NAME: SAMUEL BIJU

REG NO: 2462141

Assignment

Python Socket Port Scanner

Index

***Problem statement**:*

Port scanning is a foundational activity in network security that identifies open TCP ports on a host. The objective of this assignment is to design a Python-based tool that, when executed in Kali Linux, scans a specified target (IP address or hostname) over a defined port range and reports which ports are open. The solution should be simple, reliable, and suitable for instructional purposes, demonstrating core socket programming concepts, input handling, and basic exception management.

Methodology:

The solution employs Python socket API to attempt TCP connections to ports within a specified range. A port is considered open if a TCP connect attempt succeeds. The scanner proceeds sequentially, applying a short timeout per port to balance responsiveness and completeness, and it incorporates exception handling for common failure modes such as unresolved hostnames or interrupted execution.

Approach*:*

The tool iterates through ports using socket.connect -ex() to probe each port. A return code of 0 denotes an open port.

Assumptions:

The target is reachable; the scan is authorized; and the environment has Python 3 available on Kali Linux.

Design choices:

A modest timeout prevents stalls. Sequential scanning keeps the implementation clear for learning purposes, trading speed for simplicity.

Program code:

```
import socket
import sys

def scan-ports(target, start-port=1, end-port=1024, timeout=0.5):
    print(f"\nScanning {} from port {} to {}...\n".format(target, start-port, end-port))
    try:
        for port in range(start-port, end-port + 1):
            s = socket.socket(socket.AF-INET, socket.SOCK-STREAM)
            s.settimeout(timeout)
            result = s.connect-ex((target, port))
            if result == 0:
                print(f"Port {} is OPEN".format(port))
            s.close()
    except KeyboardInterrupt:
        print("\nScan interrupted by user.")
        sys.exit(1)
    except socket.gaierror:
        print("Hostname could not be resolved.")
        sys.exit(1)
    except socket.error:
        print("Network error: could not connect.")
        sys.exit(1)

if __name__ == "__main__":
    try:
        target = input("Enter target IP or hostname: ").strip()
        scan-ports(target)
    except Exception as e:
        print(f"Unexpected error: {}".format(e))
        sys.exit(1)
```

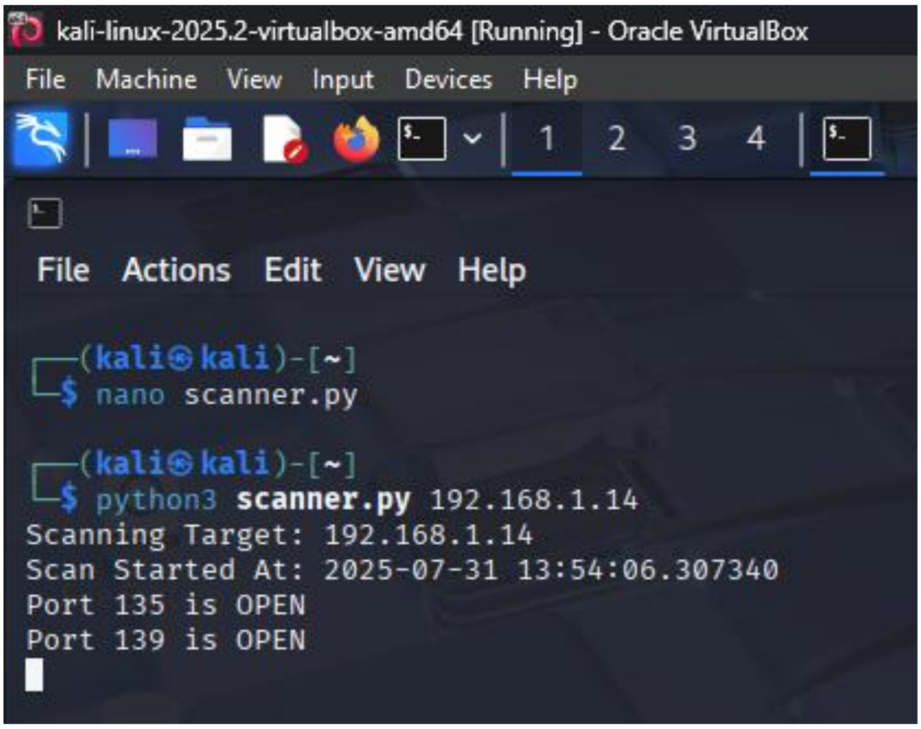
Sample output:

```
Enter target IP or hostname: scanme.nmap.org

Scanning scanme.nmap.org from port 1 to 1024...

Port 22 is OPEN
Port 80 is OPEN
```

Observation:


kali-linux-2025.2-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ nano scanner.py

┌──(kali㉿kali)-[~]
└─$ python3 scanner.py 192.168.1.14
Scanning Target: 192.168.1.14
Scan Started At: 2025-07-31 13:54:06.307340
Port 135 is OPEN
Port 139 is OPEN
```

Interpretation:

As per the scan conducted and the observed output of the ports:

- Port 135: Typically indicates a Microsoft Remote Procedure Call (RPC) service.
- Port 139: Typically indicates a NetBIOS Session Service. Open ports indicate a process is listening; actual services may vary.

Applications:

- **Network inventory and baseline:** Identify exposed services to maintain an accurate service inventory.
- **Security assessment:** Detect unintended or legacy services, reduce attack surface, and prioritize remediation.
- **Troubleshooting and validation:** Verify service availability during deployment or after firewall/ACL changes.
- **Compliance and governance:** Support periodic checks to ensure only approved services are exposed.
- **Ethical and legal considerations:** Scan only systems you own or have explicit permission to assess.

Conclusion:

This assignment demonstrates a clear, minimal implementation of a TCP port scanner using Python on Kali Linux. By leveraging socket.connect -ex() with sensible timeouts and straightforward control flow, the tool reliably enumerates open ports across a defined range. The approach emphasizes conceptual transparency over optimization, making it well-suited for instructional use while providing a base for future enhancements such as multithreading, banner grabbing, and result logging.