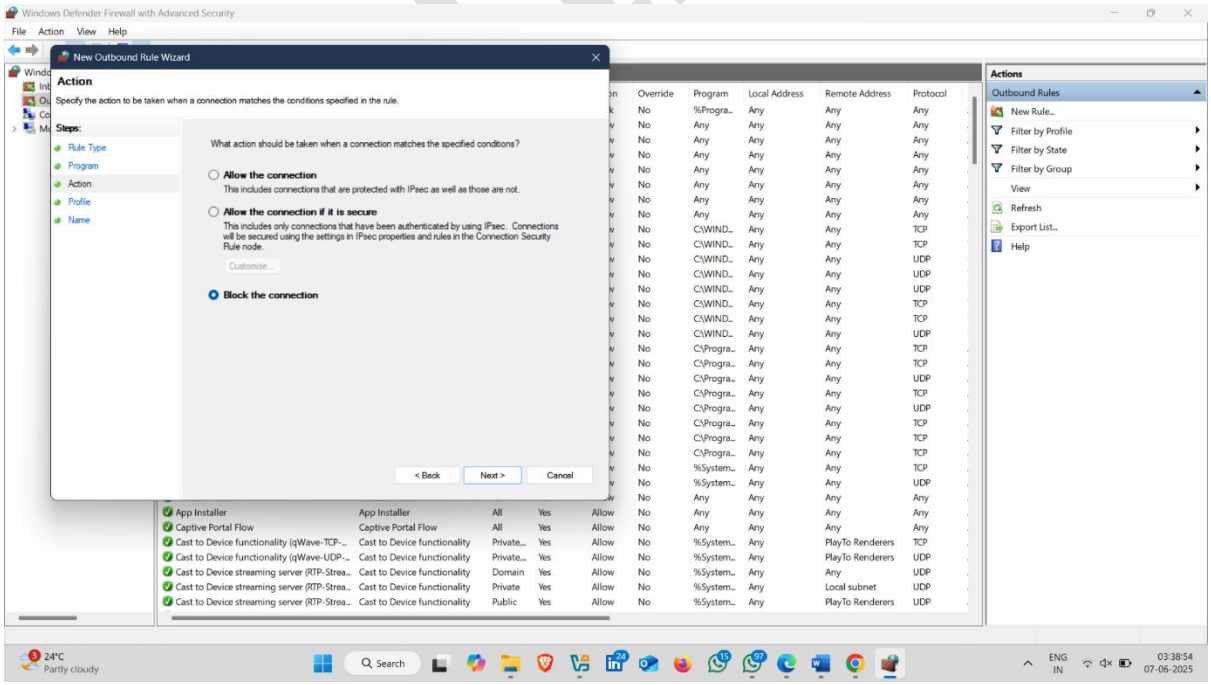
- 🔐 **Description: Only allows connections authenticated using IPsec (Internet Protocol Security).**

- 🔧 **Requires: Proper configuration of IPsec policies.**
- 📌 **Use When: You want to only permit secure (encrypted and authenticated) connections for sensitive systems or data transfer.**
- ⚠️ **Customize button is usually greyed out unless IPsec is configured.**

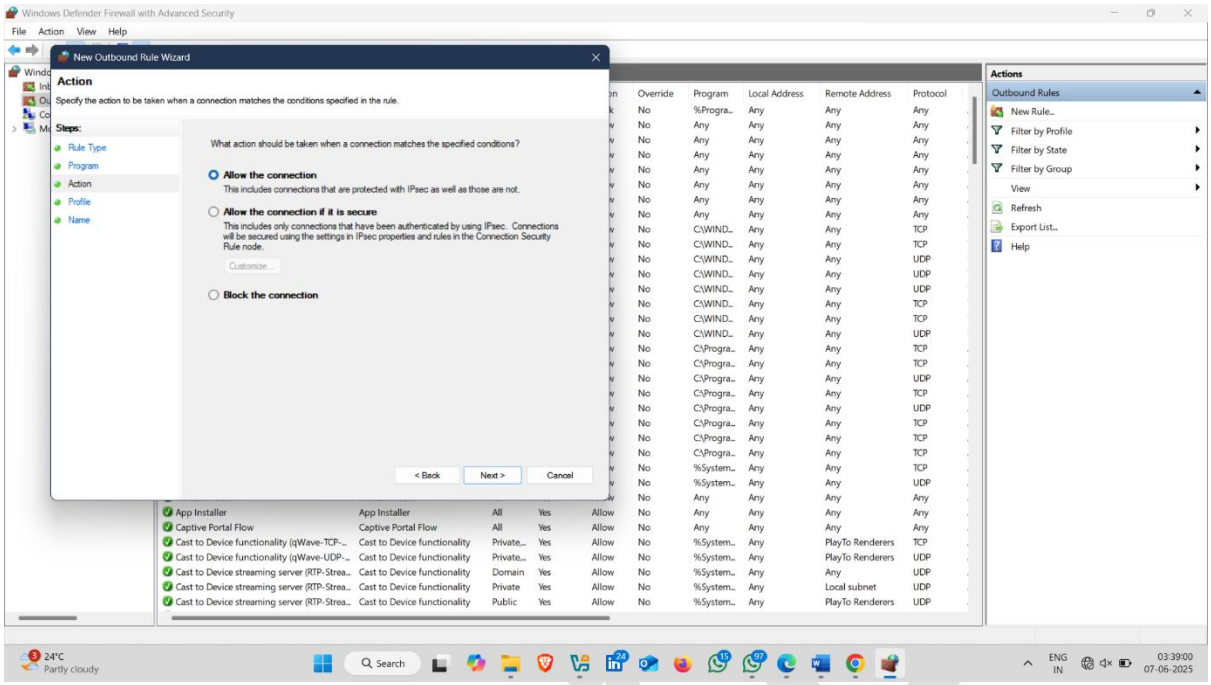
---

3. Block the connection

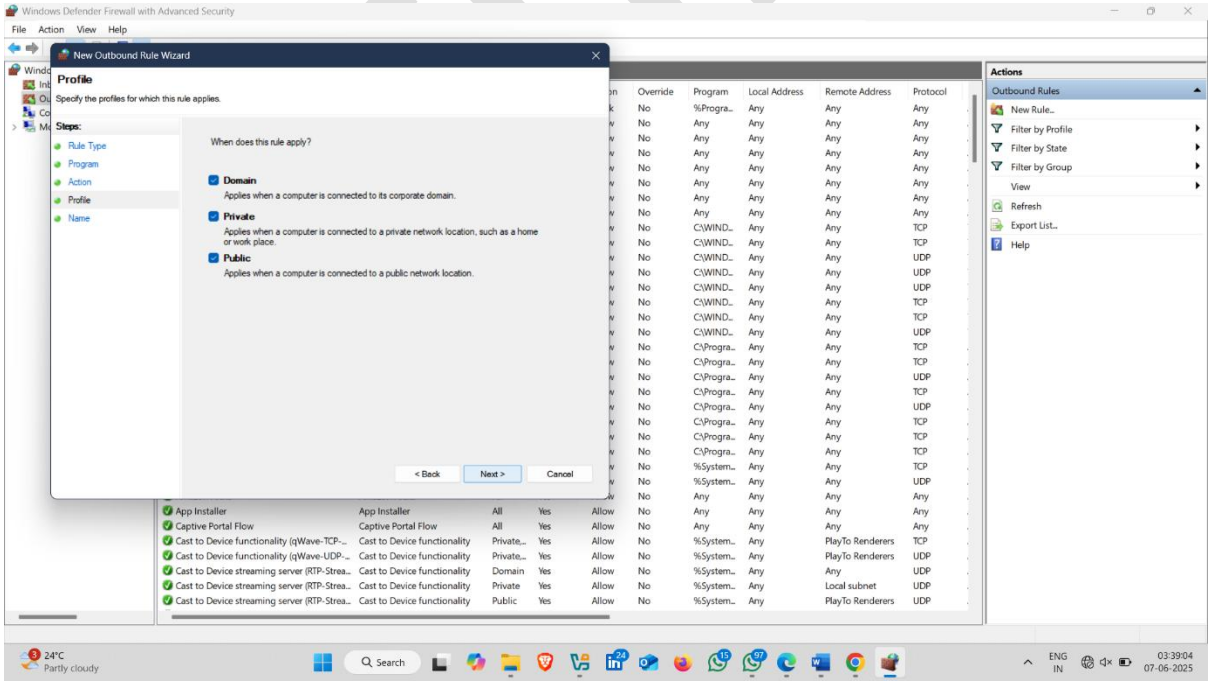
- ❌ **Description: Denies the connection completely, regardless of whether it's secure or not.**
- 🔒 **Security: Strictest setting; completely blocks the matched traffic.**
- ✅ **Use When: You want to prevent any communication through the specified ports, programs, or IP addresses.**



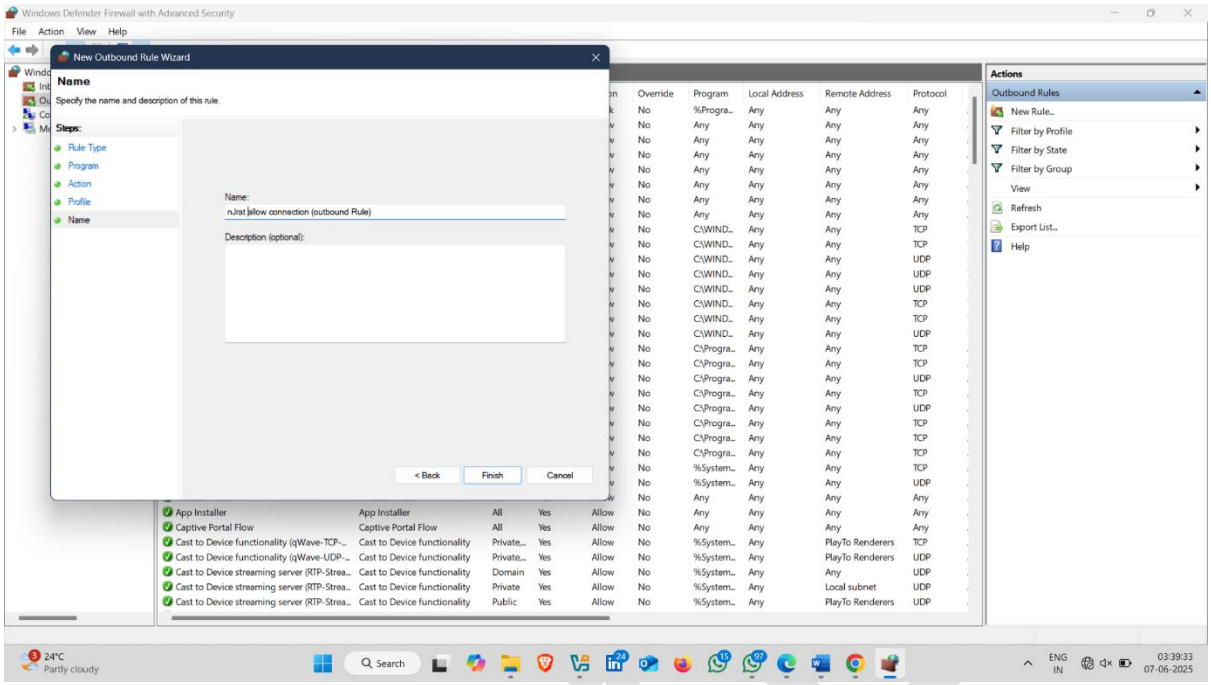
- **Select option and click on next**



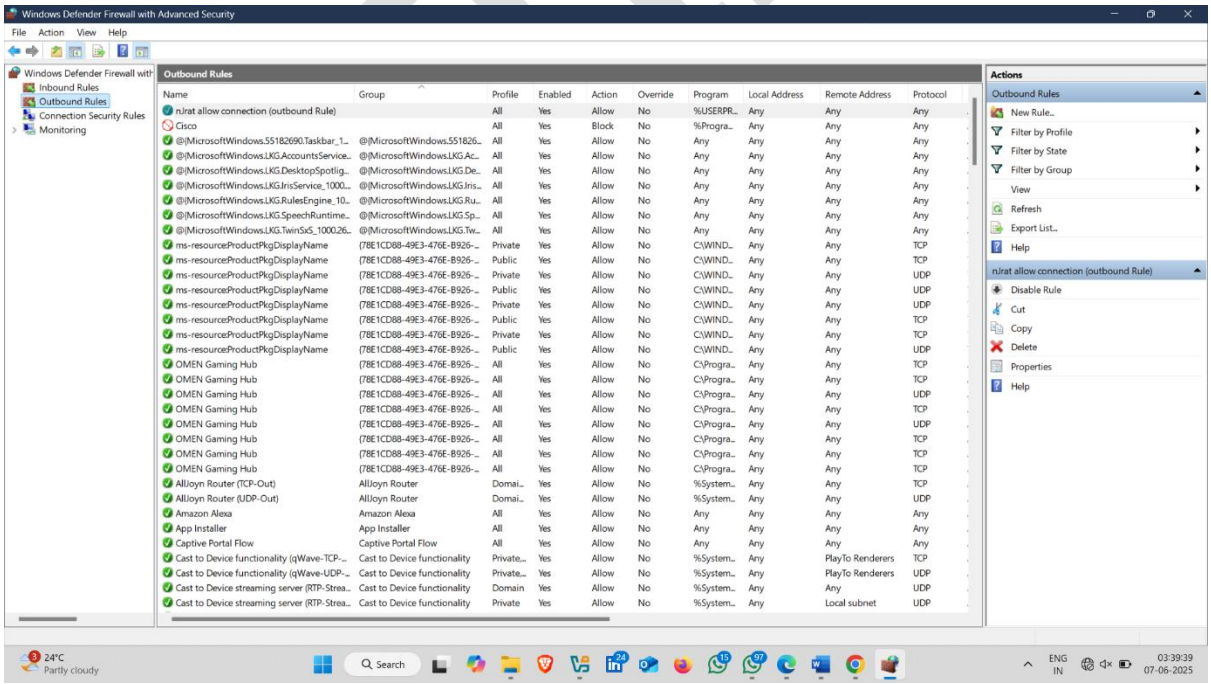
- Click on next



- Set the name of the Rule and description and click on next



- **Rule set successfully** ✅ 👇



EXTRA ACTIVITY

1.Zone Alarm Firewall

ZoneAlarm Firewall is a third-party **software firewall** for Windows developed by **Check Point Software Technologies**. It provides an extra layer of protection beyond the default **Windows Defender Firewall**, especially useful for users who want more detailed control over network activities.

🔐 **Key Features of ZoneAlarm Firewall:**

1. **Two-Way Firewall:**
    - **Monitors incoming and outgoing traffic.**
    - Blocks unauthorized inbound and outbound connections.
2. **Stealth Mode:**
    - Makes your PC **invisible** to hackers on the internet by hiding open ports.
3. **Application Control:**
    - Alerts you when programs try to access the internet.
    - Allows or blocks access on a per-application basis.
4. **Real-Time Monitoring:**
    - Tracks all network activity.
    - Shows what programs are using your internet.
5. **DefenseNet™:**

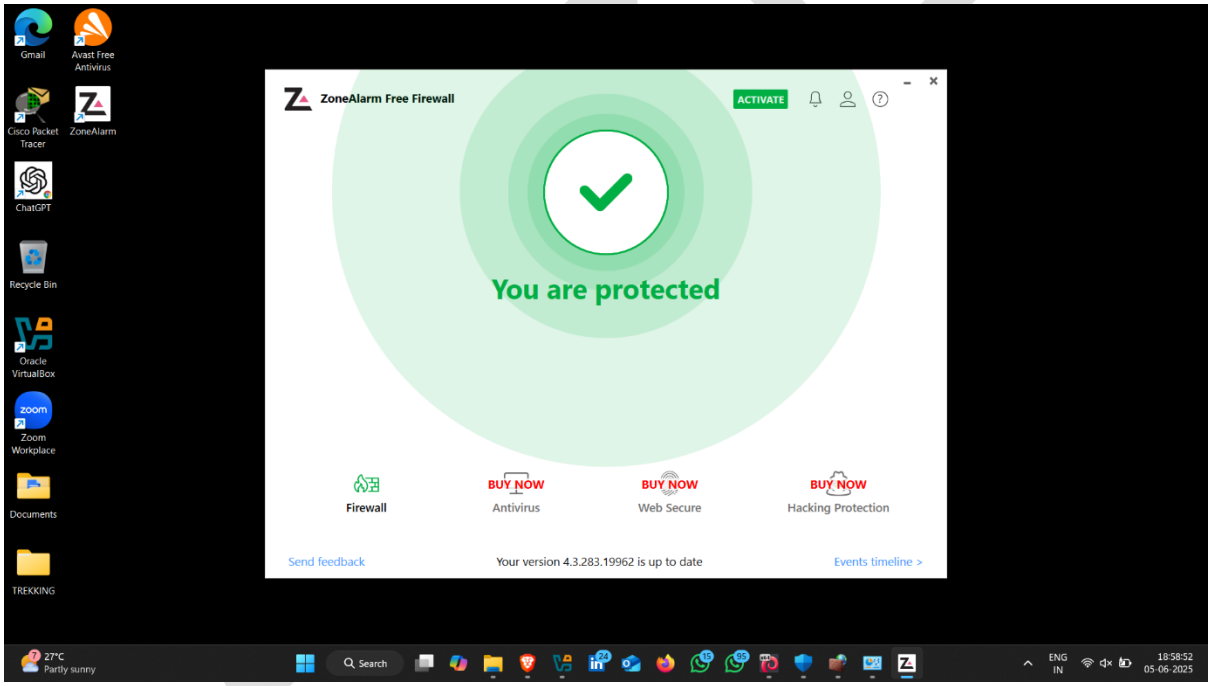
- A cloud-based feature that uses community feedback to **auto-configure** trusted apps.

6. **Identity Protection (Optional in Pro Version):**
   - Alerts you if personal data is at risk.
   - Includes credit monitoring features (U.S.-based users).

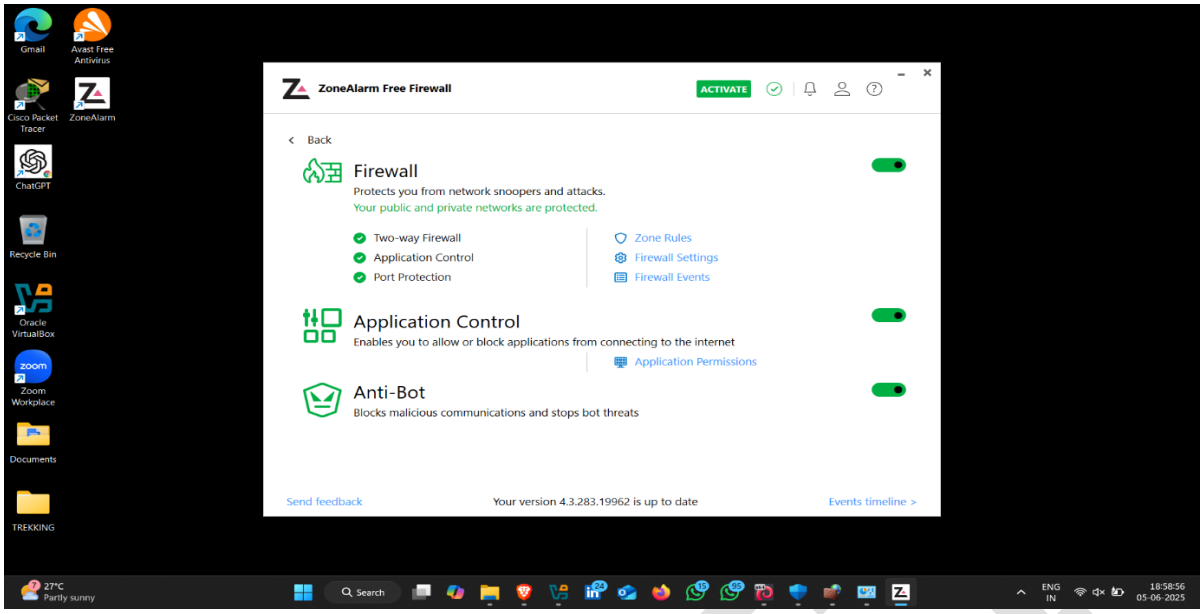
How to use it -:

After Installation , Open the application

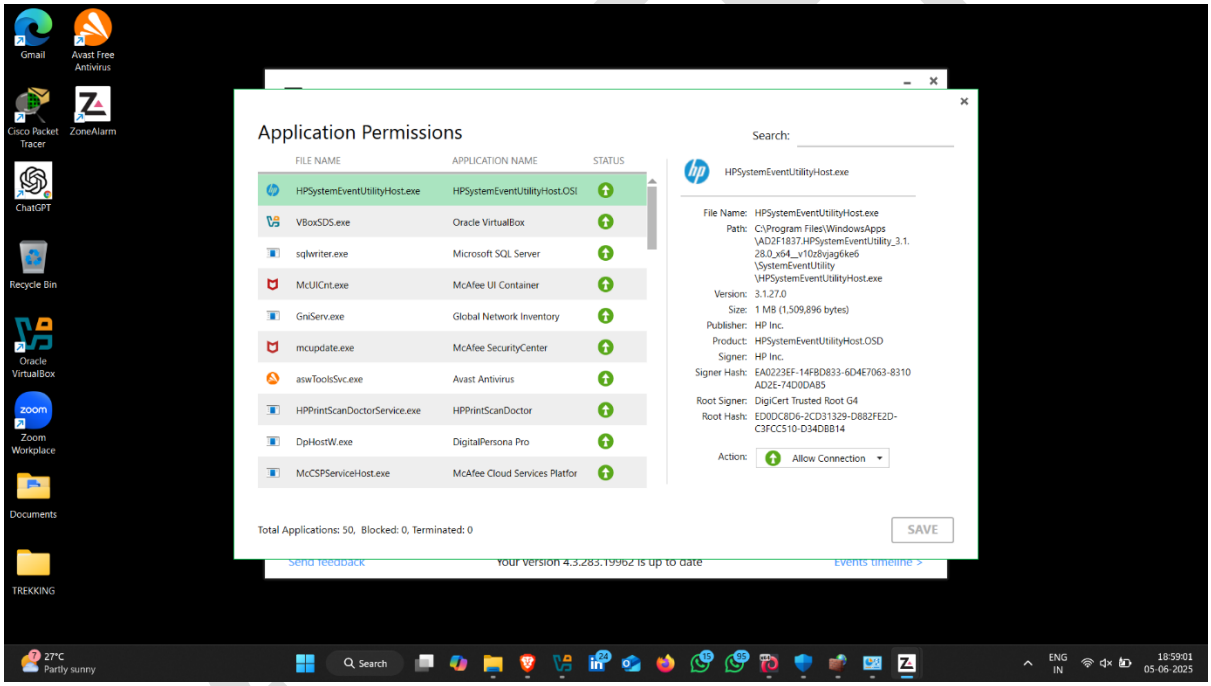
- Now click on firewall



- Click on **Application Permissions**



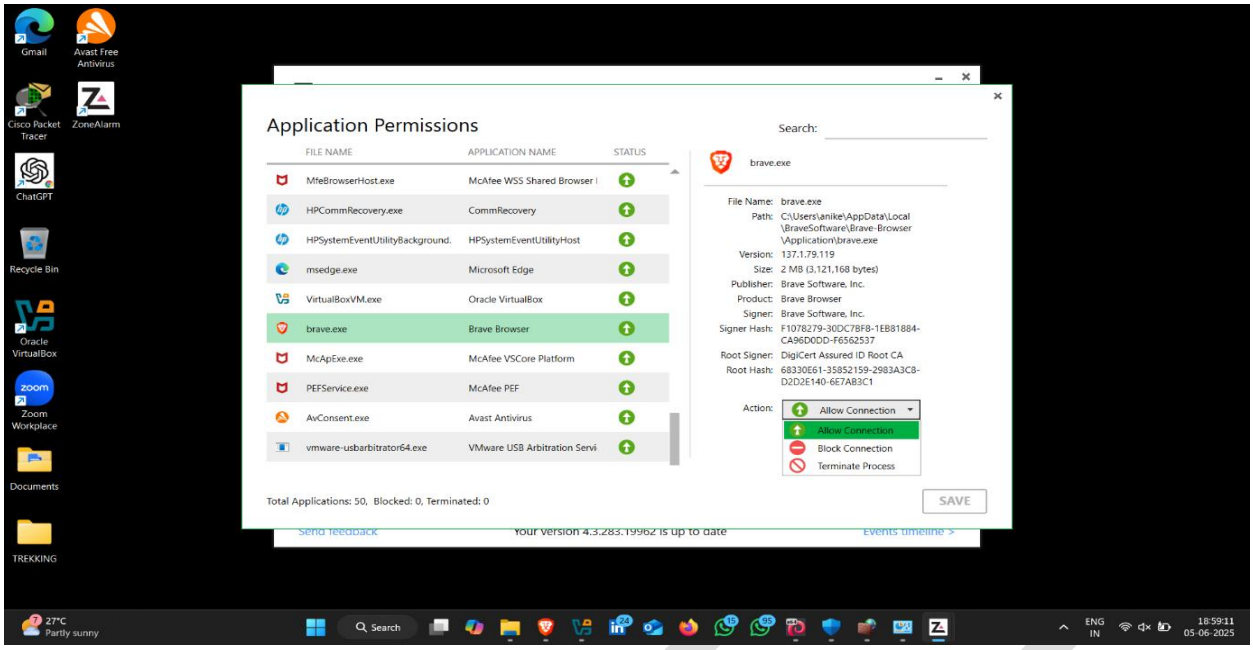
- Select the program that you want set a rule

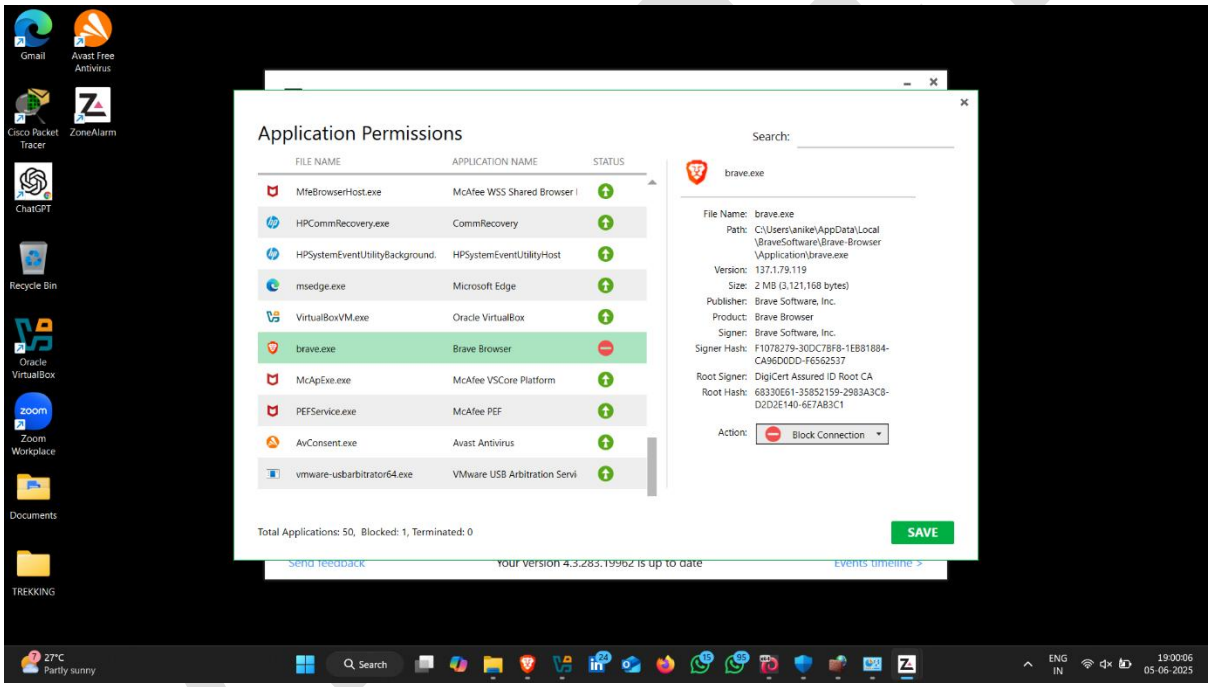


- Now , you can **allow connection** , **Block Connection** and **Terminate Process**

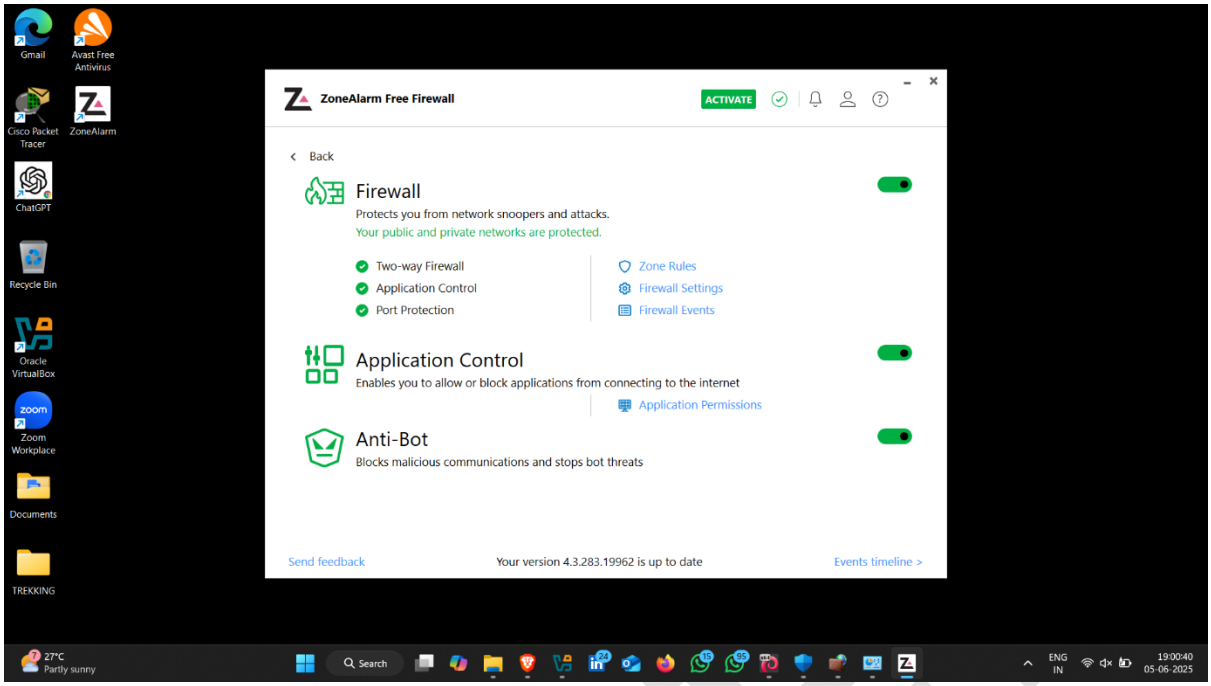
Action Dropdown:

- ✅ **Allow Connection** (Selected) – The firewall allows to access the internet.
- ⛔ **Block Connection** – Prevents from accessing the internet.
- 🛑 **Terminate Process** – Stops the process if it is running.

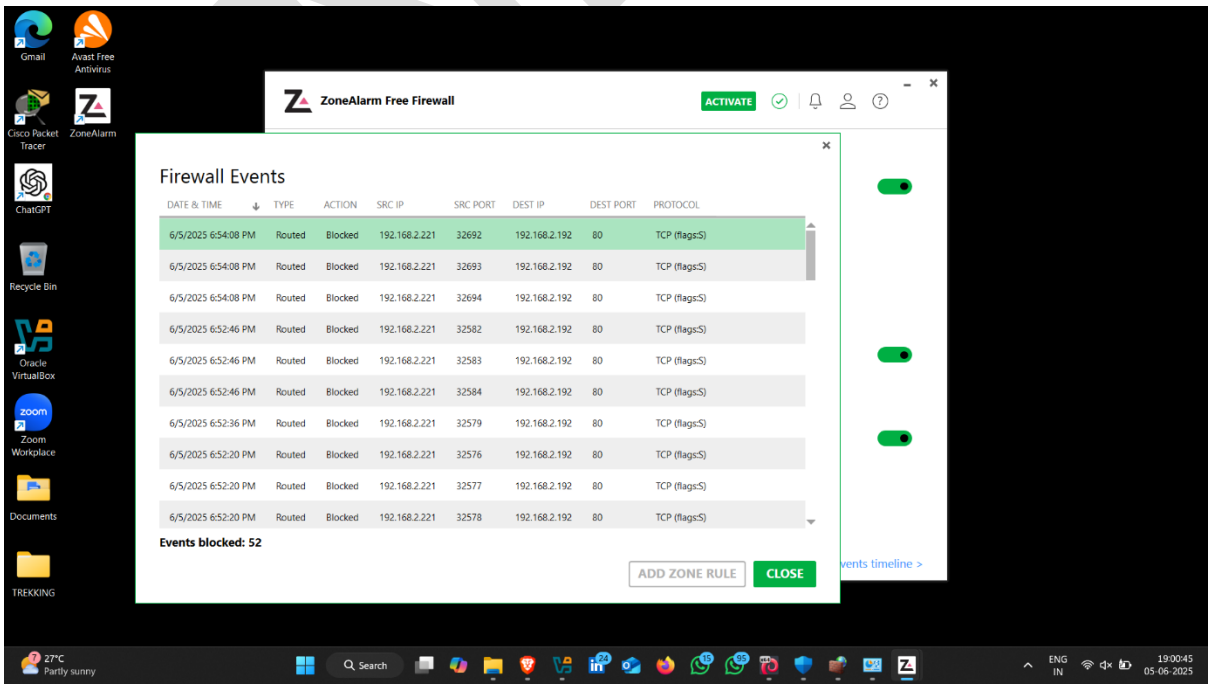




- Now click on Firewall Events that show the event logs that they capture during monitoring and capturing



- It show all the event



Attack Detection Tools

1.Perform Attack Detection Using HoneyBoT

HoneyBOT is a **Windows-based honeypot** software used for **cybersecurity monitoring** and **intrusion detection**. It simulates a vulnerable system or services to attract malicious attackers, allowing security professionals to **monitor, detect, and analyze attack behavior** in a controlled environment.

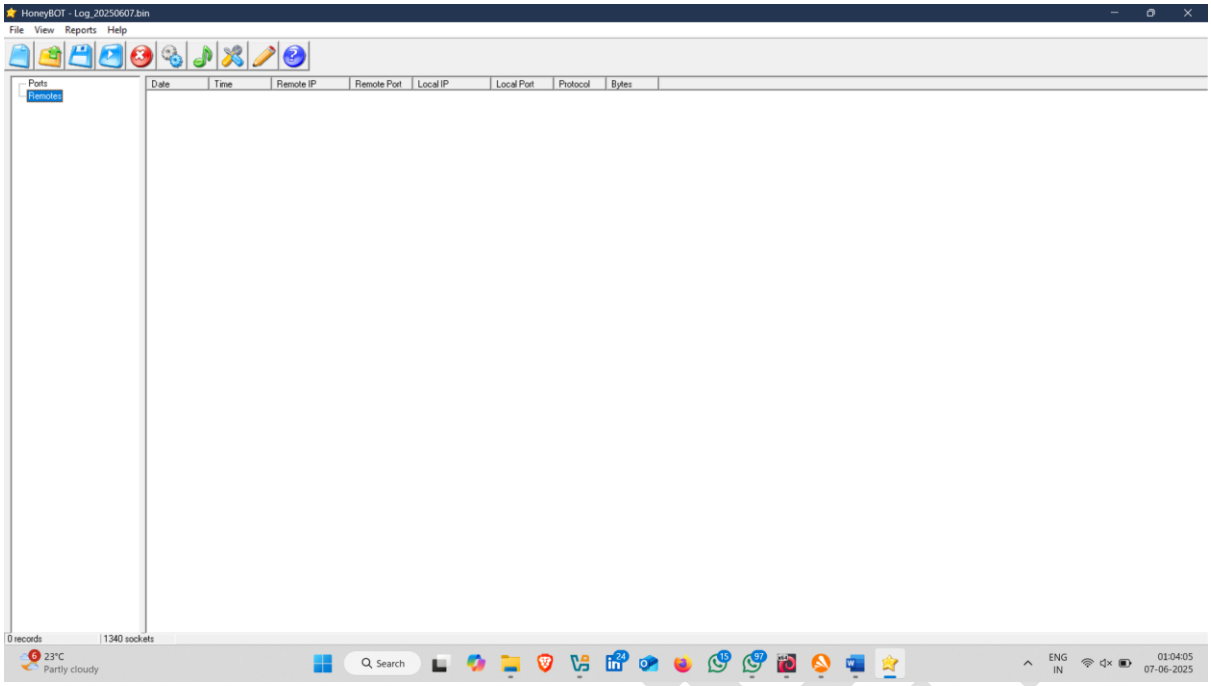
Common Use Cases:

- Detecting early signs of network intrusions.
- Learning about attackers' tactics and tools.
- Gathering intelligence on malware and exploits.
- Testing firewall and IDS/IPS effectiveness.

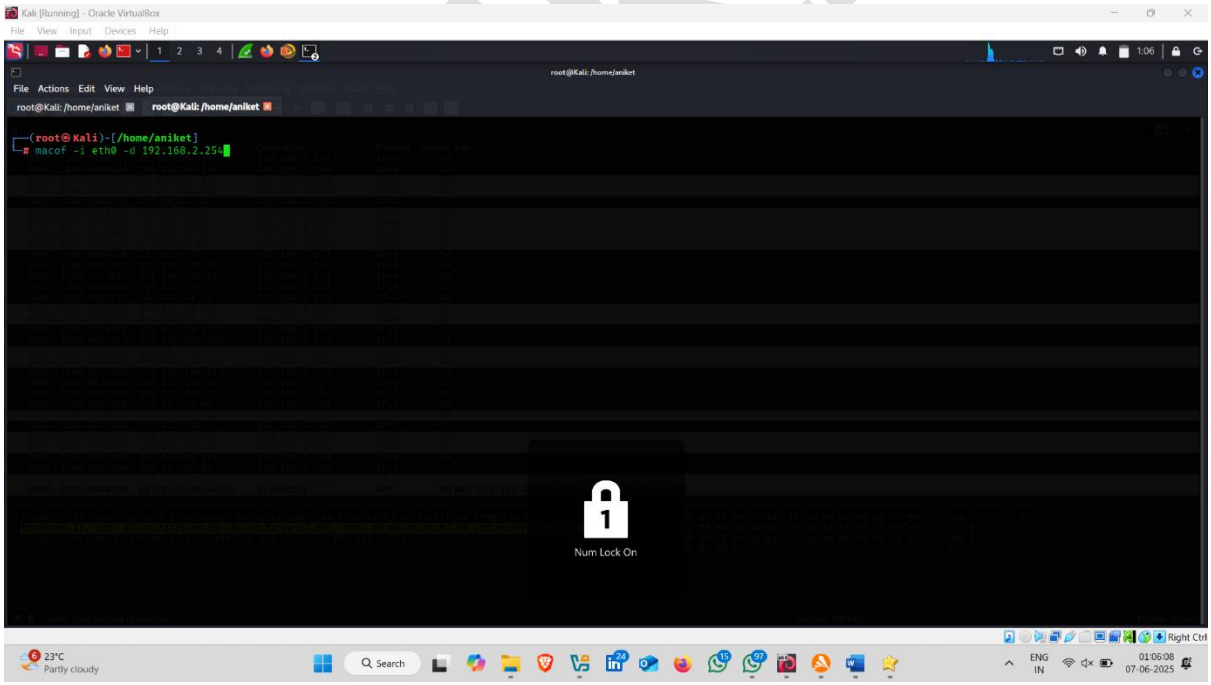
Download Link-: https://honeybot.software.informer.com/download/

How to use it -:

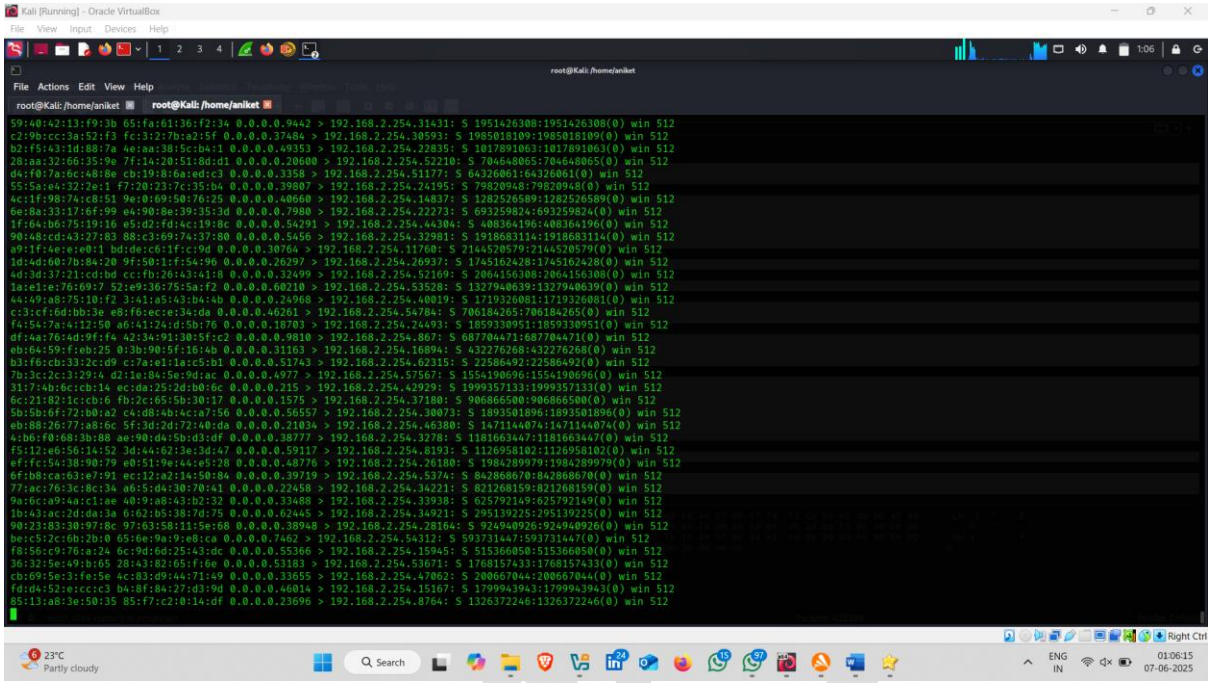
After installation HoneyBot Open it



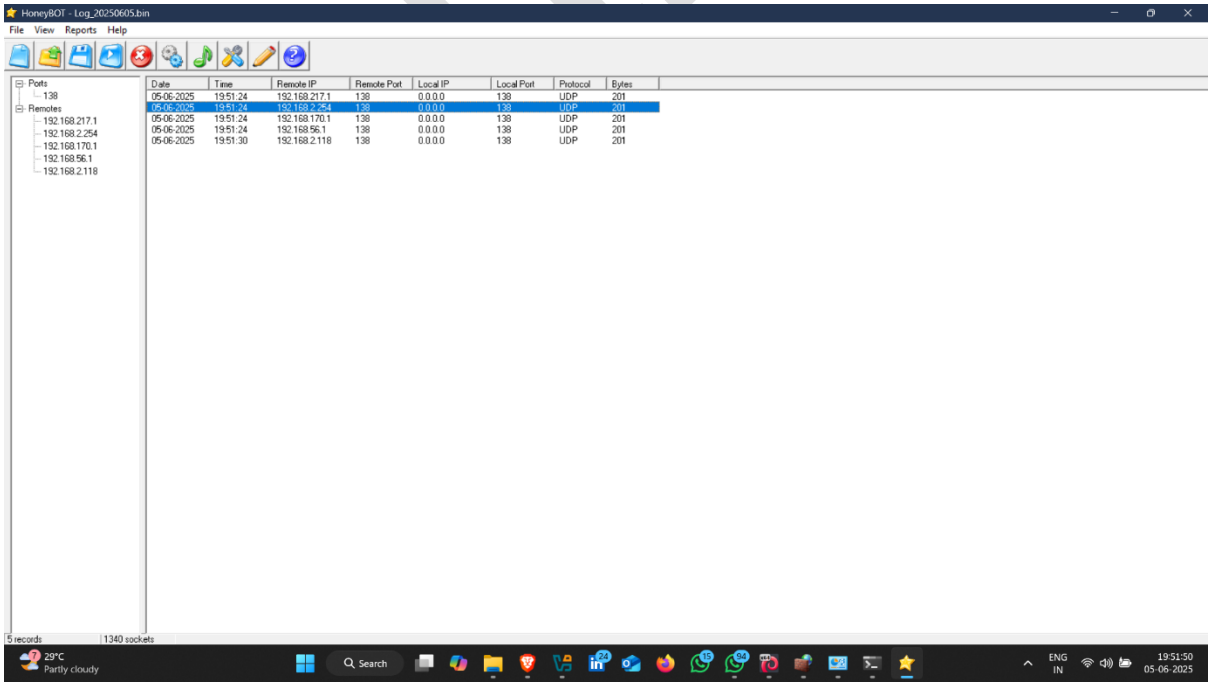
- Now open kali linux and start attack on your machine

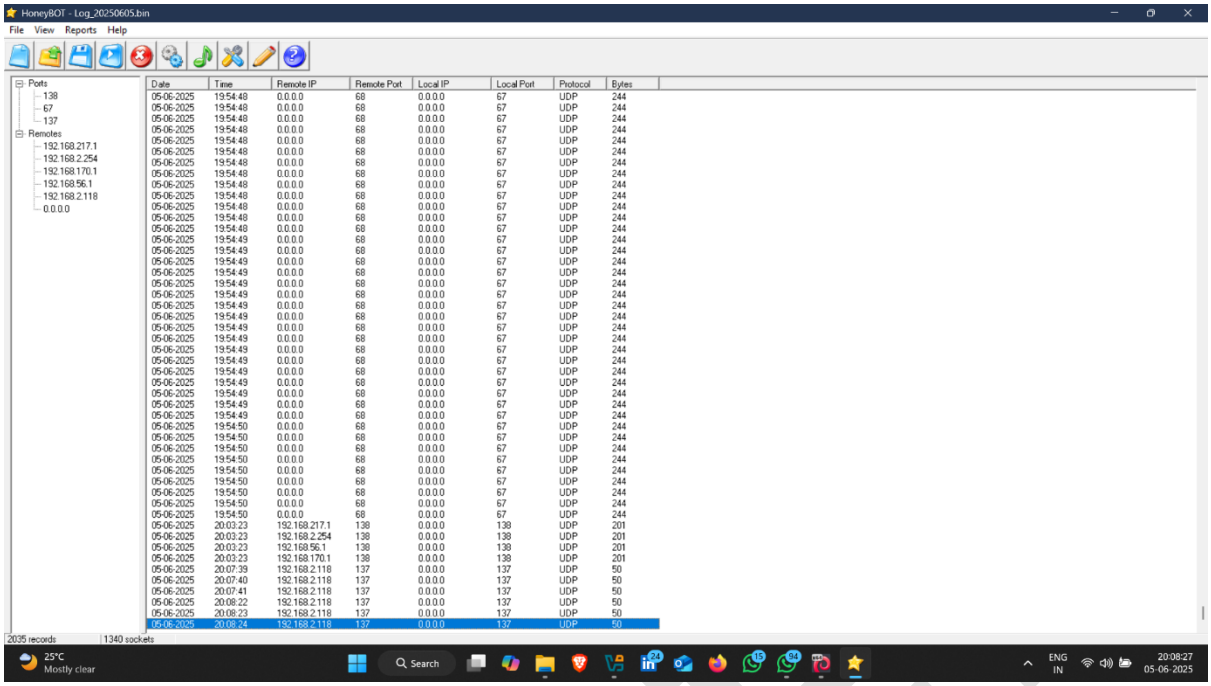


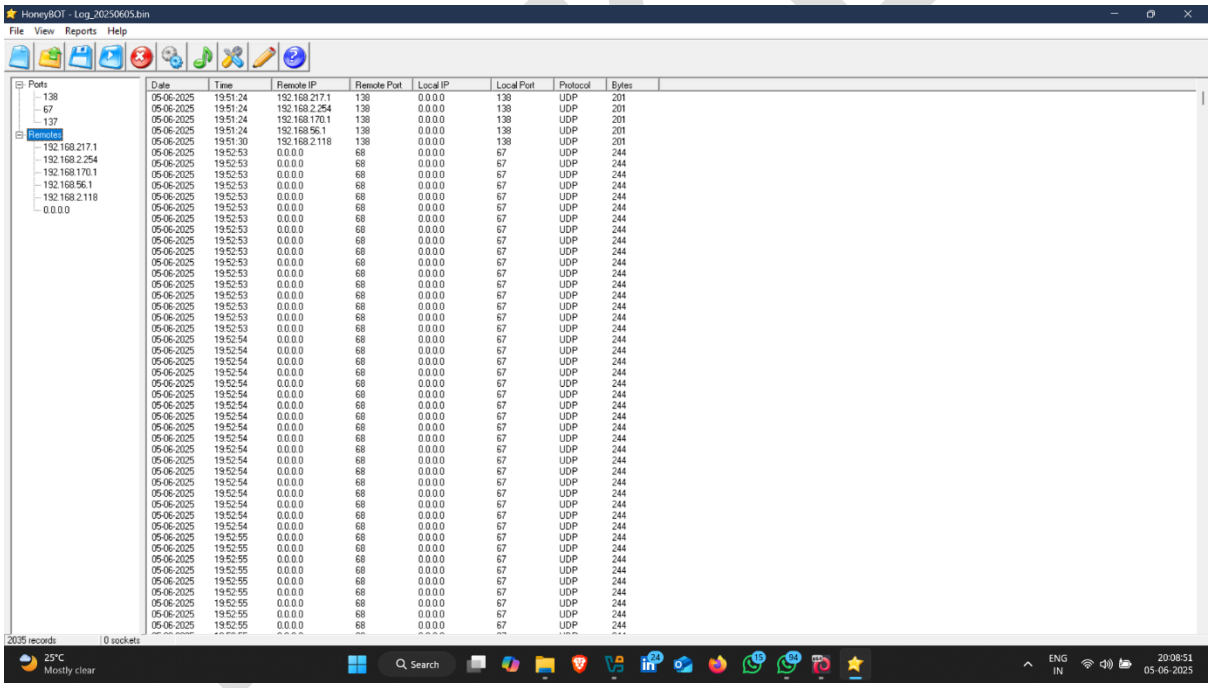
- Attack started 👇



- Detection started




HoneyBOT - Log_20250605.bin
File View Reports Help
Ports
138
67
137
Remotes
192.168.217.1
192.168.2.254
192.168.170.1
192.168.56.1
192.168.2.118
0.0.0.0
Date Time Remote IP Remote Port Local IP Local Port Protocol Bytes
2035 records
1340 sockets
25°C
Mostly clear
Search
ENG
IN
20:08:27
05-06-2025


HoneyBOT - Log_20250605.bin
File View Reports Help
Ports
138
67
137
Remotes
192.168.217.1
192.168.2.254
192.168.170.1
192.168.56.1
192.168.2.118
0.0.0.0
Date Time Remote IP Remote Port Local IP Local Port Protocol Bytes
2035 records
0 sockets
25°C
Mostly clear
Search
ENG
IN
20:08:51
05-06-2025

2.Perform Attack Detection Using KFSensor

KFSensor is a **commercial honeypot-based Intrusion Detection System (IDS)** developed for Microsoft Windows environments. It simulates vulnerable network services to attract and detect malicious activity, such as port scanning, malware infections, and exploitation attempts.

✅ Platform: Windows
✅ Type: Honeypot + IDS

🧩 Key Features of KFSensor

- Multiple Protocol Support
- Stealth Operation
- Custom Services
- Payload Capture
- GeoIP Location
- Windows Integration
- Alerting
- Log Analysis
- Plugins
- Real-time Monitoring
- Web-based Attack Detection
- Email Notification
- Syslog Integration
- Service Banner Customization
- Port Scanning Detection
- Fake File Server Emulation
- Attack Signature Matching
- Attack Source Identification
- Easy GUI Interface
- Exportable Reports

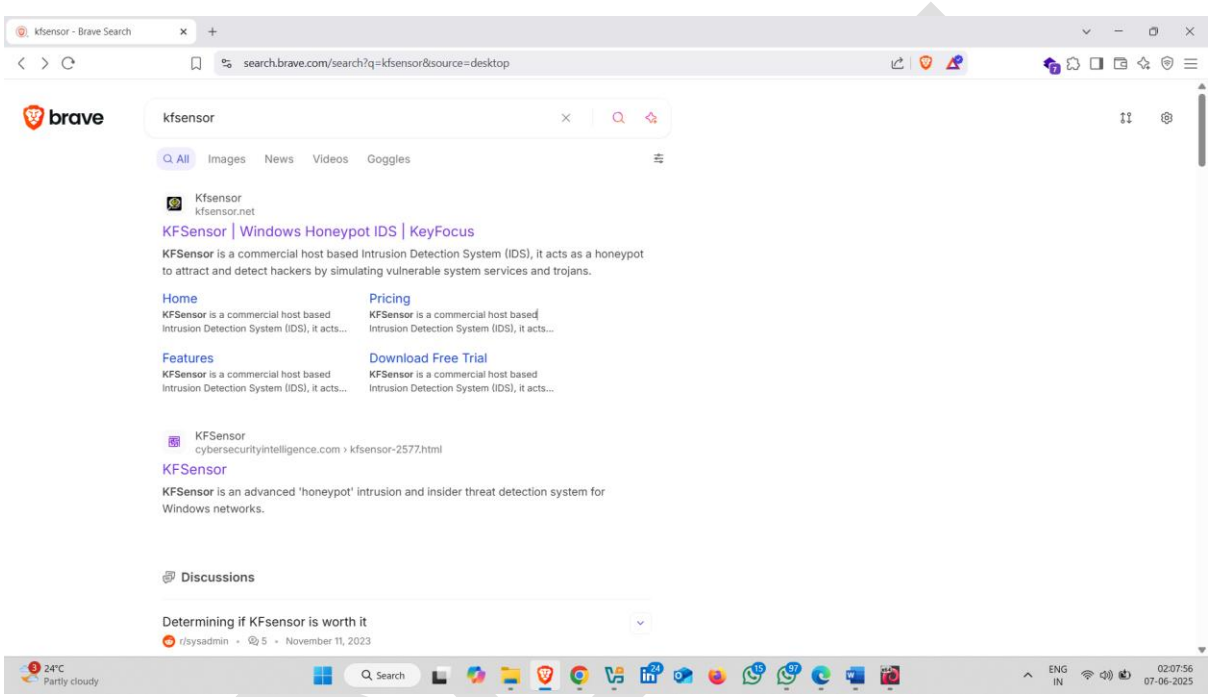
Download Link -: https://www.kfsensor.net/

How to download it -:

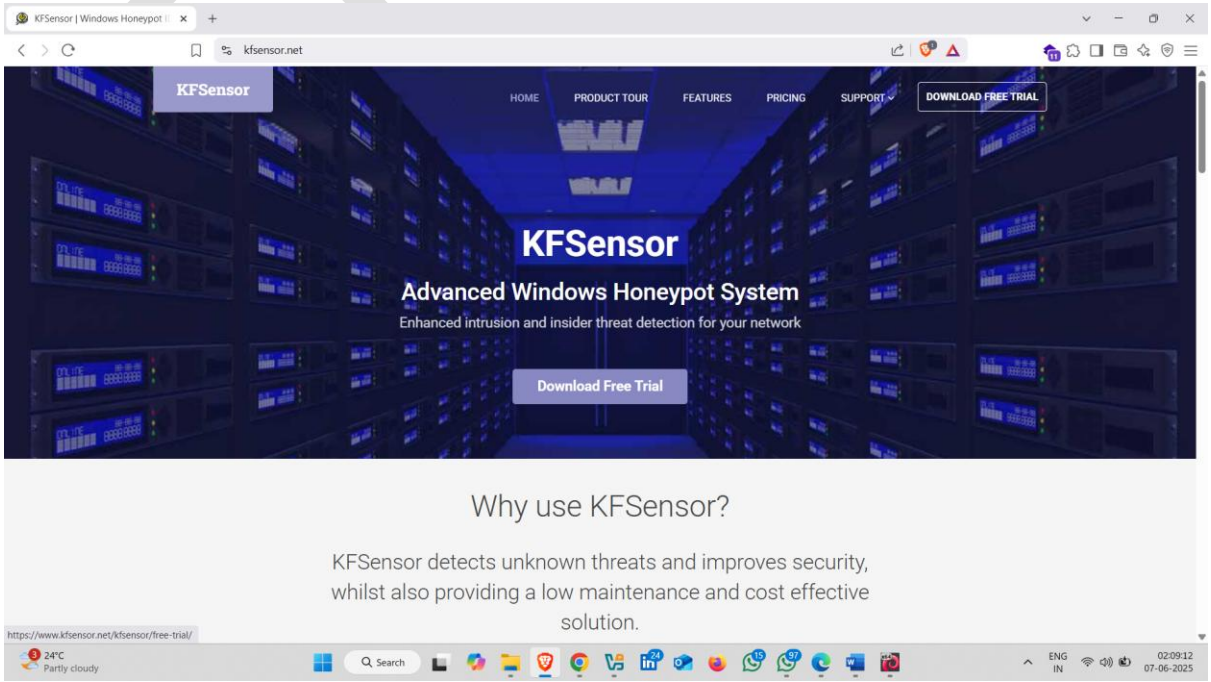
- Simply open the browser and search kfsensor download

Note-: kfsensor is a paid tool , Download its free trail Version

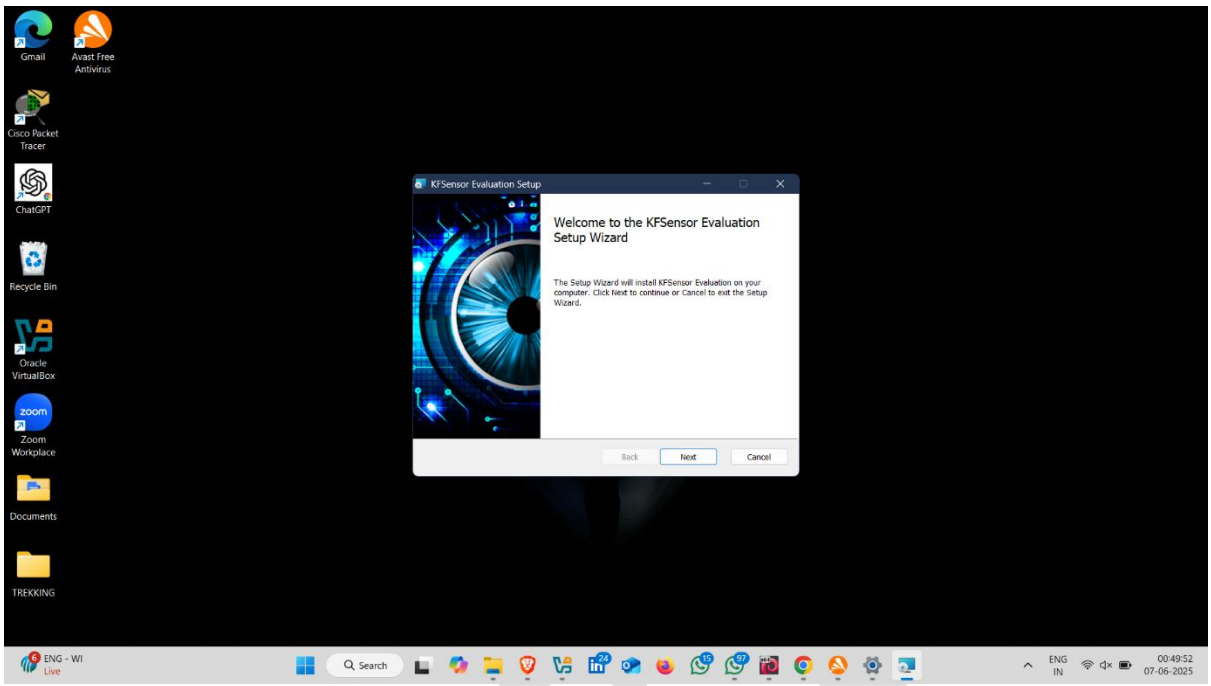
- Open First official Website 👇



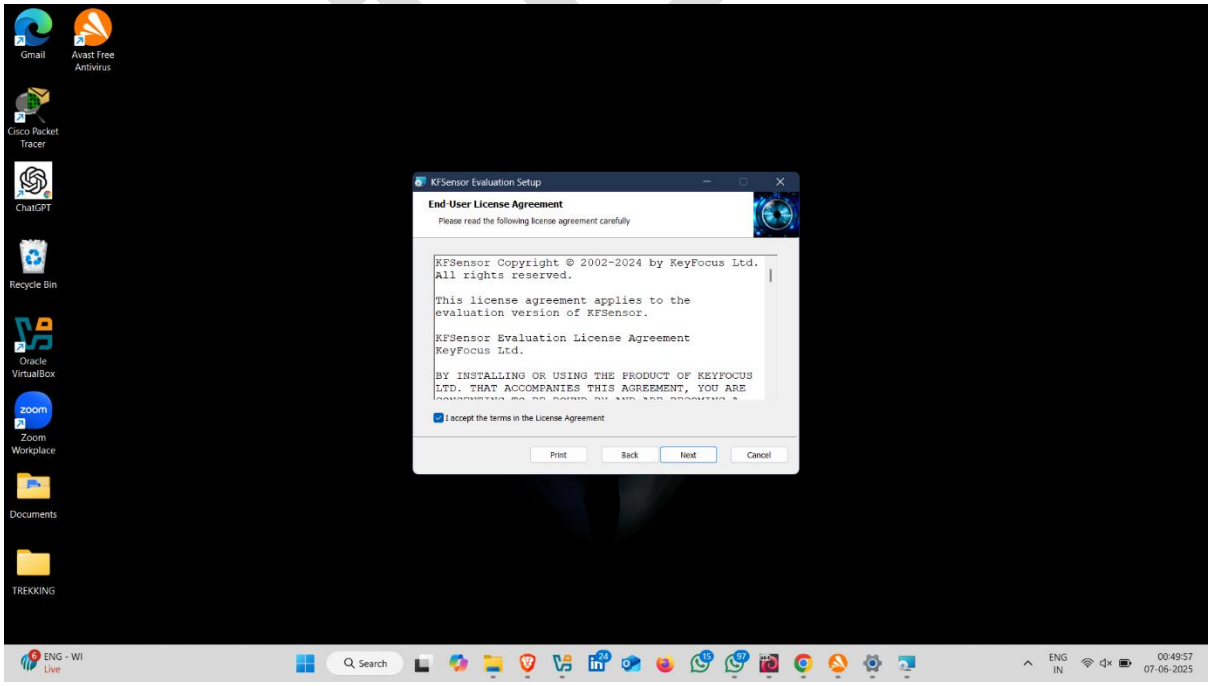
- Click on **Download Free Trial** and Download It



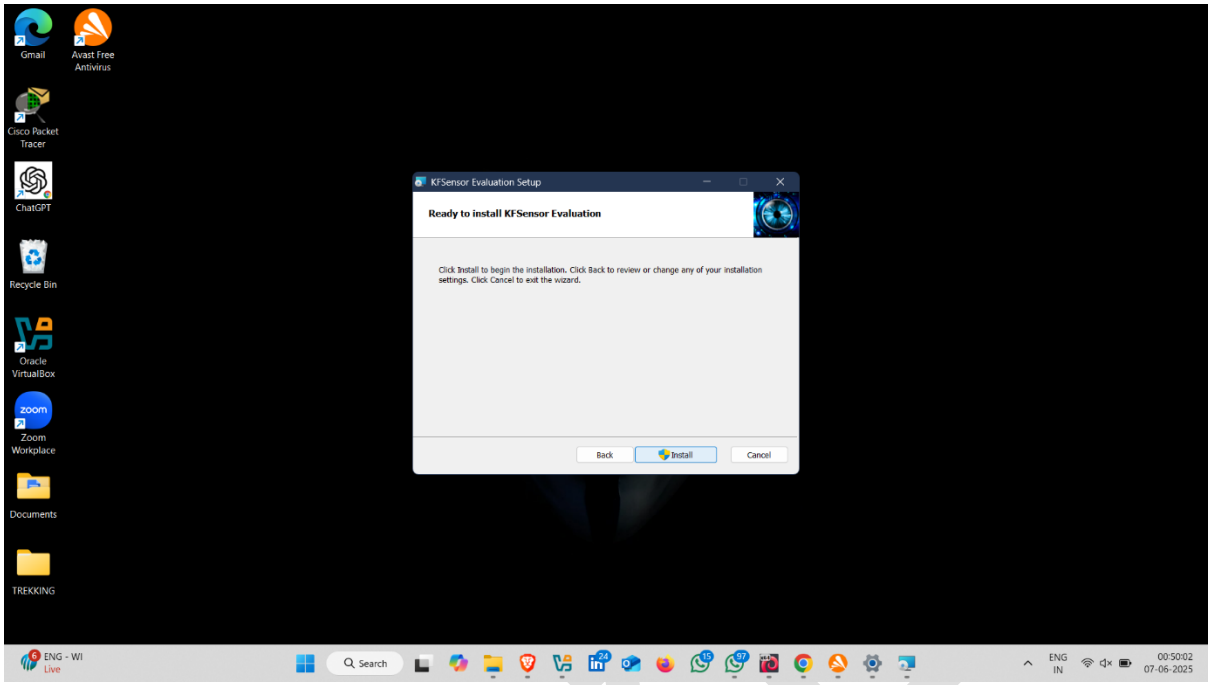
- After installation open the kfsensor.
- Click on Next



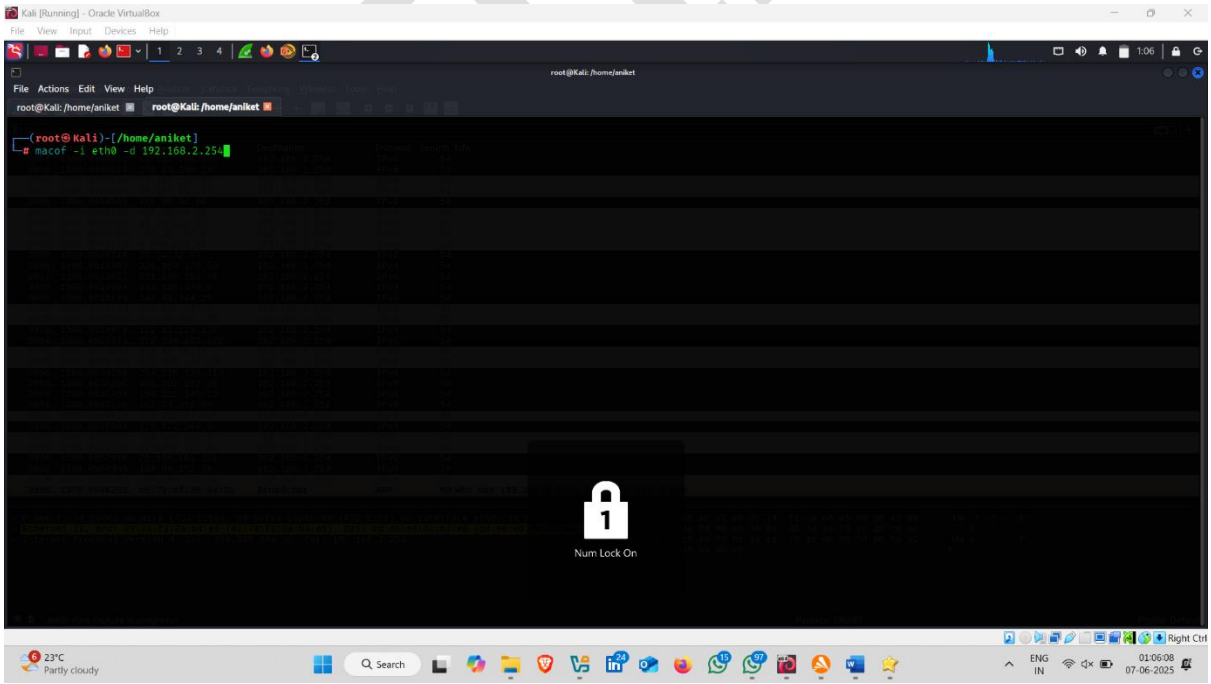
- Now accept the license and then click on next



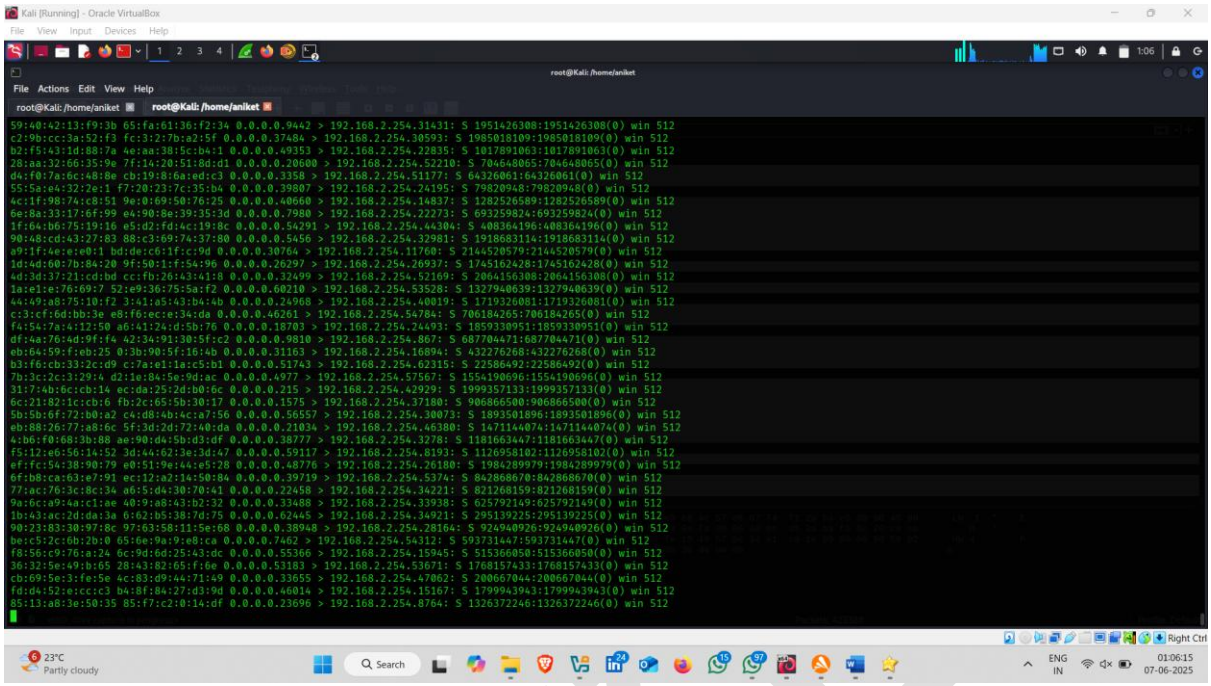
- Click on Finish



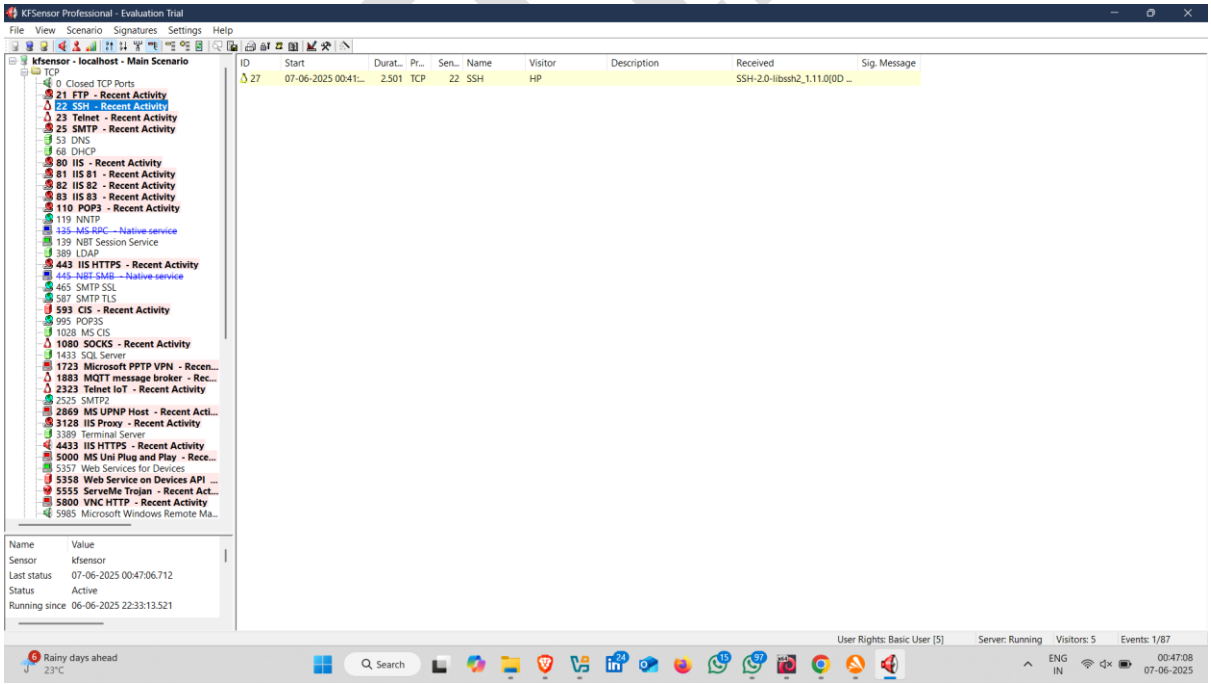
- Now open kali linux virtual machine and start attack 👇

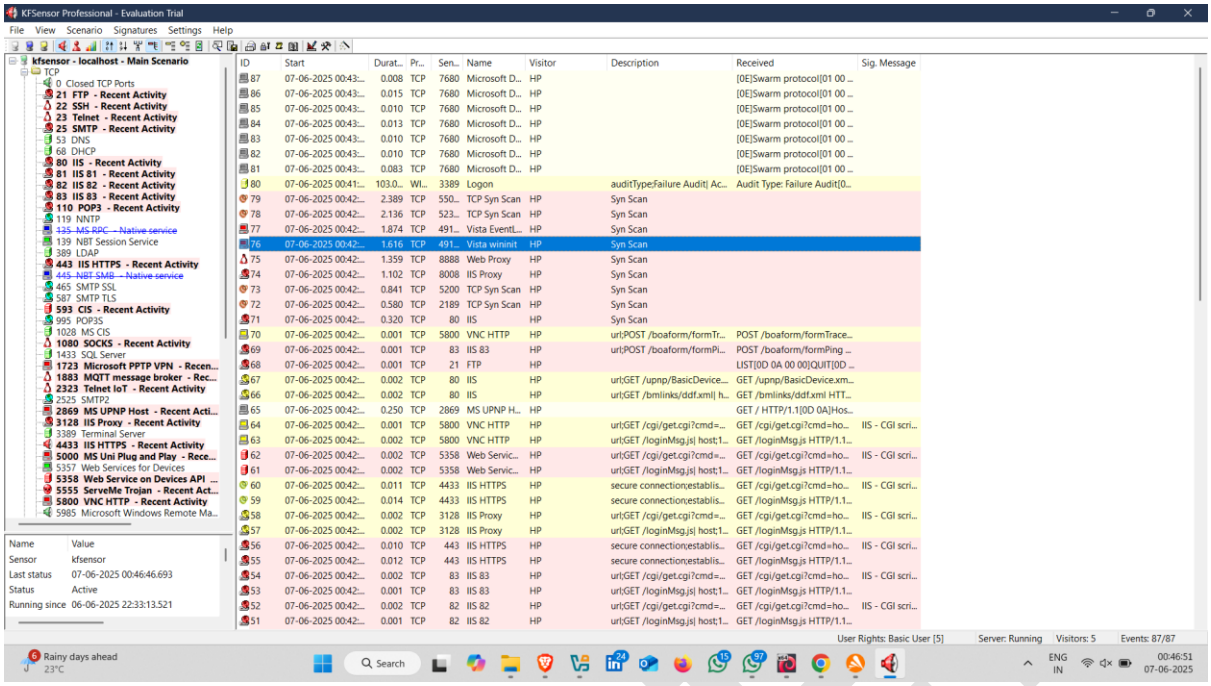


- Attack Started 👇



- Go back to the kfsensor , here detection started





3.Perform Attack Detection Using Wireshark

Wireshark is a **free and open-source network protocol analyzer** used to capture and inspect packets in real time across networks. It's used for **network troubleshooting, security analysis, and protocol development.**

🔧 Wireshark Features :

1. **Packet Capture**
2. **Protocol Analysis**
3. **Display Filters**
4. **Capture Filters**
5. **Color Coding**
6. **Promiscuous Mode**

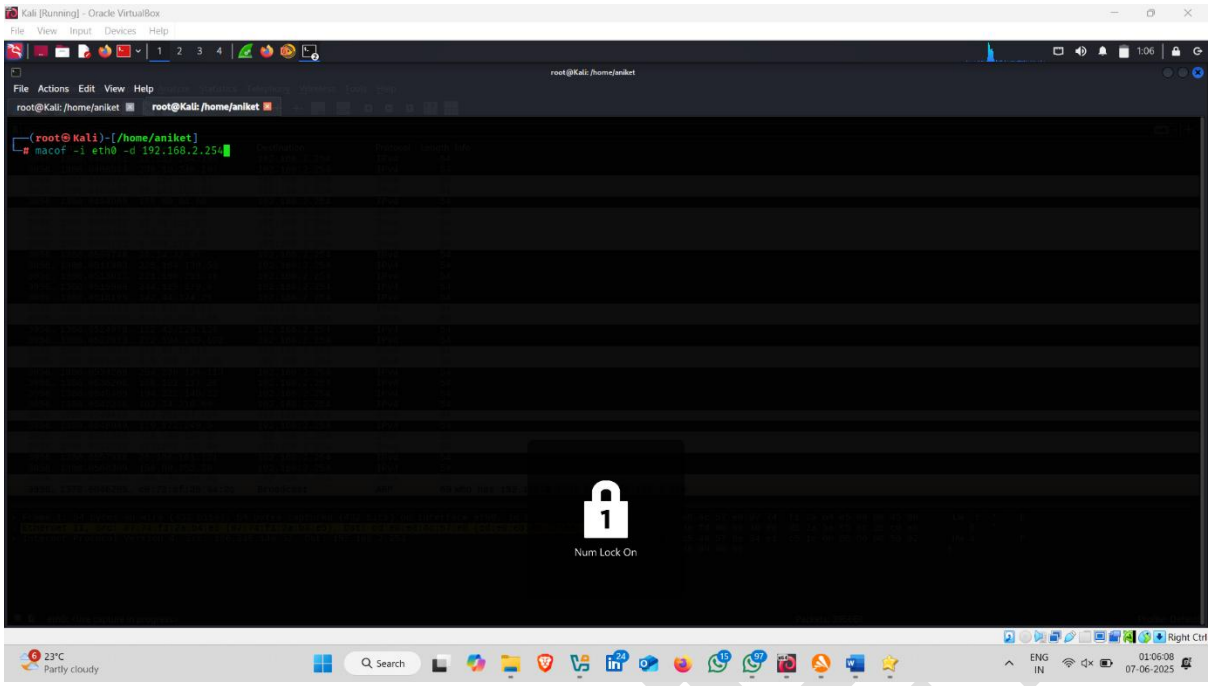
7. **Follow TCP/UDP Stream**
8. **Live Traffic Monitoring**
9. **Packet Reassembly**
10. **Expert Information**
11. **Name Resolution**
12. **Multiple Interface Capture**
13. **Command-Line Support (TShark)**
14. **Packet Export**
15. **Custom Columns**
16. **VoIP Analysis**
17. **Decryption Support**
18. **Statistics Tools**
19. **Cross-Platform Support**
20. **Free & Open Source**

How to use wireshark as a detection tool-:

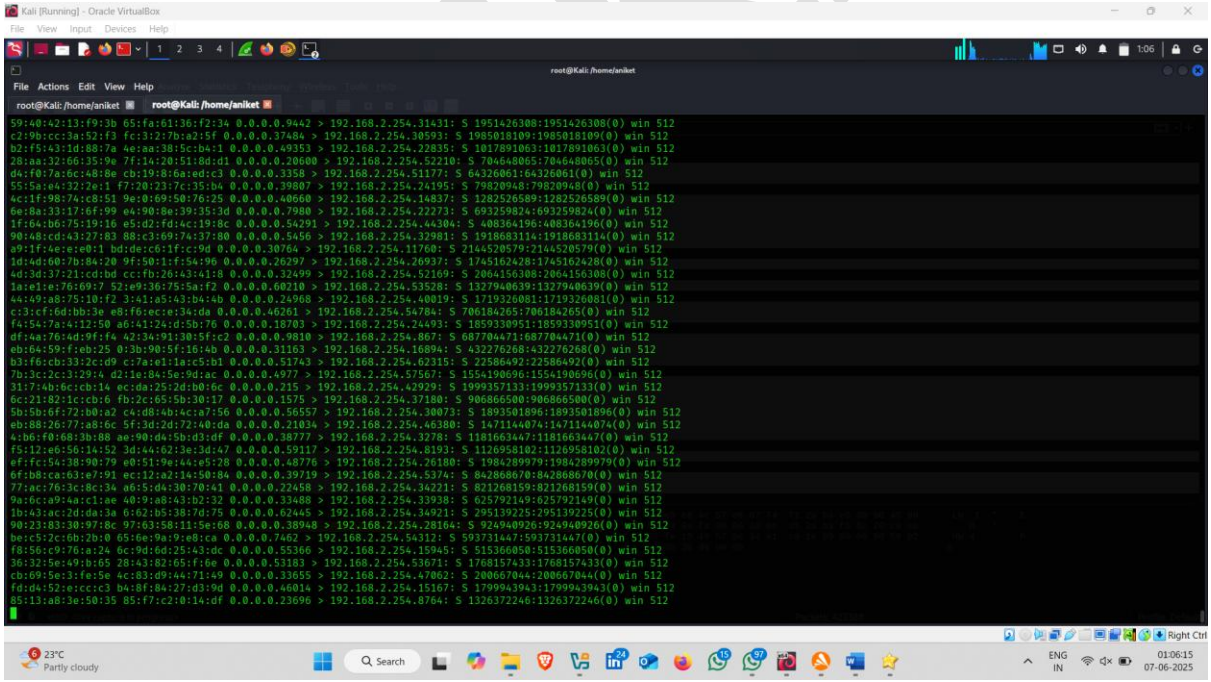
Attacker machine – Kali Linux

Victim Machine – Windows 11

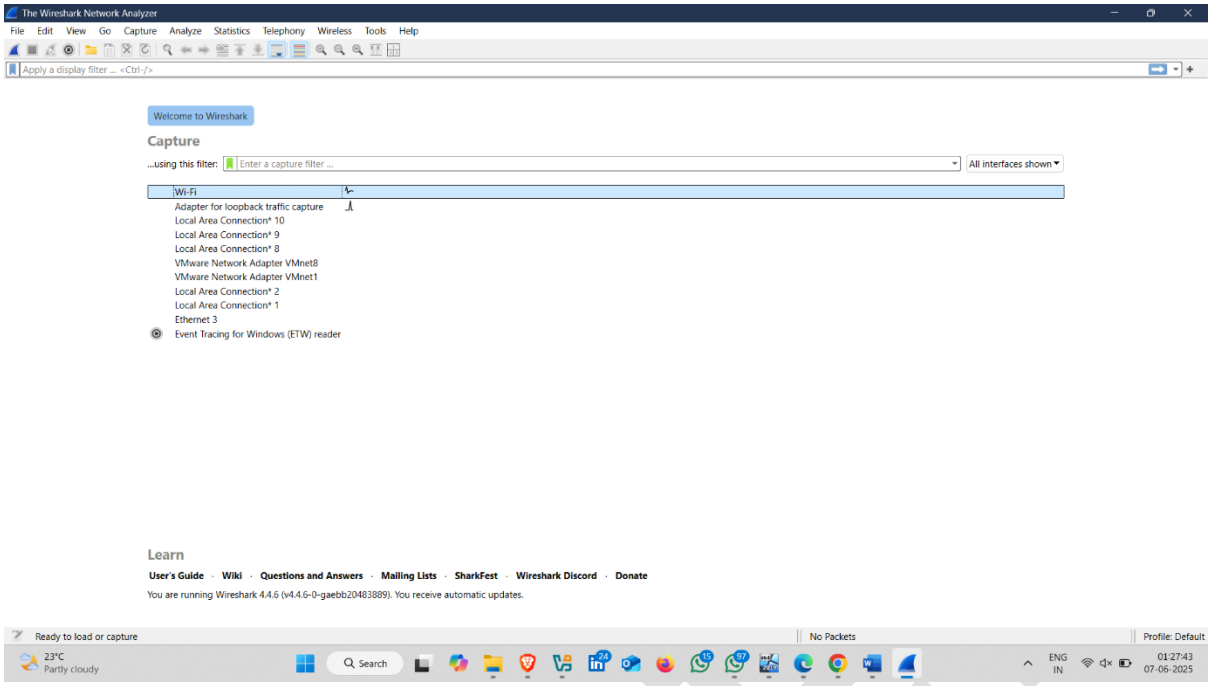
- Open Kali linux for Attack on Target
- Attack on target 👇



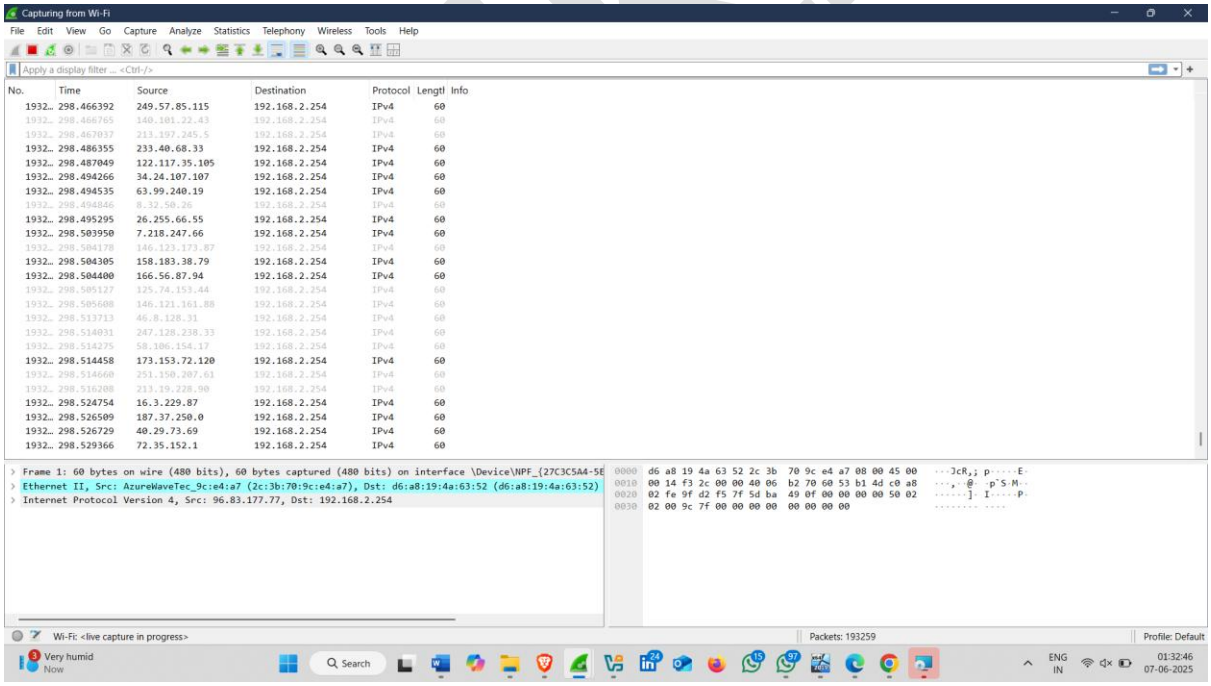
- **Attack Started ✅ 👇**



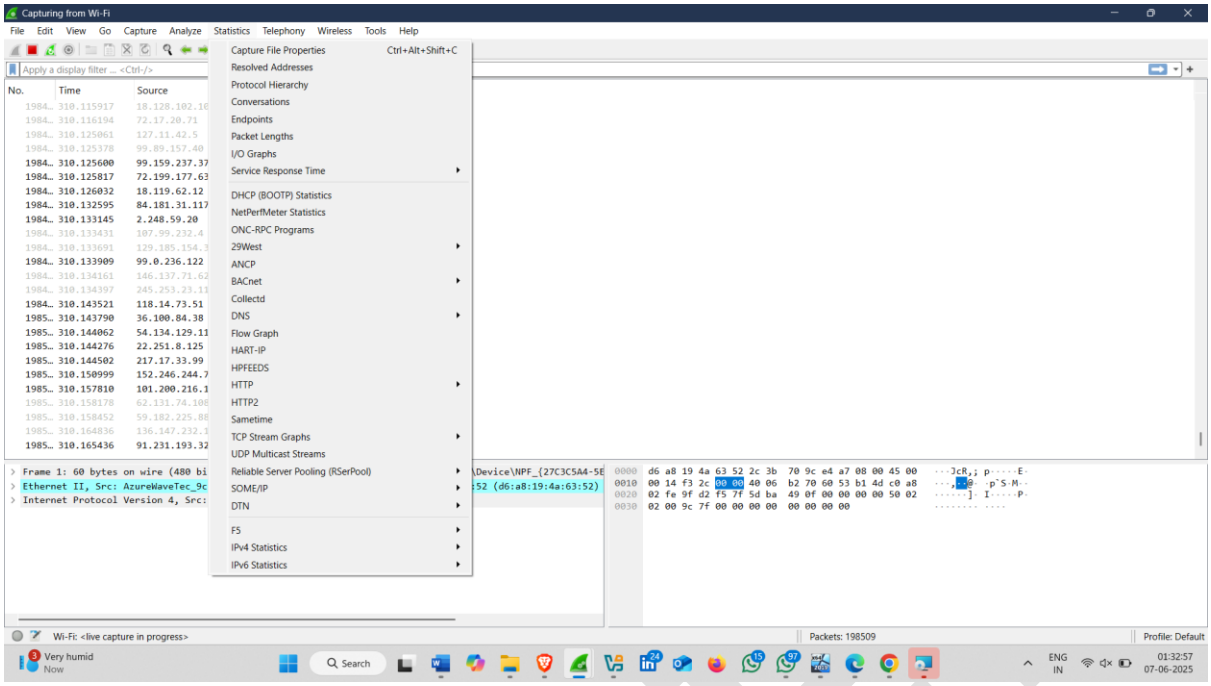
Now Back to the Victim machine and open Wireshark



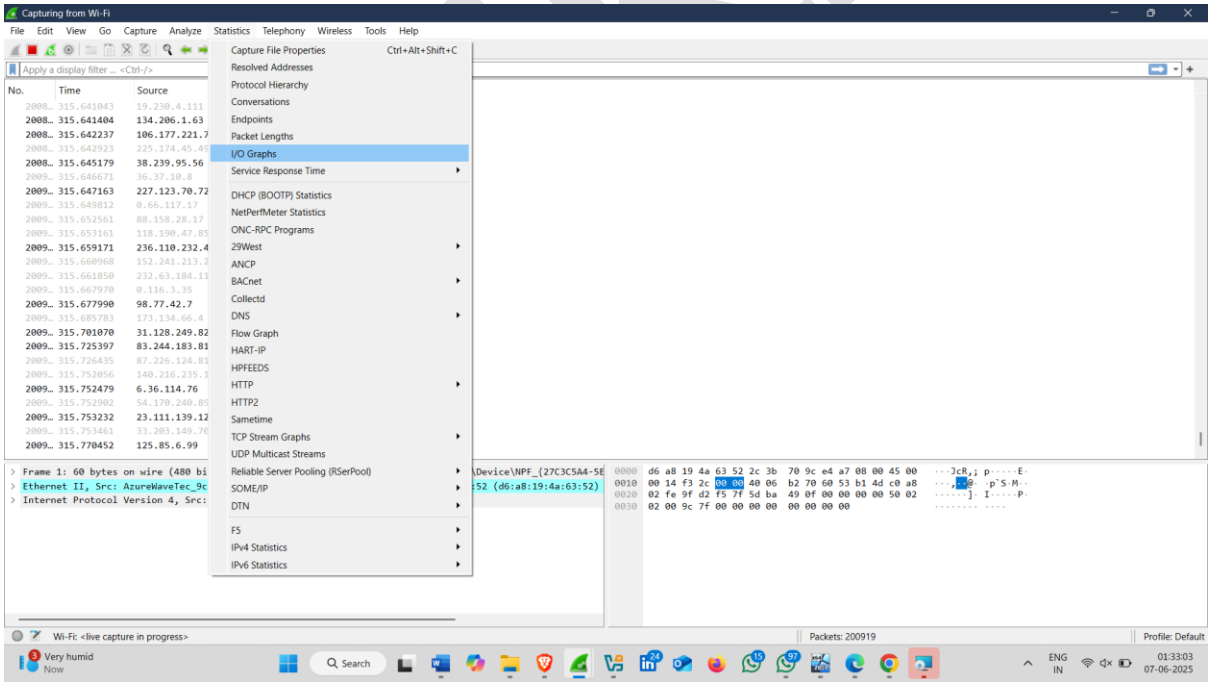
- Here , unknown flood of packets are received and captured



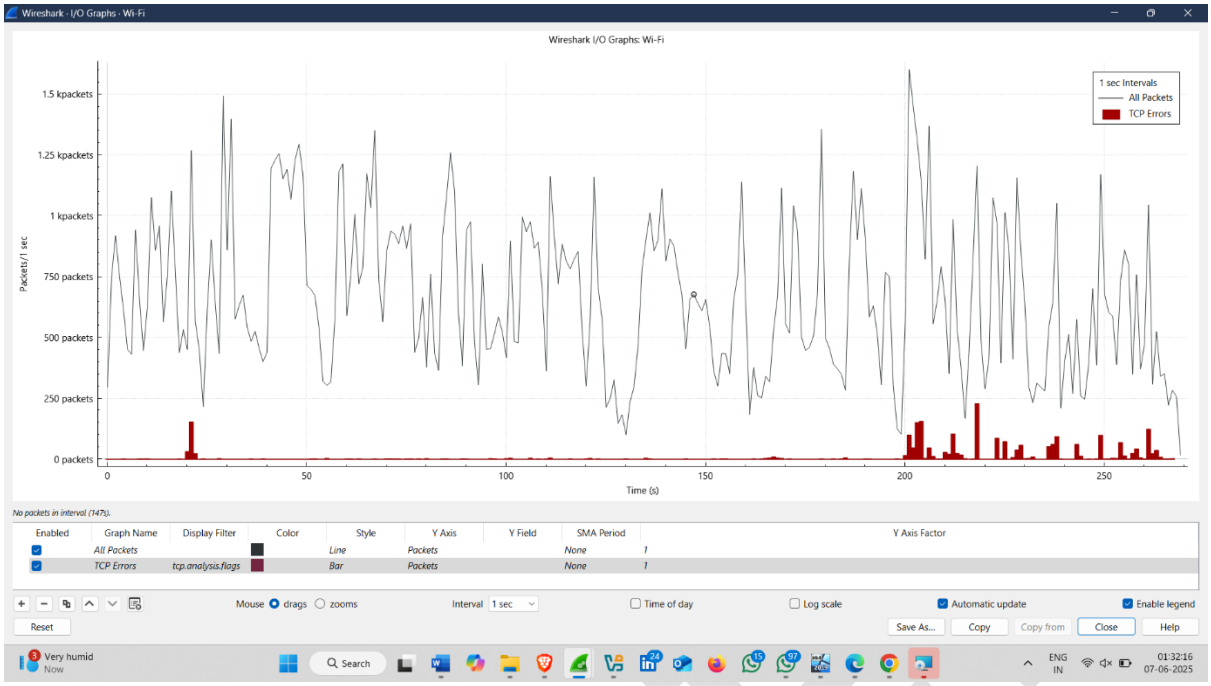
- You can also view on I/O Graphs
- Click on Statistics tab



- Click on I/O Graphs



- Here, You can See the Graph


Wireshark · I/O Graphs · Wi-Fi
Wireshark I/O Graphs: Wi-Fi
1 sec Intervals
All Packets
TCP Errors
1.5 kpackets
1.25 kpackets
1 kpackets
750 packets
500 packets
250 packets
0 packets
Packets / 1 sec
0
50
100
150
200
250
Time (s)
No packets in interval (147s).
Enabled
Graph Name
Display Filter
Color
Style
Y Axis
Y Field
SMA Period
Y Axis Factor
All Packets
Line
Packets
None
1
TCP Errors
tcp.analysis.flags
Bar
Packets
None
1
Mouse drags zooms
Interval 1 sec
Time of day
Log scale
Automatic update
Enable legend
Reset
Save As...
Copy
Copy from
Close
Help
Very humid
Now
Search
ENG
IN
01:32:16
07-06-2025

1.How To Defend Against Firewall Evasion

Common Firewall Evasion Techniques Attackers Use

1. **IP Spoofing**
    - Faking the source IP address to appear as a trusted host.
2. **Port Spoofing and Changing**
    - Using non-standard or allowed ports to sneak in traffic instead of blocked ones.
3. **Fragmentation Attacks**
    - Splitting malicious packets into fragments to bypass inspection.
4. **Protocol Anomalies and Violations**
    - Sending malformed or unexpected packets to confuse firewall inspection.
5. **Tunneling and Encapsulation**
    - Wrapping malicious traffic inside allowed protocols (e.g., HTTP tunneling, DNS tunneling, VPN).
6. **Use of Encrypted Traffic**
    - Using SSL/TLS encryption so firewall cannot inspect packet contents.
7. **Source Routing**
    - Manipulating IP header fields to control packet route, potentially bypassing firewall filters.
8. **Session Hijacking and TCP Sequence Prediction**
    - Manipulating TCP session details to bypass firewall's stateful inspection.

9. **Fragment Overlaps and Reassembly Confusion**
    - Exploiting firewall's inability to properly reassemble and inspect fragmented packets.
10. **Malicious Payload in Allowed Traffic**
    - Hiding malware inside allowed web traffic (like HTTPS or DNS).

How to Defend Against Firewall Evasion: Best Practices

1. Implement Stateful Inspection Firewalls

- Use firewalls that track the full state of active connections.
- Stateful firewalls can reject out-of-state or unexpected packets.

2. Enable Strict Packet Filtering and Validation

- Validate protocol compliance for all traffic.
- Drop malformed, unexpected, or suspicious packets.

3. Use Deep Packet Inspection (DPI)

- Inspect beyond headers, look into packet payloads.
- Detect tunneling and malicious payload hidden in allowed protocols.

4. Block IP Spoofing

- Enable **Reverse Path Forwarding (RPF)** or Unicast RPF on routers/firewalls.
- Reject packets with source IP addresses that don't match legitimate routes.

5. Disallow Source Routing

- Block IP packets with source routing options set.
- Many firewalls provide this option to reject such packets.

6. Reassemble Fragments for Inspection

- Ensure firewall can fully reassemble fragmented IP packets before filtering.
- Prevent attackers from hiding data in packet fragments.

7. Inspect and Control Encrypted Traffic

- Use SSL/TLS inspection or proxy servers to decrypt and analyze HTTPS traffic.
- Detect malware or command-and-control traffic hidden inside encryption.

8. Implement Strict Port and Protocol Controls

- Only allow necessary ports and protocols explicitly.
- Block all unused or unnecessary services.

9. Regularly Update Firewall Firmware and Rules

- Keep firewall software and firmware updated to patch evasion vulnerabilities.
- Regularly review and tune firewall rules.

10. Use Intrusion Prevention System (IPS) Alongside Firewall

- IPS can detect and block evasive attack signatures the firewall misses.
- Adds an extra layer of inspection.

11. Monitor Logs and Traffic Patterns

- Continuously monitor firewall logs for unusual or suspicious traffic.
- Detect patterns that may indicate evasion attempts.

12. Employ Network Segmentation and Zero Trust

- Limit access between network segments.

- Use zero-trust principles so no traffic is trusted by default.

13. Implement Rate Limiting and Traffic Shaping

- Limit traffic bursts and scanning attempts.
- Helps block slow or stealthy evasion attempts.

14. Use Anti-Spoofing Filters at Network Edge

- Configure routers and switches to drop spoofed packets.
- Helps reduce malicious traffic before reaching firewall.

2.How To Defend Against IDS Evasion

Common IDS Evasion Techniques Attackers Use

1. **Fragmentation**
    - Breaking malicious payloads into small IP fragments so IDS cannot reassemble or detect full attack.
2. **Packet Overlapping**
    - Sending overlapping TCP/IP packets with conflicting data to confuse IDS packet reassembly.
3. **Protocol Anomalies**
    - Using unusual or malformed packets that IDS may not parse correctly.
4. **Polymorphic Shellcode**
    - Changing malware code structure or encryption to avoid signature detection.
5. **Payload Encoding**

- Encoding payload in formats like Base64, URL encoding, or Unicode to hide attack patterns.

6. **Traffic Timing and Rate**
    - Sending packets slowly or in bursts to avoid threshold-based detection.
7. **Using Encrypted Traffic**
    - Using SSL/TLS or VPN tunnels so IDS cannot inspect payload.
8. **Evasion of Signature Matching**
    - Using variants or zero-day exploits that IDS signatures don't cover.
9. **Avoiding Known Ports**
    - Using uncommon or allowed ports to hide malicious traffic.
10. **Session Splicing**
    - Splitting attack across multiple sessions or connections to avoid detection.

How to Defend Against IDS Evasion: Key Strategies

1. Use a Combination of Detection Methods

- Combine **signature-based** IDS with **anomaly-based** IDS.
- Anomaly-based IDS can detect unusual traffic behavior even if signature evasion is attempted.

2. Enable Full Packet Reassembly

- Configure IDS to fully reassemble fragmented IP packets and TCP streams before analysis.
- Prevent evasion via fragmentation or overlapping packets.

3. Regularly Update IDS Signatures

- Frequently update IDS rules and signatures to cover latest exploits and evasion techniques.
- Use threat intelligence feeds and vendor updates.

4. Use Deep Packet Inspection (DPI)

- Inspect packet payloads in detail, not just headers.
- Detect obfuscated or encoded payloads by decoding before analysis.

5. Implement SSL/TLS Inspection

- Use SSL decryption capabilities or proxies to inspect encrypted traffic.
- Helps detect malicious content inside encrypted sessions.

6. Behavioral & Contextual Analysis

- Correlate IDS alerts with logs from other systems (firewalls, endpoints).
- Use context like user behavior, time of day, and historical data to identify evasion.

7. Deploy Multi-Layered Security

- Combine IDS with Intrusion Prevention Systems (IPS), firewalls, endpoint protection, and network segmentation.
- Defense in depth reduces chances of evasion success.

8. Tune IDS Rules to Reduce False Negatives

- Customize rules to your environment to catch evasions without overwhelming false positives.
- Remove or modify ineffective rules.

9. Monitor Traffic Patterns

- Look for unusual traffic timing, burstiness, or inconsistent flows.
- Use Network Behavior Anomaly Detection (NBAD) tools.

10. Log and Audit Everything

- Keep detailed logs of IDS alerts, network flows, and system events.
- Helps analyze suspicious activity and detect stealthy attacks.

11. Use Honeypots and Deception Techniques

- Deploy honeypots to lure attackers and detect evasive behaviors.
- Can help identify new evasion methods.

12. Train Security Personnel

- Keep SOC analysts and network admins aware of IDS evasion tactics.
- Train to recognize suspicious activity that may bypass IDS.

3.How To Defend Against IPS Evasion

Common IPS Evasion Techniques Attackers Use

1. **Packet Fragmentation**
    - Splitting attack payload into tiny fragments to avoid detection during packet reassembly.
2. **TCP Stream Manipulation**
    - Manipulating TCP packets (sequence numbers, overlaps, out-of-order segments) to confuse IPS stream reassembly.

3. **Protocol Violations and Anomalies**
    - Using malformed packets or unusual protocol behavior to evade signature or protocol checks.
4. **Encoding and Obfuscation**
    - Encoding payloads (e.g., Base64, Unicode) or using polymorphic malware to bypass signature detection.
5. **Traffic Timing Evasion**
    - Sending attacks slowly over time (low-and-slow) to avoid triggering threshold-based IPS rules.
6. **Use of Encrypted Traffic**
    - Encrypting attack payloads inside SSL/TLS so IPS can't inspect contents without decryption.
7. **Tunneling and Encapsulation**
    - Wrapping attacks inside allowed protocols (HTTP, DNS, SSH tunnels).
8. **Payload Padding and NOP Sleds**
    - Adding no-op instructions or junk bytes to disrupt signature matching.
9. **Attack Variants and Zero-day Exploits**
    - Using unknown or slightly modified exploits unknown to IPS signatures.
10. **Session Splicing**
    - Splitting attacks across multiple TCP sessions or connections.

How to Defend Against IPS Evasion: Key Strategies

1. Use Stateful and Full Stream Reassembly

- IPS should perform full TCP stream reassembly.
- Proper handling of out-of-order, overlapping, or fragmented packets to reconstruct complete payload.

2. Employ Deep Packet Inspection (DPI)

- Analyze packet payloads deeply, including decoding common encodings.
- Detect obfuscated or encoded attacks.

3. Decrypt Encrypted Traffic

- Use SSL/TLS inspection capabilities or proxies to inspect encrypted sessions.
- Identify attacks hidden inside encrypted tunnels.

4. Combine Signature-based and Anomaly-based Detection

- Signature-based detects known patterns.
- Anomaly-based detects unusual or suspicious behavior.

5. Keep IPS Signatures and Software Up-to-Date

- Regularly update to detect new exploits and evasion methods.
- Subscribe to threat intelligence feeds.

6. Use Behavioral Analysis and Correlation

- Monitor traffic behavior, session patterns, and correlate with endpoint logs.
- Detect stealthy, low-rate, or polymorphic attacks.

7. Implement Multi-layered Security

- Combine IPS with firewalls, endpoint protection, network segmentation.

- Defense in depth reduces evasion success.

8. Tune IPS Rules Carefully

- Adjust sensitivity to balance false positives and negatives.
- Customize rules for your environment.

9. Monitor and Analyze Logs Continuously

- Use Security Information and Event Management (SIEM) tools.
- Detect signs of evasion attempts.

10. Perform Regular Penetration Testing

- Test IPS resilience to evasion.
- Identify weaknesses and improve configuration.

THANK YOU