# PORT SCANNER – INTERNSHIP PROJECT DOCUMENTATION

Jagadeesh Kommineni

# Port Scanner – Internship Project Documentation

## Project Title

**Multi-threaded Port Scanner with Real-Time GUI using Python and Tkinter**

---

## Table of Contents


1. Overview
2. Objective
3. Technologies Used
4. Features
5. System Architecture and Workflow
6. Code Explanation
7. User Interface Design
8. Security and Ethical Considerations
9. Future Enhancements
10. Conclusion


---

## 1. Overview

This project presents a real-time graphical **Port Scanning Application** that enables users to test and detect open or closed ports on a target host (IP or domain) across a specified port range. It uses multi-threading for efficiency and provides a user-friendly graphical interface developed with **Tkinter**, Python's built-in GUI framework.

The scanner uses raw socket programming to attempt connections to TCP ports and visually represents the results in a responsive table with status categorization. It is suitable for security students, penetration testing beginners, or system administrators validating firewall rules and exposure.

---

## 2. Objective

The main goal of this project is to:

- Provide a reliable, responsive port scanning tool.
- Leverage multi-threading for high-speed scanning.
- Offer real-time GUI feedback for each scanned port.
- Maintain a clean, accessible, and intuitive interface.
- Display summarized scan results for better analysis.

---

# 3. Technologies Used

| Technology | Description |
|---|---|
| Python | Core programming language |
| Tkinter | GUI framework |
| Socket | Network communication for port testing |
| Threading | Multi-threaded scanning operations |
| time | Measuring scan durations |

---

# 4. Features

- Accepts IP address or hostname as input.
- User-defined start and end port range.
- Real-time update of scan results in a tree view.
- Multi-threaded scanning for high performance.
- Visual differentiation between open and closed ports.
- Displays total open ports and scan time in a summary popup.
- Option to stop scan midway using the "Stop Scan" button.

---

# 5. System Architecture and Workflow

[User Input]
  ↓
[GUI Interface - Tkinter]
  ↓
[Input Validation]
  ↓
[IP Resolution]
  ↓
[Threaded Port Scanning (TCP)]

↓
[Open/Closed Port Result]
↓
[Treeview Visualization]
↓
[Summary Report - Scan Completion]

---

# 6. Code Explanation

**a. scan_port(ip, port, tree)**

Attempts to establish a TCP connection to a single port.
If successful, port is marked as open; else closed.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.settimeout(0.5)
result = sock.connect_ex((ip, port))
```

**b. start_scan(...)**

Performs:

- Input validation
- IP resolution
- Thread initialization for each port in range
- Status update before and after scan
- Launches scan threads and displays popup summary

**c. stop_scan(...)**

Sets the global flag `is_scanning` to `False`, stopping further scans and updating UI status.

**d. show_popup(...)**

Displays a modal window summarizing:

- Number of open ports detected
- Time taken for the scan

---

# 7. User Interface Design

The UI is developed using Tkinter and customized with a dark color scheme for visual appeal and accessibility.

### Components:

- **Input Fields**: Target, Start Port, End Port
- **Buttons**: Start Scan, Stop Scan
- **Treeview Table**:
  - Columns: Port, Status
  - Tag colors: Open ports (dark teal), Closed ports (dark gray)
- **Status Bar**: Live status or messages
- **Summary Popup**: Scan completion report

### Design Principles:

- Minimalist and dark-themed for focus
- Clear distinction of output states
- Responsive and clean layout

---

## 8. Security and Ethical Considerations

- The tool is designed for educational and ethical use only.
- It does not attempt to bypass firewalls or intrusion detection systems.
- All scanning is limited to TCP connect scans, which are non-intrusive.
- Always seek permission before scanning any network or host not owned or controlled by the user.

---

## 9. Future Enhancements

| Feature | Description |
|---|---|
| UDP Port Scanning | Support for scanning UDP services |
| Export Feature | Allow exporting results to CSV or JSON |
| Banner Grabbing | Fetch service banners from open ports |
| Concurrent Threads Limit | Dynamic thread management for optimization |
| Scan Profiles | Save preferred IP and port ranges |
| Dark/Light Theme Toggle | UI customization options for better UX |
| Service Detection | Identify known services by port |

## 10. Conclusion

This project demonstrates proficiency in:

- Python programming
- Socket-based networking
- Multi-threaded applications
- Graphical user interface development

By integrating clean UI design with network-level functionality, this project effectively replicates behavior similar to lightweight scanners used in real-world network diagnostics or reconnaissance. It is well-suited for internship portfolios, academic project submissions, and cybersecurity learning environments.