**Overview of the Network Scanning & Attack Tool – TCP SCAT**

This Python-based **Network Scanning & Attack Tool** is designed to simulate common network scanning techniques and network-based attacks for educational purposes, particularly in classroom environments. The tool provides students with hands-on experience in understanding how network scanning works, how various attacks are conducted, and how they manifest when monitored using network analysis tools such as **Wireshark**.

**Key Features**:

* **Nmap-Style Scanning Techniques**: Simulate multiple port scanning techniques such as SYN scans, Connect scans, and FIN scans.
* **Network Attacks**: Simulate TCP-based attacks like SYN Flood, RST Injection, Slowloris, etc.
* **IP Spoofing**: Enable IP spoofing for applicable attacks.
* **Custom Verbose and Stealth Modes**: Control verbosity for detailed feedback or use stealth scans for more covert scanning.
* **Nmap Command Simulation**: The tool prints the **Nmap command equivalent** of the selected scan for educational purposes, helping students understand the correlation between the scan and Nmap commands.
* **Wireshark-Compatible Monitoring**: All scans and attacks are designed to manifest in network analysis tools like Wireshark for real-time observation.

**Scan Modes**

**1. Ping Scan (ICMP)**

* **Nmap Equivalent**: nmap -sn <target-ip>
* **Function**: Sends an ICMP Echo Request to check if the target is up.
* **Expected Wireshark Output**:
  + ICMP Echo Request packets sent from the source machine and ICMP Echo Reply packets from the target (if the host is up).
  + If the target is down, no ICMP replies will be seen.
* **Required Input**:
  + **Target IP**: This is necessary because the ping scan needs to know which machine to check for availability.

**2. TCP SYN Scan (Stealth Scan)**

* **Nmap Equivalent**: nmap -sS <target-ip> -p <port-range>
* **Function**: Sends SYN packets to the target ports, attempting to initiate a half-open TCP connection without completing the TCP handshake.
* **Expected Wireshark Output**:
  + SYN packets sent to the target.
  + If the port is open, the target responds with a SYN-ACK packet.
  + If the port is closed, the target responds with an RST packet.
  + The scan doesn't complete the handshake, making it stealthier and less detectable by logging systems.
* **Required Input**:
  + **Target IP**: Specifies the machine to scan.
  + **Port Range**: Allows the user to select specific ports or scan all ports. This is important because not all ports may be relevant, and scanning a wide range of ports can take more time.

**3. TCP Connect Scan**

* **Nmap Equivalent**: nmap -sT <target-ip> -p <port-range>
* **Function**: Completes the full TCP handshake by sending SYN, receiving SYN-ACK (if open), and completing the handshake with an ACK.
* **Expected Wireshark Output**:
  + Full TCP handshakes (SYN, SYN-ACK, and ACK) visible for open ports.
  + If the port is closed, expect SYN followed by RST packets.
* **Required Input**:
  + **Target IP**: Specifies the machine to scan.
  + **Port Range**: The user must define the ports to be scanned.

**4. TCP FIN Scan**

* **Nmap Equivalent**: nmap -sF <target-ip> -p <port-range>
* **Function**: Sends FIN packets to target ports. Open ports generally do not respond, while closed ports send RST packets.
* **Expected Wireshark Output**:
  + FIN packets sent from the scanner to the target.
  + Closed ports respond with RST packets, while no response is expected from open/filtered ports.
* **Required Input**:
  + **Target IP**: Specifies the machine to scan.
  + **Port Range**: The port range is critical to determine which ports are being targeted for the scan.

**5. TCP Xmas Scan**

* **Nmap Equivalent**: nmap -sX <target-ip> -p <port-range>
* **Function**: Sends packets with FIN, URG, and PSH flags set. Closed ports should respond with RST, while open or filtered ports generally do not respond.
* **Expected Wireshark Output**:
  + Xmas packets sent from the source machine.
  + Closed ports return RST packets, while no response indicates an open or filtered port.
* **Required Input**:
  + **Target IP**: Specifies the machine to scan.
  + **Port Range**: Defines which ports are being scanned.

**6. TCP Null Scan**

* **Nmap Equivalent**: nmap -sN <target-ip> -p <port-range>
* **Function**: Sends TCP packets with no flags set. Closed ports respond with RST, while open or filtered ports typically do not respond.
* **Expected Wireshark Output**:
  + Null packets (TCP packets with no flags) sent to the target.
  + RST responses from closed ports, no response from open/filtered ports.
* **Required Input**:
  + **Target IP**: Specifies the machine to scan.
  + **Port Range**: The port range allows users to scan the required ports.

**Attack Modes**

**7. TCP SYN Flood Attack**

* **Function**: Overwhelms a target by sending a large number of SYN packets to exhaust the target's resources and prevent legitimate connections.
* **Expected Wireshark Output**:
  + A rapid and large volume of SYN packets sent to the target port (typically port 80 for HTTP services).
  + The target may respond with SYN-ACK packets but won't receive the final ACK to complete the handshake, potentially causing SYN queue exhaustion on the target.
* **Required Input**:
  + **Target IP**: Specifies the machine to attack.
  + **Number of SYN packets**: Determines how many SYN packets will be sent to flood the target. This affects the intensity of the attack.
  + **IP Spoofing Option**: Determines whether to enable spoofing, which hides the attacker’s IP by sending the packets with a false source IP.

**8. TCP RST Injection Attack**

* **Function**: Sends forged TCP RST (Reset) packets to terminate an existing connection between the target and another host.
* **Expected Wireshark Output**:
  + RST packets appearing in the middle of an established TCP conversation, disrupting or terminating the session.
* **Required Input**:
  + **Target IP**: Specifies the machine to attack.
  + **Target Port**: The port where the active connection exists.
  + **IP Spoofing Option**: The attacker can spoof the source IP to hide the true origin of the attack.

**9. TCP ACK Flood Attack**

* **Function**: Overwhelms the target by sending large volumes of TCP ACK packets, which can flood the target’s resources or confuse stateful packet inspection systems.
* **Expected Wireshark Output**:
  + A large number of ACK packets sent from the source machine to the target, potentially without an initial SYN or completed TCP handshake.
* **Required Input**:
  + **Target IP**: Specifies the machine to attack.
  + **Number of ACK packets**: Determines how many ACK packets will be sent to flood the target.
  + **IP Spoofing Option**: The attacker can use IP spoofing to hide the true source of the attack.

**10. Slowloris Attack**

* **Function**: Attempts to exhaust the target web server by keeping multiple HTTP connections open and sending incomplete headers, preventing legitimate users from connecting.
* **Expected Wireshark Output**:
  + Multiple partially opened TCP connections to the target.
  + Incomplete HTTP headers will be visible in Wireshark, with long intervals between header fragments.
* **Required Input**:
  + **Target IP**: Specifies the web server to attack.
  + **Target Port**: The target port, typically port 80 (HTTP).
  + **Number of Connections**: Specifies how many concurrent connections to open with the target.
  + **Sleep Time Between Headers**: Determines how frequently the headers are sent, affecting the intensity of the attack.

**11. DNS Amplification Attack**

* **Function**: Sends spoofed DNS requests to a DNS server with the target’s IP, causing the DNS server to send large responses to the target in a reflection-style attack.
* **Expected Wireshark Output**:
  + DNS queries sent from the attacker's IP (spoofed as the target).
  + Large DNS responses directed at the target from the DNS server, potentially overwhelming the target’s resources.
* **Required Input**:
  + **Target IP**: Specifies the machine to attack.
  + **DNS Server IP**: The DNS server that will be used to send responses to the target.
  + **Number of DNS Requests**: The number of DNS requests to send, determining the volume of the amplification attack.

**12. ARP Spoofing Attack**

* **Function**: Sends forged ARP packets to associate the attacker's MAC address with the IP of a target, enabling interception or modification of traffic.
* **Expected Wireshark Output**:
  + ARP requests and ARP replies appearing from the attacker with a false MAC-IP association.
  + Man-in-the-middle (MITM) behavior can be observed, where traffic to the target is rerouted through the attacker.
* **Required Input**:
  + **Target IP**: Specifies the target machine whose ARP entry will be spoofed.
  + **Spoof IP**: The IP address that will be associated with the attacker’s MAC address.
  + **Network Interface**: The network interface (e.g., eth0) on which the attack will be conducted. This is important as ARP operates at the local network level.

**Monitoring in Wireshark**

By monitoring network traffic with **Wireshark**, students can observe how each of these scans and attacks manifests in a real-time environment:

* **Packet Flags**: Different flags (e.g., SYN, FIN, RST) can be seen in the TCP headers, depending on the type of scan or attack.
* **Connection Behavior**: Scans like the SYN scan or TCP Connect scan show how a TCP handshake works and what happens when ports are open, closed, or filtered.
* **Attack Patterns**: Flooding attacks will generate a large number of packets in a short period, making the target system's resources work harder, which may show high packet rates in Wireshark.
* **Spoofing**: In ARP spoofing, Wireshark will reveal MAC address mismatches as the attacker spoofs the target's MAC address.

**Conclusion**

This tool provides a safe, educational environment for students to explore and experiment with network scanning and attacks. Each scan and attack mirrors real-world techniques used in penetration testing and cybersecurity research. By leveraging Wireshark, students gain valuable hands-on experience in monitoring and analyzing network traffic, enhancing their understanding of both offensive and defensive networking concepts.