# Port Scanner Using Python

**By Inlighn Tech**

## Objective:

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

## Project Overview:

Port scanning is a fundamental technique used in cybersecurity to detect open ports and running services on a networked system. This project implements a multi-threaded port scanner that scans a specified range of ports on a target IP, identifies open ports, and attempts to retrieve service banners. The scanner provides insights into potential security vulnerabilities by revealing exposed services.

## How the Project 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.

### Expected Outcomes:

By completing this project, students will:

- Gain hands-on experience with socket programming and network scanning.
- Understand the importance of port scanning in cybersecurity.
- Learn how to implement efficient multithreading for performance optimization.
- Develop a real-world tool used in ethical hacking and penetration testing.

### Next Steps:

**Students should implement their own version of the port scanner using the outlined concepts. A video lecture will be provided later to demonstrate the correct implementation and solution. This project serves as a foundational step for network security and penetration testing tasks in Python.**

**For further enhancements, students can:**

- **Scan UDP Ports: Extend the scanner to detect open UDP ports in addition to TCP.**
- **Integrate OS Detection: Identify the operating system of the target machine.**
- **Develop a GUI: Create a graphical interface for better usability and visualization of scan results.**