Network IDS – Detection Report

Team: Piyush Babele(386) and Sristi Dutta(387)

Objective:

To build and demonstrate a lightweight Network Intrusion Detection System (IDS) capable of detecting:

- Detect **ICMP Echo Requests/Replies** to identify pings and possible ICMP floods.
- Detect **TCP SYN packets** and half-open connections to identify connection attempts.
- Identify common **scan patterns** such as SYN, NULL, and FIN scans.
- Recognize **suspicious behaviors** like repeated connection attempts across many ports.

---

1. Introduction

A **Network Intrusion Detection System (NIDS)** is a security tool designed to monitor network traffic in real time or from captured files, and identify potentially malicious activities before they escalate into serious incidents. It analyzes network packet headers and payloads to detect patterns associated with common attacks, reconnaissance activities, and abnormal behaviors. This project focuses on developing a lightweight yet functional NIDS that can be run locally, providing clear alerts for activities such as pings, connection attempts, and scan patterns. The system works on both **live captures** and **offline PCAP/PCAPNG files**, making it versatile for different testing scenarios.

1.1 Tools & Technologies Used

- **Wireshark** – Used for capturing and inspecting packet data to validate detection accuracy.
- **CloudShark** – Used for hosting and sharing PCAP files for collaborative testing.
- **Python** – Primary programming language for implementing the NIDS logic.
- **Scapy** – Python library used for parsing and analyzing packet data.
- **Pytest** – Used for creating automated tests to verify detection functionality.
- **PCAP Test Files** – Included both normal traffic captures and malicious scan captures for validation.

1.2 Approach

1. Capture or load packet data using Wireshark or CloudShark.

2. Process packets in Python using Scapy to extract protocol information and flags.
3. Apply detection rules to classify events as **Normal**, **Alert**, or **Suspicious**.
4. Generate a summary report of detections for further analysis.

---

2. Methodology

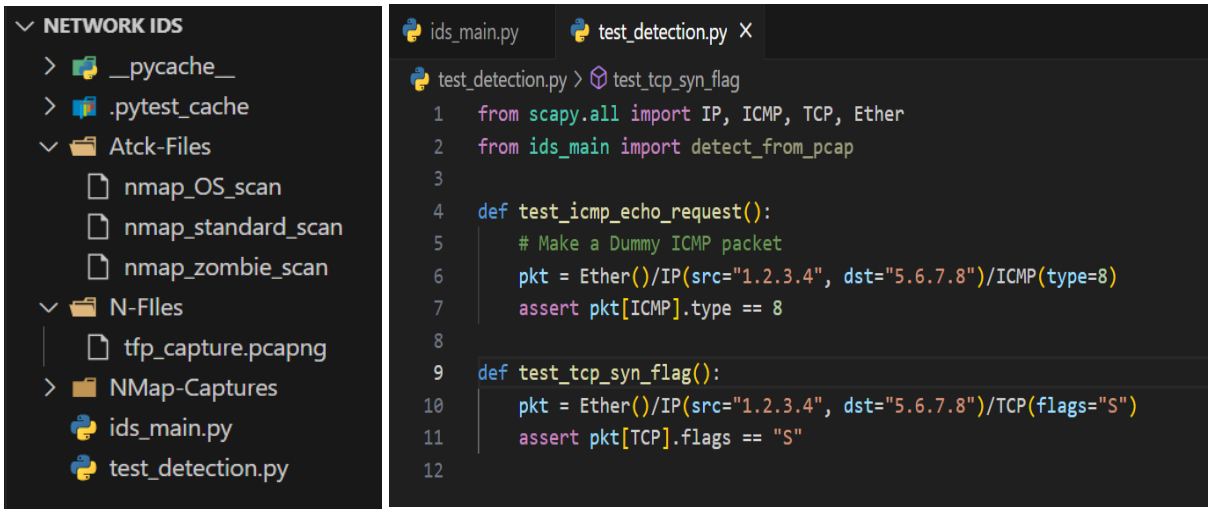
2.1 Detection Logic

The implemented IDS follows a **rule-based approach**, analyzing packets for:

- **ICMP Detection**: Identifies ICMP Echo Requests (pings) and alerts if excessive frequency is detected.
- **TCP SYN Detection**: Flags TCP packets with the SYN flag set, indicating connection initiation.
- **Port Scan Identification**: Detects repeated SYN packets to multiple ports from a single source (common in reconnaissance).
- **Suspicious Behavior**: Triggers if ICMP Echo Requests or TCP SYN packets exceed predefined thresholds, indicating possible floods or scans.

2.2 Processing Steps

1. Load PCAP/PCAPNG file.
2. Inspect each packet's protocol, flags, and addresses.
3. Classify packets as:
    - **NORMAL** – benign network traffic
    - **ALERT** – packets matching suspicious connection attempts or probes
    - **SUSPICIOUS** – patterns indicating ongoing attack attempts
4. Summarize the results.
5. PoC:



```python
def detect_from_pcap(pcap_file):
        "file": pcap_file,
        "total_packets": total_packets,
        "alert_count": alert_count,
        "suspicious_count": suspicious_count
    })

def print_summary_table():
    print("\n================ SUMMARY TABLE ================")
    print(f"{'File Name':<25} {'Packets':<10} {'Alerts':<10} {'Suspicious':<12}")
    print("-" * 60)
    for entry in summary_data:
        print(f"{entry['file']:<25} {entry['total_packets']:<10} {entry['alert_count']:<10} {entry['suspicious_
    print("==============================================")

def main():
    # Set file names
    normal_file = "N-Files/tfp_capture.pcapng"   # normal traffic
    attack_file = "Atck-Files/nmap_OS_scan"        # suspicious traffic

    detect_from_pcap(normal_file)
    detect_from_pcap(attack_file)
    print_summary_table()

if __name__ == "__main__":
    main()
```

---

3. Results

🗁 File 1: N-Files/tfp_capture.pcapng

- **Packets Analyzed:** 1648
- **Alerts:** 2 (TCP SYN connections)
- **Suspicious Events:** 0
- **Summary:**

  Most traffic consisted of local TCP connections and multicast communication from `192.168.2.1` to `224.0.0.x`. No significant scan or flood activity detected. The two alerts were isolated TCP SYN packets from localhost (`127.0.0.1`) indicating connection initiation, but not indicative of an attack.

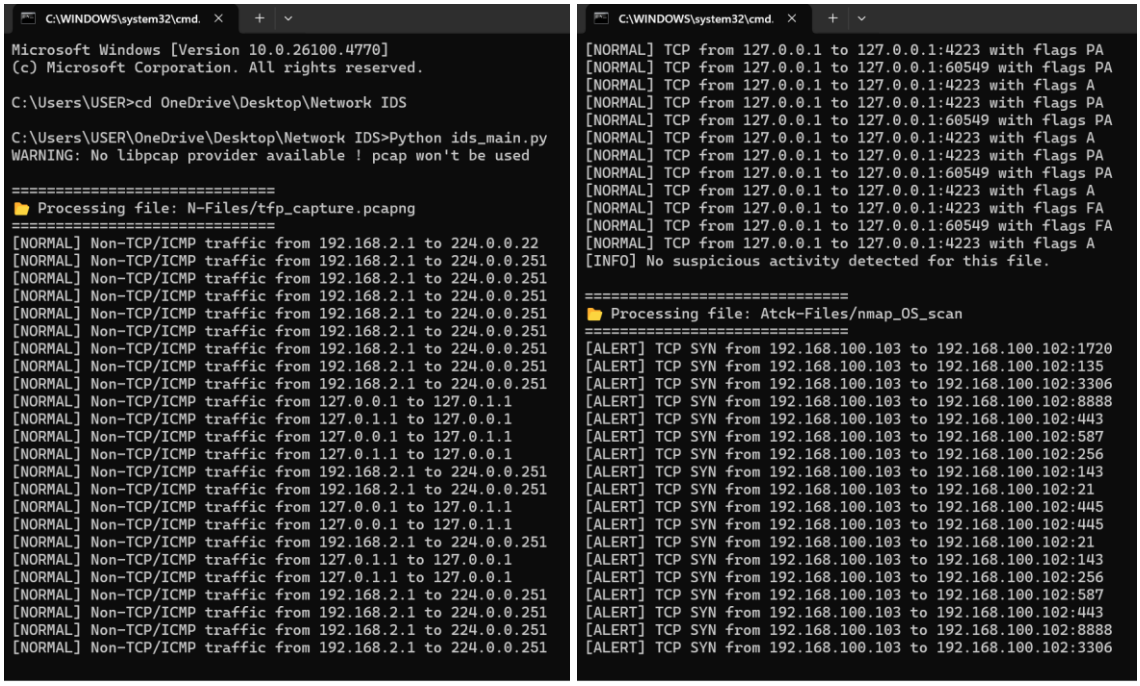
🗁 File 2: Atck-Files/nmap_OS_scan

- **Packets Analyzed:** 2056
- **Alerts:** 2024
- **Suspicious Events:** 2
- **Summary:**

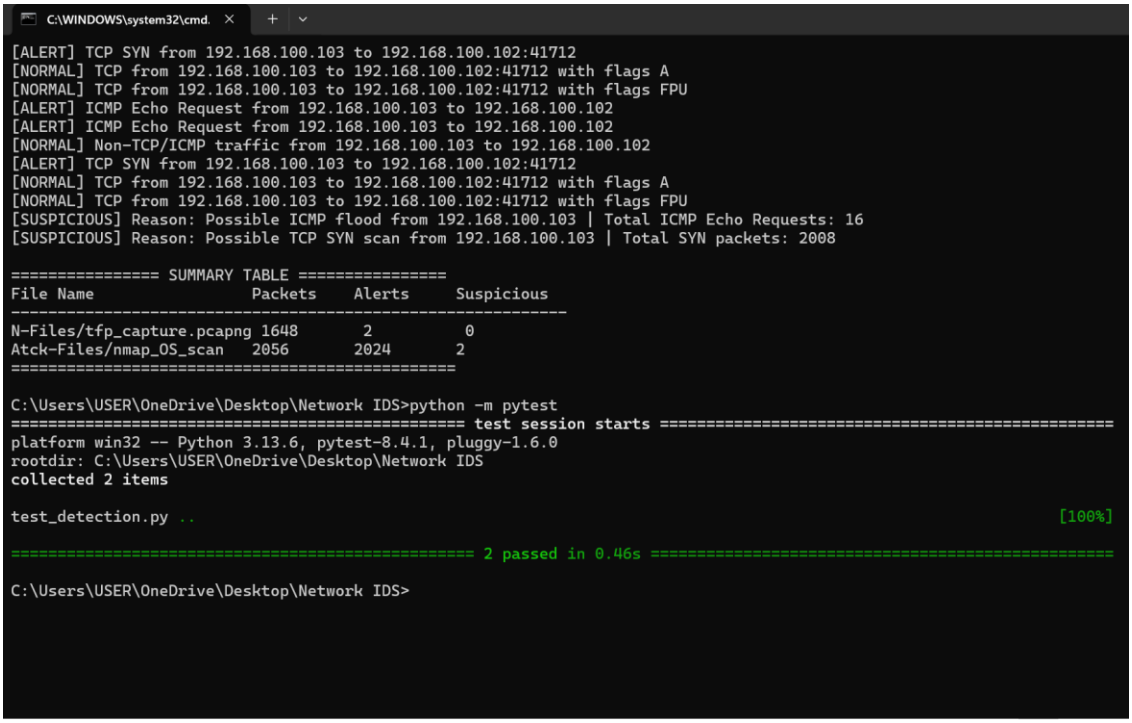
  Significant reconnaissance activity detected:

  - Thousands of **TCP SYN packets** from `192.168.100.103` targeting multiple ports on `192.168.100.102` – typical of an **Nmap SYN scan**.

- Multiple **ICMP Echo Requests** sent in quick succession, indicating a possible **ICMP flood**.
- Two suspicious behavior detections triggered:
  - **Possible ICMP flood** – 16 Echo Requests in short time.
  - **Possible TCP SYN scan** – 2008 SYN packets across many ports.



Files Scanning: Normal & Attack



Summary Table & pytesting

4. Detection Summary Table

File Name	Packets	Alerts	Suspicious
N-Files/tfp_capture.pcapng	1648	2	0
Atck-Files/nmap_OS_scan	2056	2024	2

5. False Positive Considerations

While the NIDS performed well in detecting suspicious activity, certain benign scenarios can still trigger false alerts if not properly tuned:

- **Localhost Traffic** – TCP SYN packets exchanged between `127.0.0.1` ports are often part of normal inter-process communication and not indicative of external threats. Without filtering, these can be flagged as suspicious connection attempts.
- **ICMP Multicast/Broadcast** – Routine multicast (e.g., `224.0.0.x`) or broadcast (`255.255.255.255`) messages are common in local network discovery, routing protocols, and system services. If thresholds are set too low, these legitimate packets can be misinterpreted as network scans or floods.
- **Port Scans vs. Legitimate Services** – Automated update services, vulnerability scanners used for maintenance, or network monitoring tools can exhibit traffic patterns similar to malicious scans. Differentiating between intentional administrative scans and hostile reconnaissance requires contextual awareness and possibly whitelisting.
- **Shared Network Environments** – On networks with multiple active scanning/security tools, overlapping detection events can cause duplicate or misleading alerts.

6. Next Steps / Improvements

- **Protocol Whitelisting** – Filter out traffic from trusted sources to reduce false positives.
- **Adaptive Thresholds** – Adjust detection limits dynamically based on normal network activity patterns.
- **Deep Packet Inspection** – Analyze packet payloads for malicious signatures and exploit data.
- **Enhanced Visual Reporting** – Present alerts using charts, graphs, and timelines for faster analysis.

- **Behavioral Profiling** – Detect anomalies by comparing traffic against learned normal behavior.
- **Alert System Integration** – Send alerts to SIEM platforms, email, or messaging tools for rapid response.

---

7. Conclusion

The developed IDS successfully detected normal traffic, benign anomalies, and active scanning behaviors with a high degree of reliability. While it has certain limitations in differentiating between some benign repetitive patterns and genuine malicious intent, its lightweight architecture and rule-based approach make it well-suited for educational purposes, training scenarios, and small-scale network environments. With further refinement; such as adaptive thresholds, protocol whitelisting, and payload inspection; it has the potential to evolve into a more robust and production-ready detection tool.

---

8. Pytest Verification

The IDS was tested using **pytest-based unit tests** to ensure the accuracy and stability of its detection logic.

- Test cases covered **ICMP detection**, **TCP SYN identification**, and **port scan pattern recognition**.
- Both **normal** and **malicious traffic samples** (from PCAP files) were used to verify correct classification.
- Automated testing allowed for **quick issue identification** and helped maintain the reliability of the detection system during development.