Port Scanner using Python

Project Report Prepared by: Tanmoy Dan

Date: September 12, 2025

Project: Port Scanner using Python

Abstract

This document describes a small port scanner implemented using Python, The objective of this project is to develop a Python-based port scanner that identifies open ports on a target machine. This project helps to understand network scanning, socket programming, and multithreading for efficient scanning.

Design & How it works

1.Target Input: The user enters the target IP address and the range of ports to scan.

2. Socket Connection: The script attempts to establish a connection to each port.

3. Service Identification: If a port is open, the scanner tries to identify the running service.

4. Banner Grabbing: The scanner retrieves any available banner information from the open port.

5. Multithreading for Speed: The scanning process is optimized using Python's ThreadPoolExecutor, allowing concurrent scanning of multiple ports for faster results.

6. Formatted Output: The results are displayed in a structured table, highlighting open ports, detected services, and banners.

Key Concepts Covered:

- Socket Programming: Using Python's socket module to establish connections.
- Banner Grabbing: Extracting information about the service running on open ports.
- Multithreading: Speeding up the scan by handling multiple ports simultaneously.
- Command-line Interaction: Accepting user inputs for target IP and port range.
- Formatted Output Display: Structuring scan results for readability.

Step-by-Step Implementation:

1. Import the required modules (socket, concurrent.futures, sys).

2. Define a function to scan a given port using a socket connection.

3. Implement a get_banner() function to retrieve the banner from open ports.

4. Use ThreadPoolExecutor to scan multiple ports concurrently.

5. Capture and store results, including open ports, services, and banners.

6. Format and display the results in a structured manner.

7. Provide user input prompts for specifying the target IP and port range.

8. Handle errors and exceptions for robust execution.

```
import socket
import concurrent.futures
import sys

RED = "\033[91m"
GREEN = "\033[92m"
RESET = "\033[0m"

def format_port_results(results):
    formatted_results = "Port Scan Results:\n"
    formatted_results += "{:<8} {:<15} {:<10}\n".format("Port", "Service", "Status")
    formatted_results += '-' * 85 + "\n"
    for port, service, banner, status in results:
        if status:
            formatted_results += f"{RED}{port:<8} {service:<15} {'Open':<10}{RESET}\n"
            if banner:
                banner_lines = banner.split('\n')
                for line in banner_lines:
                    formatted_results += f"{GREEN}{'':<8}{line}{RESET}\n"
    return formatted_results

def get_banner(sock):
    try:
        sock.settimeout(1)
        banner = sock.recv(1024).decode().strip()
```

```
        return banner
    except:
        return ""

def scan_port(target_ip, port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((target_ip, port))
        if result == 0:
            try:
                service = socket.getservbyport(port, 'tcp')
            except:
                service = 'Unknown'
            banner = get_banner(sock)
            return port, service, banner, True
        else:
            return port, "", "", False
    except:
        return port, "", "", False
    finally:
        sock.close()

def port_scan(target_host, start_port, end_port):
    target_ip = socket.gethostbyname(target_host)
    print(f"Starting scan on host: {target_ip}")
```

```
    results = []
    with concurrent.futures.ThreadPoolExecutor(max_workers=400) as executor:
        futures = {executor.submit(scan_port, target_ip, port): port for port in range(start_port, end_port + 1)}
        total_ports = end_port - start_port + 1
        for i, future in enumerate(concurrent.futures.as_completed(futures), start=1):
            port, service, banner, status = future.result()
            results.append((port, service, banner, status))
            sys.stdout.write(f"\rProggress: {i}/total_ports ports scanned")
            sys.stdout.flush()

    sys.stdout.write("\n")
    print(format_port_results(results))

if __name__ == '__main__':
    target_host = input("Enter your target ip: ")
    start_port = int(input("Enter the start port: "))
    end_port = int(input("Enter end port: "))

    port_scan(target_host, start_port, end_port)
```

Screenshot of SCAN Output

c:\Users\Personal\Desktop\project\Port Scanner>python3 port_scanner.py

Enter your target ip: 192.168.29.1

Enter the start port: 1

Enter end port: 1000

Starting scan on host: 192.168.29.1

Proggress: 1000/total_ports ports scanned

Port Scan Results:

Port Service Status

←[91m80 http Open ←[0m

←[91m443 https Open ←[0m

c:\Users\Personal\Desktop\project\Port Scanner>python3 port_scanner.py

Enter your target ip: 192.168.29.1

Enter the start port: 1

Enter end port: 10000

Starting scan on host: 192.168.29.1

Proggress: 10000/total_ports ports scanned

Port Scan Results:

Port Service Status

←[91m80 http Open ←[0m

←[91m443 https Open ←[0m

←[91m1900 ssdp Open ←[0m

←[91m2872 Unknown Open ←[0m

←[91m5068 Unknown Open ←[0m

←[91m7443 Unknown Open ←[0m

←[91m8080 Unknown Open ←[0m

←[91m8443 Unknown Open ←[0m

c:\Users\Personal\Desktop\project\Port Scanner>