Course	COMP 7005
Program	BScACS, Network Security
Term	September 2024

- This is an individual assignment.

Objective

- This assignment introduces students to network packet crafting using Scapy.
- You will build a basic port scanner that mimics the behaviour of hping3.
- This exercise develops skills in working with TCP/IP protocols, interpreting packet responses, and handling command-line arguments effectively.

Learning Outcomes

- Upon completing this assignment, students will be able to:
    - Understand the role and mechanics of port scanning.
    - Learn to create and send custom TCP packets using Scapy.
    - Implement a SYN scan to determine the status of ports.
    - Gain experience handling command-line input and implementing optional features.

Assignment Details

- Your task is to write a Python port scanner using Scapy.
- This tool will send TCP SYN packets to specified ports on a target machine to determine their status (open, closed, or filtered).
- You will create command-line arguments to specify the target, the port range, and optional delays between scans.

How Port Scanning Works

- Port scanning sends packets to specific ports on a host and waits for a response to determine if the ports are open, closed, or filtered.
    - Open Port: The target sends back a SYN-ACK packet.
    - Closed Port: The target responds with an RST packet.
    - Filtered Port: No response or an ICMP unreachable message is received.

- This scanning technique is known as a TCP SYN scan and is commonly used because it provides basic information about the target's services without fully opening a TCP connection.

Command-Line Arguments

- Target IP Address: This is required to specify which machine to scan.
- Port Range:
    - The scanner should check all ports from 1 to 65535 if no range is provided.
    - If only the start port is provided, scan from start to 65535.
    - If only the end port is provided, scan from 1 to the end port.
    - If both start and end are provided, scan from start to end.
- Optional Delay: Allow for a delay (in milliseconds) between each scan.

```
python3 port_scanner.py 192.168.1.10 --start 20 --end 80
--delay 100
```

Constraints

- Your program must run on a UNIX-based operating system (e.g., Linux, FreeBSD, macOS).
- Scapy must be used to craft the TCP packets and send them to the target.
- Handle network errors gracefully, including unreachable hosts and invalid IP addresses.
- Ensure the command-line interface provides meaningful feedback for incorrect inputs.

Resources

- Any examples that I provided or that you find.

Submission

- Ensure your submission meets all the guidelines, including formatting, file type, and submission.
- Follow the AI usage guidelines.
- Be aware of the late submission policy to avoid losing marks.
- ***Note: Please strictly adhere to the submission requirements to ensure you don't lose any marks.***

Evaluation

Topic	**Value**

Port Scanning	60%
Testing	20%
Documentation	20%
Total	100%

Hints

- Start with a small port range on a known local machine to validate your logic.
- For external scans, use a controlled environment or lab setup.
- Scapy requires elevated privileges to send raw packets.
- Use sudo if necessary to run your program.
- Convert milliseconds to seconds to manage the delay effectively between port scans.
- Look for specific TCP flags in the response to determine the status of each port.