Intrusion Detection System (IDS) Report

1. Introduction

An Intrusion Detection System (IDS) is a security tool that continuously monitors network traffic to identify suspicious activities or potential attacks. It helps detect malicious behavior, such as network scanning, flooding, or unauthorized access attempts, by analyzing packet patterns, connection behavior, and traffic anomalies. This report details the logic, implementation, and limitations of a custom IDS developed using Python and Scapy.

2. IDS Logic and Implementation

The IDS monitors TCP and ICMP traffic in real-time. The key idea is to detect ICMP floods and possible scanning activities, particularly SYN scans, ACK scans, RST-based reactions for half-open connections, as well as NULL and FIN scans. The logic works as follows:

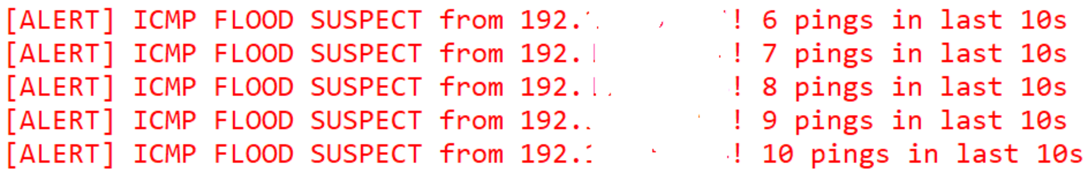
1. **SYN Scan Detection**:
    - When a packet with the SYN flag set is received, it is stored against the source IP.
    - The IDS records the destination port to detect multiple connection attempts.
    - If multiple unique ports are targeted within a short window (e.g., 10 seconds), it is flagged as a SYN scan.
2. **ACK Scan / RST Reaction**:
    - If the same source IP later sends ACK packets, the system checks whether a proper handshake was completed. If not, it is considered suspicious.
    - RST flags are also monitored. If RST responses occur frequently after SYN packets from the same IP, this indicates the host being scanned is rejecting connections, confirming scanning activity.
3. **NULL Scan Detection**:
    - A NULL scan is identified when a TCP packet is received with **no flags set**.
    - Since such packets are unusual in normal communication, repeated NULL packets from a source IP targeting multiple ports are flagged as a potential stealth scan attempt.
4. **FIN Scan Detection**:
    - A FIN scan is detected when TCP packets are received with only the **FIN flag set** and no prior connection context.
    - Similar to NULL scans, repeated FIN packets directed to multiple ports suggest an attempt to bypass standard firewall or IDS rules, triggering an alert.
5. **ICMP Flood Detection**:
    - The IDS counts the number of ICMP Echo Requests (pings) per IP within a given window (e.g., 10 seconds).

- If the count exceeds a threshold (e.g., 50), it flags a possible ICMP flood attack.

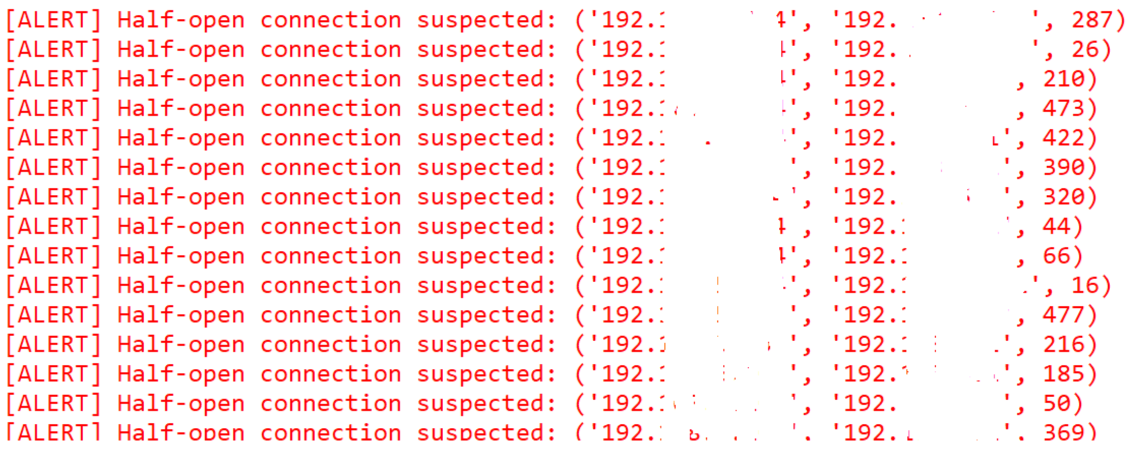
3. Alerting Mechanism

- Alerts include source IP, type of suspected activity (ICMP flood, SYN flood, SYN/NULL/FIN/RST scan), counts, and timestamps.
- Alerts are printed to the console and optionally logged to a file (simple append-based logging – alert_log.txt). The log format includes a human timestamp and the alert message.

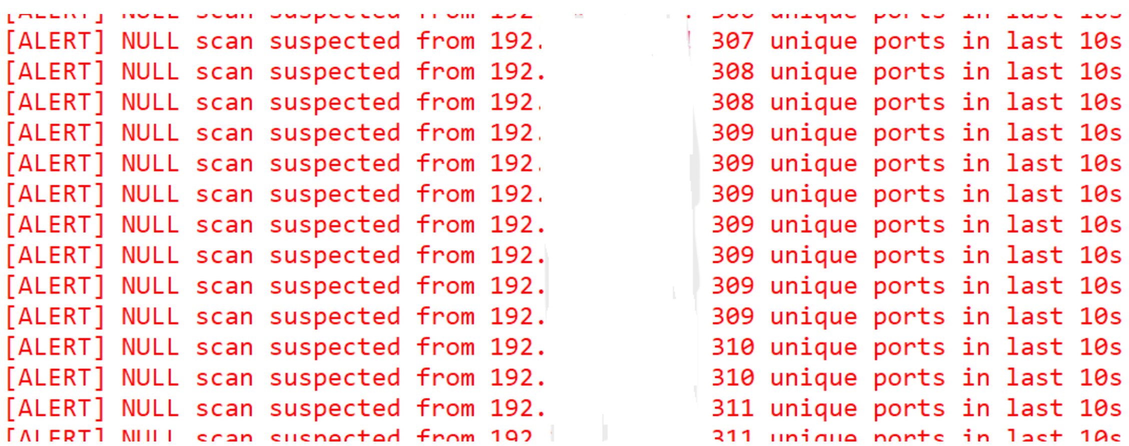
Example alert messages:



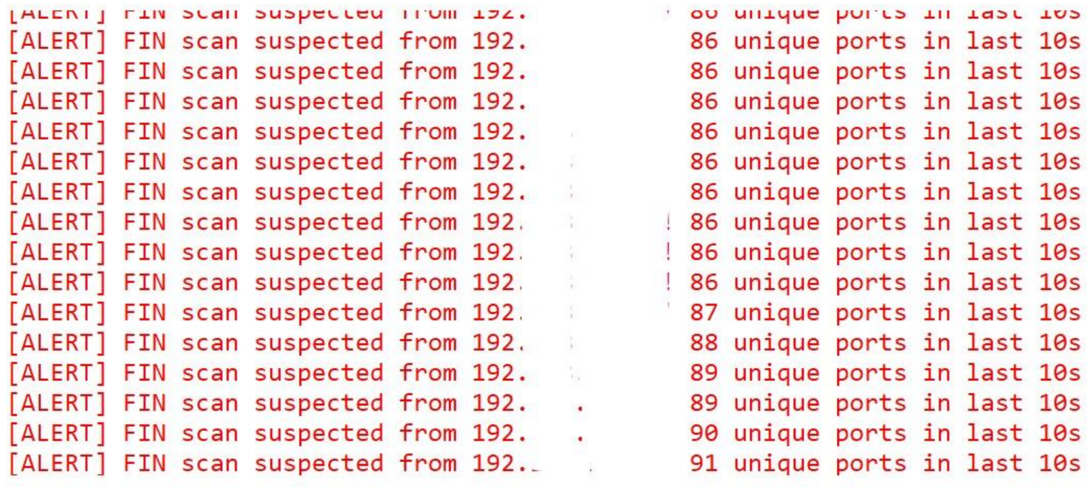
```
[ALERT] ICMP FLOOD SUSPECT from 192.          ! 6 pings in last 10s
[ALERT] ICMP FLOOD SUSPECT from 192.          ! 7 pings in last 10s
[ALERT] ICMP FLOOD SUSPECT from 192.          ! 8 pings in last 10s
[ALERT] ICMP FLOOD SUSPECT from 192.          ! 9 pings in last 10s
[ALERT] ICMP FLOOD SUSPECT from 192.          ! 10 pings in last 10s
```



```
[ALERT] Half-open connection suspected: ('192.1       4', '192.         ', 287)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 26)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 210)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 473)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 422)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 390)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 320)
[ALERT] Half-open connection suspected: ('192.1        ', '192.1        ', 44)
[ALERT] Half-open connection suspected: ('192.1        ', '192.1        ', 66)
[ALERT] Half-open connection suspected: ('192.1        ', '192.1        ', 16)
[ALERT] Half-open connection suspected: ('192.1        ', '192.1        ', 477)
[ALERT] Half-open connection suspected: ('192.1        ', '192.1        ', 216)
[ALERT] Half-open connection suspected: ('192.1        ', '192.1        ', 185)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 50)
[ALERT] Half-open connection suspected: ('192.1        ', '192.         ', 369)
```



```
[ALERT] NULL scan suspected from 192.          307 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          308 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          308 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          309 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          310 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          310 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          311 unique ports in last 10s
[ALERT] NULL scan suspected from 192.          311 unique ports in last 10s
```



```
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          86 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          87 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          88 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          89 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          89 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          90 unique ports in last 10s
[ALERT] FIN scan suspected from 192.          91 unique ports in last 10s
```

Code Flow

1. **Packet Capture**
   - The IDS continuously sniffs network packets in real time using Scapy.
   - Each incoming packet is passed to the packet_handler function for inspection.
2. **Protocol & Flag Identification**
   - The system checks whether the packet belongs to TCP or ICMP
   - For TCP: specific flags like SYN, FIN, NULL, and ACK are inspected.
   - For ICMP: echo requests (pings) are monitored.
3. **Dictionaries for Tracking**
   - icmp_count: stores timestamps of ICMP requests per source IP.
   - syn_count: stores timestamps of TCP SYN packets per source IP.
   - fin_count: stores timestamps of TCP FIN packets per source IP.
   - null_count: stores timestamps of TCP NULL (no flag) packets per source IP.

   Background Process (Clearing Old Entries)

   - A separate background thread runs continuously.
   - It removes old timestamps from each dictionary that are outside the time window (e.g., 10 seconds).
   - This ensures that only recent activity is analyzed, preventing false positives.
4. **Threshold Checking (Detection Logic)**
   - If the number of entries in a dictionary exceeds a predefined threshold within the time window, it indicates suspicious activity.
   - Example:
     - Too many SYN packets from one IP → **SYN scan**.
     - Too many FIN packets → **FIN scan**.
     - Too many NULL packets → **NULL scan**.
     - Too many ICMP requests → **Ping flood**.

5. **Alert Generation**
    - When a threshold is crossed, the IDS prints an alert message and logs it in alert_log.txt specifying:
        - The source IP.
        - The type of suspicious activity detected.

4. Limitations

While effective for basic threat detection, the current IDS has several limitations:

- It primarily monitors common TCP scan types and ICMP packets; other attack types, such as UDP floods or IP fragmentation attacks, are not detected.
- Threshold-based detection may result in false positives in high-traffic or busy networks.
- Alerts are generated in real-time only; there is no persistent logging system for long-term analysis or auditing.
- Attackers can bypass detection using slow or stealthy scanning techniques (low-and-slow attacks).

5. Improvements

The IDS can be enhanced through the following measures:

- Implement persistent logging to files or databases for audit, historical analysis, and forensic purposes.
- Adopt statistical anomaly detection methods instead of relying solely on fixed thresholds to reduce false positives.
- Expand monitoring to additional protocols and attack types, such as UDP scans, DNS attacks, and other reconnaissance methods.
- Integrate with intrusion prevention systems (IPS) to automatically block or mitigate malicious IPs.
- Develop a graphical user interface (GUI) dashboard for real-time visualization and easier monitoring of network activity.

6. Conclusion

This custom Python-based IDS demonstrates the logic of detecting TCP SCAN types –SYN, ACK, RST, NULL, and FIN, and ICMP floods. While it provides a foundational approach to intrusion detection, further enhancements in scalability, accuracy, and prevention integration can make it a more robust solution for real-world network security monitoring.