Distributed Port Scanning Project Report

1. Introduction

Port scanning is a vital cybersecurity technique used to identify open ports and potential vulnerabilities on networked devices. Traditional port scanners, limited by sequential or single-machine execution, are often slow when scanning large networks. This project develops a **distributed port scanning system** that leverages multiple workers to scan different IP addresses and port ranges in parallel, significantly improving speed and efficiency.

2. Objectives

- Develop a distributed port scanning framework using Python and Nmap.
- Implement a controller-worker architecture for efficient task distribution and result aggregation.
- Demonstrate the reduction in scanning time and enhanced scalability through distributed scanning.

3. Methodology

Architecture

- **Controller:** A central server that listens on a network socket, maintains a queue of scanning tasks (IP addresses and port ranges), and assigns tasks to connected workers.
- **Workers:** Clients that connect to the controller, receive scanning tasks, perform Nmap scans, and return results to the controller.

Task Distribution

- The controller manages a queue of tasks and assigns one task per worker at a time.
- Workers operate independently, scanning their assigned targets and reporting results back to the controller.

- Multiple workers can run concurrently, enabling parallel scanning across the network.

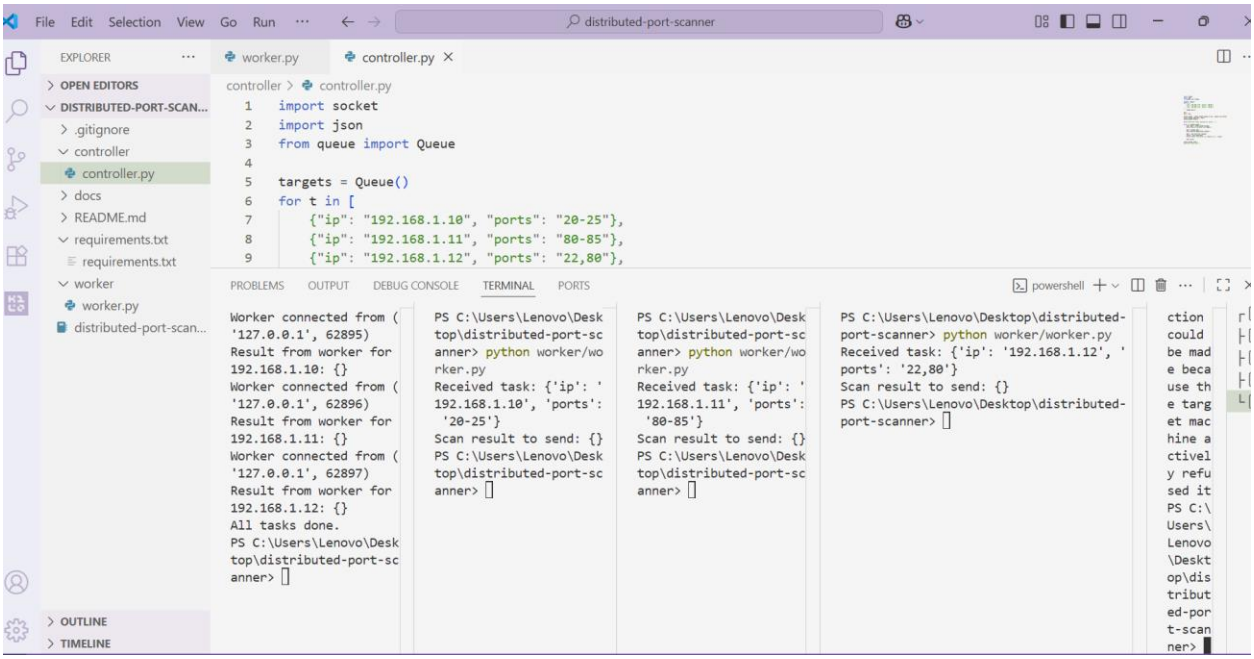
Tools

- Python's `socket` library for network communication between controller and workers.
- Python Nmap module (`python-nmap`) for automated port scanning.

4. Implementation

Controller

- Listens on TCP port 5055 for worker connections.
- Maintains a queue of IP addresses and port ranges to be scanned.
- Accepts worker connections, assigns tasks, and collects scan results.



Worker

- Connects to the controller using the specified IP and port.
- Receives task data in JSON format.
- Executes Nmap scans on the assigned IP and port range.
- Returns scan results to the controller in JSON format.

5. Results

- The system successfully distributed scanning tasks across multiple worker processes.
- Parallel scanning by workers significantly reduced total scan time compared to sequential scanning.
- Scan results accurately identified open ports, categorized by protocol (TCP/UDP).

6. Challenges and Limitations

- The controller processes workers sequentially; implementing multi-threading could enhance performance.
- Workers disconnect after completing a single task; persistent connections for multiple tasks would improve efficiency.
- Error handling for unreachable hosts or failed scans requires further development.
- Testing was conducted on a local or small network segment; large-scale network testing is needed for validation.

7. Conclusion and Future Work

This project successfully demonstrates a prototype of a distributed port scanning system using Python and Nmap. The distributed architecture enables parallel scanning, improving both speed and scalability. Future improvements include:

- Implementing a multi-threaded controller for concurrent task handling.
- Supporting persistent worker connections for multiple tasks.
- Enhancing error handling for robust operation.
- Testing the system on large-scale networks with multiple physical or virtual machines.

8. References

- Nmap Official Website: https://nmap.org
- Python socket programming documentation: https://docs.python.org/3/library/socket.html
- python-nmap GitHub: https://github.com/thezbyg/python-nmap