# Port Scanner

Due Date: April 18th, 2025 11:59pm

## Introduction

Port scanning is an essential technique used in network security to identify open ports on a computer or network device. These open ports are potential points of entry for attackers and must be monitored closely. A port scanner is a tool that helps system administrators and security professionals check which ports on a network are open or closed, and by extension, which services are running on those ports. This project aims to develop a simple port scanner using Python, focusing on the ability to scan multiple IP addresses and ports concurrently while being able to identify active services and assess potential vulnerabilities.

## Objectives

- Develop a port scanning tool in Python capable of scanning multiple IP addresses for open ports.

- Utilize socket programming to interact with network ports, sending SYN packets and receiving responses to determine the state of ports.

- Support different scanning techniques such as TCP Connect scan and SYN scan (half-open scan) to assess the state of ports.
  - TCP Connect Scan: The simplest form of port scanning involves attempting to establish a full TCP connection to a given port. If the connection is successful, the port is open. Otherwise, it is closed or filtered.
  - SYN Scan: This scan sends a SYN packet to a target port and analyzes the response. If the target responds with a SYN-ACK, the port is open. If a RST (reset) packet is received, the port is closed. This method is faster than a full TCP connection scan.

- Parallel scanning: Use threading or asynchronous methods to scan multiple ports or hosts simultaneously, improving efficiency.

- Provide detailed output that identifies the port status (open, closed, filtered) and any services associated with the open ports.

- Parallel Scanning: To improve the speed of the tool, threading or asynchronous I/O will be implemented to scan multiple ports concurrently.

- User Input and Output: Provide a command-line interface where users can specify target IPs and ports. The output will list the open ports and the associated services (if detectable, e.g., HTTP on port 80).

## Flags and Options:

-h, --help: Display the help message and usage instructions.

Description: Displays information on how to use the port scanner and explains the various flags and options available.

-t, --target: Specify the target IP or range of IPs to scan.

Description: Allows the user to input one or more target IP addresses (e.g., 192.168.1.1 or 192.168.1.1,192.168.1.2).

Example: -t 192.168.1.1 or -t 192.168.1.1-192.168.1.10

-p, --ports: Specify the range of ports to scan.
Description: The user can input specific ports (e.g., 80,443) or a range of ports (e.g., 1-1024).
Example: -p 80,443 or -p 1-1024

-s, --scan-type: Choose the type of scan to perform (TCP Connect or SYN Scan).
Description: This flag allows the user to specify which scanning technique to use.

By default, a TCP Connect scan can be used, but a SYN Scan can also be selected for stealthier scanning.

Example: -s syn or -s connect

-v, --verbose: Enable verbose output.

Description: When this flag is used, the scanner will provide more detailed information about the scan process, including progress, individual port status, and any errors encountered.

Example: -v

-o, --output: Specify the output format or file.

Description: Allows the user to specify the format for the scan results (e.g., plain text, JSON, CSV) or direct the output to a file (e.g., scan_results.txt).

Example: -o scan_results.txt or -o json

-r, --retry: Number of retries on failed connection attempts.

Description: This flag specifies the number of times to retry scanning a port if the initial connection attempt fails. Useful for networks with intermittent connectivity or firewalls.

Example: -r 3

-t, --timeout: Set a timeout for each scan attempt.

Description: Specifies the maximum time in seconds for a connection attempt to a port. If the connection takes longer than the specified timeout, the port is considered unresponsive.

Example: -t 2 (timeout of 2 seconds for each port)

-n, --no-resolve: Disable reverse DNS resolution.

Description: By default, the scanner may resolve IP addresses to domain names. This flag disables that behavior, making the scan faster, especially when scanning large networks or unfamiliar IP ranges.

Example: -n
-u, --user-agent: Specify a custom user-agent string for HTTP-based scans.

Description: When performing HTTP-related port scans (e.g., on port 80), a custom user-agent string can be provided. This can help disguise the scanner as a regular web browser or other application.

Example: -u "Mozilla/5.0"

--exclude: Exclude specific IPs or ports from the scan.

Description: Allows the user to exclude certain IP addresses or ports from the scan. Useful for excluding known safe devices or ports that should not be scanned.

Example: --exclude 192.168.1.1,443

--banner: Enable service banner grabbing.

Description: When enabled, this option will attempt to grab the service banner of open ports (e.g., HTTP version, SSH version) to help identify the service running on the open port.

Example: —banner

## Example Usage:

```
python3 port_scanner.py -t 192.168.1.1 -p 1-1024 -s syn -v -o
scan_results.txt --banner --retry 3 --timeout 2
```

Explanation:

- -t 192.168.1.1: Scan the IP address 192.168.1.1.
- -p 1-1024: Scan ports from 1 to 1024.
- -s syn: Use SYN scan for stealthy scanning.
- -v: Display verbose output.
- -o scan_results.txt: Output results to scan_results.txt.
- --banner: Grab service banners from open ports.
- --retry 3: Retry scanning a port 3 times if the initial connection fails.
- --timeout 2: Set a timeout of 2 seconds per port scan attempt.

## Expected Outcomes

To demonstrate this project you will use your port scanner to scan cse3320.org. While port scanning cse3320.org you will also capture the network traffic to and from the scanning pc using wireshark. You will deliver a report of the scan results a pcap file from wireshark showing the scanning in action, and a screen recording of your scanning of cse3320.org.

Your code, report, screen recording and pcap file will be delivered in your repository at: https://github.com/CSE4380-Spring-2025-Honors