Automated Port Scanner Using Python for Network Reconnaissance

Author: Ansuman Behera

Affiliation: InLighnX Global Pvt. Ltd.

Intern ID: ITID0510

Abstract

This project involves developing a simple automated port scanner using Python to perform active network reconnaissance. The tool scans a target IP address or domain to identify open ports and their associated services. This mimics the first phase of penetration testing — information gathering — which is essential in any offensive cybersecurity engagement.

Port Scanner

A **port scanner** is a cybersecurity tool used to identify open ports and services running on a target machine. In the context of **offensive security**, port scanning is a critical step during the **reconnaissance phase** of penetration testing, as it helps attackers or ethical hackers map the network and detect potential entry points.

This project involves creating a **Python-based automated port scanner** that connects to specified ports on a target IP address or domain to check their status (open/closed). The scanner also attempts to retrieve service banners from open ports, providing further insight into the system's exposed services.

By automating this process using Python and multithreading, the tool allows fast and efficient scanning of port ranges, mimicking real-world penetration testing techniques.

Objectives:

- Scan target IP for open ports
- Identify services from banners
- Log or display results clearly
- Understand ethical usage in penetration testing

Tools Used:

- Python 3
- Libraries: socket, argparse, concurrent.futures

Methodology:

- 1. Target IP is resolved.
- 2. Ports scanned in parallel threads.
- 3. Open ports are logged with banners.
- 4. Results displayed in console.

Source Code

```
port_scanner.py > \( \Omega \) run_scanner
     import socket
     import argparse
     from concurrent.futures import ThreadPoolExecutor
     def scan_port(ip, port):
             sock = socket.socket()
             sock.settimeout(1)
             sock.connect((ip, port))
                 banner = sock.recv(1024).decode().strip()
                 banner = "No banner"
             return (port, "OPEN", banner)
     def run_scanner(ip, ports):
         open ports = []
         with ThreadPoolExecutor(max_workers=100) as executor:
           results = executor.map(lambda p: scan_port(ip, p), ports)
              for result in results:
                 if result:
                     open_ports.append(result)
         return open_ports
      if __name__ == "__main__":
          parser = argparse.ArgumentParser(description="Simple Python Port Scanner")
         parser.add_argument("target", help="Target IP or domain to scan")
         parser.add_argument("--start", type=int, default=1, help="Start port")
         parser.add_argument("--end", type=int, default=1024, help="End port")
         args = parser.parse_args()
         print(f"Scanning {args.target} from port {args.start} to {args.end}...\n")
         ports = list(range(args.start, args.end + 1))
         results = run_scanner(args.target, ports)
          for port, status, banner in results:
              print(f"[+] Port {port} is {status} - Banner: {banner}")
```

Sample Run

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\InlignTtech_Internship\Projects> python port_scanner.py scanner.nmap.org --start 1 --end 100
```

Output

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\InlignTtech_Internship\Projects> python port_scanner.py scanner.nmap.org --start 1 --end 100 Scanning scanne.nmap.org from port 1 to 100...

[+] Port 22 is OPEN - Banner: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13
[+] Port 80 is OPEN - Banner: No banner
PS D:\InlignTtech_Internship\Projects>
```

Extensions (Optional):

- Save results to a text/CSV file
- Add UI with Tkinter or web version with Flask
- Scan for vulnerabilities based on banner info

Conclusion

The Python-based port scanner developed in this project demonstrates the core concepts of network reconnaissance in offensive cybersecurity. By scanning and identifying open ports and service banners on a target system, the tool aids in detecting potential vulnerabilities that could be exploited in further stages of penetration testing.

This project not only reinforces the practical use of Python in cybersecurity but also emphasizes the importance of ethical hacking practices. The port scanner provides a solid foundation for understanding how attackers gather information and how defenders can proactively secure exposed services.

References

- Python Documentation Sockets, Threading
 https://docs.python.org/3/library/socket.html
 https://docs.python.org/3/library/concurrent.futures.html
- Nmap Network Scanning Gordon "Fyodor" Lyon https://nmap.org/book/
- OWASP Reconnaissance Guide
 https://owasp.org/www-community/Information_Gathering
- Port Scanning Techniques SANS Institute Whitepapers
 https://www.sans.org/white-papers/
- Learn Python Full Course for Beginners [freeCodeCamp]
 https://www.youtube.com/watch?v=rfscVS0vtbw