Project Report: Port Scanner Using Python

Title: Port Scanner Using Python

Developer: Shubham Parab

Language: Python 3.12.4

Objective:

1.Introduction

A **port scanner** is a tool used to identify open ports and services available on a target machine. Network administrators and cybersecurity professionals use port scanners to audit system security, while attackers may use them for malicious reconnaissance. This project involves developing a simple yet effective port scanner using Python to help understand the basics of socket programming and network analysis.

2. Requirements

**Software Requirements:**

* Python 3.12.4
* Code editor (VS Code, PyCharm, or any text editor)
* Command-line interface (Terminal or CMD)

**Python Modules:**

* socket
* threading (for multi-threaded version)
* datetime (for optional timestamp logging)

**Hardware Requirements:**

* A computer with internet or LAN access

3. Code Overview

import socket

import concurrent.futures

import tkinter as tk

from tkinter import messagebox, ttk, scrolledtext, filedialog

import csv

import datetime

import subprocess

import platform

# Function to grab banner

def get\_banner(ip, port):

    try:

        with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

            s.settimeout(timeout\_val.get())

            s.connect((ip, port))

            return s.recv(1024).decode(errors='ignore').strip()

    except:

        return ""

# Function to scan a single port

def scan\_port(ip, port):

    try:

        with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

            s.settimeout(timeout\_val.get())

            result = s.connect\_ex((ip, port))

            if result == 0:

                try:

                    service = socket.getservbyport(port)

                except:

                    service = "Unknown"

                banner = get\_banner(ip, port)

                return (port, service, banner)

    except Exception as e:

        with open("error\_log.txt", "a") as err\_log:

            err\_log.write(f"Error on port {port}: {e}\n")

        return None

    return None

def validate\_ip(ip):

    try:

        socket.gethostbyname(ip)

        return True

    except socket.error:

        return False

def start\_scan():

    ip = ip\_entry.get().strip()

    if not validate\_ip(ip):

        messagebox.showerror("Invalid IP", "Enter a valid IP address or hostname.")

        return

    try:

        start\_port = int(start\_port\_entry.get())

        end\_port = int(end\_port\_entry.get())

    except ValueError:

        messagebox.showerror("Invalid Input", "Port range must be integers.")

        return

    if start\_port < 0 or end\_port > 65535 or start\_port > end\_port:

        messagebox.showerror("Invalid Range", "Enter a valid port range (0-65535).")

        return

    try:

        hostname = socket.gethostbyaddr(ip)[0]

    except socket.herror:

        hostname = "Unknown"

    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    scan\_log.append(f"{timestamp} - Scanned {ip} ({hostname}) ports {start\_port}-{end\_port}")

    results\_text.config(state='normal')

    results\_text.delete(1.0, tk.END)

    results\_text.insert(tk.END, f"Scanning {ip} ({hostname}) from port {start\_port} to {end\_port}...\n")

    results\_text.insert(tk.END, f"{'Port':<10}{'Service':<20}Banner\n")

    results\_text.insert(tk.END, "-"\*60 + "\n")

    open\_ports.clear()

    total\_ports = end\_port - start\_port + 1

    progress\_bar['value'] = 0

    progress\_bar['maximum'] = total\_ports

    def scan\_and\_display():

        max\_threads = thread\_count.get()

        with concurrent.futures.ThreadPoolExecutor(max\_workers=max\_threads) as executor:

            futures = {executor.submit(scan\_port, ip, port): port for port in range(start\_port, end\_port + 1)}

            completed = 0

            for future in concurrent.futures.as\_completed(futures):

                result = future.result()

                if result:

                    open\_ports.append(result)

                    port, service, banner = result

                    results\_text.insert(tk.END, f"{port:<10}{service:<20}{banner}\n")

                completed += 1

                progress\_bar['value'] = completed

                root.update\_idletasks()

        if not open\_ports:

            results\_text.insert(tk.END, "\nNo open ports found in the specified range.")

        results\_text.config(state='disabled')

    root.after(100, scan\_and\_display)

def run\_nmap():

    ip = ip\_entry.get().strip()

    if not validate\_ip(ip):

        messagebox.showerror("Invalid IP", "Enter a valid IP address or hostname.")

        return

    results\_text.config(state='normal')

    results\_text.delete(1.0, tk.END)

    results\_text.insert(tk.END, f"Running Nmap on {ip}...\n")

    try:

        command = ["nmap", "-sV", ip]

        result = subprocess.run(command, capture\_output=True, text=True, timeout=30)

        results\_text.insert(tk.END, result.stdout)

    except FileNotFoundError:

        results\_text.insert(tk.END, "Nmap is not installed or not found in PATH.")

    except Exception as e:

        results\_text.insert(tk.END, f"Error running Nmap: {e}")

    results\_text.config(state='disabled')

def scan\_common\_ports():

    ip = ip\_entry.get().strip()

    if not validate\_ip(ip):

        messagebox.showerror("Invalid IP", "Enter a valid IP address or hostname.")

        return

    common\_ports = [21, 22, 23, 25, 53, 80, 110, 139, 143, 443, 445, 993, 995, 1723, 3306, 3389, 5900, 8080]

    try:

        hostname = socket.gethostbyaddr(ip)[0]

    except socket.herror:

        hostname = "Unknown"

    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    scan\_log.append(f"{timestamp} - Scanned {ip} ({hostname}) common ports")

    results\_text.config(state='normal')

    results\_text.delete(1.0, tk.END)

    results\_text.insert(tk.END, f"Scanning common ports on {ip} ({hostname})...\n")

    results\_text.insert(tk.END, f"{'Port':<10}{'Service':<20}Banner\n")

    results\_text.insert(tk.END, "-"\*60 + "\n")

    open\_ports.clear()

    progress\_bar['value'] = 0

    progress\_bar['maximum'] = len(common\_ports)

    def scan\_and\_display():

        max\_threads = thread\_count.get()

        with concurrent.futures.ThreadPoolExecutor(max\_workers=max\_threads) as executor:

            futures = {executor.submit(scan\_port, ip, port): port for port in common\_ports}

            completed = 0

            for future in concurrent.futures.as\_completed(futures):

                result = future.result()

                if result:

                    open\_ports.append(result)

                    port, service, banner = result

                    results\_text.insert(tk.END, f"{port:<10}{service:<20}{banner}\n")

                completed += 1

                progress\_bar['value'] = completed

                root.update\_idletasks()

        if not open\_ports:

            results\_text.insert(tk.END, "\nNo common ports open.")

        results\_text.config(state='disabled')

    root.after(100, scan\_and\_display)

def save\_results():

    content = results\_text.get(1.0, tk.END)

    with open("scan\_results.txt", "w") as f:

        f.write(content)

    messagebox.showinfo("Saved", "Results saved to scan\_results.txt")

def export\_to\_csv():

    if not open\_ports:

        messagebox.showwarning("No Data", "No results to export.")

        return

    file\_path = filedialog.asksaveasfilename(defaultextension=".csv", filetypes=[("CSV Files", "\*.csv")])

    if file\_path:

        with open(file\_path, "w", newline='') as f:

            writer = csv.writer(f)

            writer.writerow(["Port", "Service", "Banner"])

            for row in open\_ports:

                writer.writerow(row)

        messagebox.showinfo("Exported", f"Results exported to {file\_path}")

def copy\_to\_clipboard():

    root.clipboard\_clear()

    root.clipboard\_append(results\_text.get(1.0, tk.END))

    messagebox.showinfo("Copied", "Results copied to clipboard")

def toggle\_theme():

    if style.theme\_use() == "clam":

        style.theme\_use("default")

    else:

        style.theme\_use("clam")

# GUI Setup

root = tk.Tk()

root.title("Pro Python Port Scanner")

root.geometry("800x650")

root.resizable(True, True)

style = ttk.Style()

style.theme\_use("clam")

frame = ttk.Frame(root, padding="10")

frame.pack(fill=tk.BOTH, expand=True)

open\_ports = []

scan\_log = []

# Inputs

ttk.Label(frame, text="Target IP or Hostname:").grid(row=0, column=0, sticky=tk.W, pady=5)

ip\_entry = ttk.Entry(frame, width=30)

ip\_entry.grid(row=0, column=1, pady=5, sticky=tk.W)

ttk.Label(frame, text="Start Port:").grid(row=1, column=0, sticky=tk.W, pady=5)

start\_port\_entry = ttk.Entry(frame, width=10)

start\_port\_entry.grid(row=1, column=1, sticky=tk.W, pady=5)

ttk.Label(frame, text="End Port:").grid(row=2, column=0, sticky=tk.W, pady=5)

end\_port\_entry = ttk.Entry(frame, width=10)

end\_port\_entry.grid(row=2, column=1, sticky=tk.W, pady=5)

ttk.Label(frame, text="Timeout (s):").grid(row=3, column=0, sticky=tk.W, pady=5)

timeout\_val = tk.DoubleVar(value=1.0)

timeout\_entry = ttk.Entry(frame, textvariable=timeout\_val, width=10)

timeout\_entry.grid(row=3, column=1, sticky=tk.W, pady=5)

ttk.Label(frame, text="Threads:").grid(row=4, column=0, sticky=tk.W, pady=5)

thread\_count = tk.IntVar(value=100)

thread\_entry = ttk.Entry(frame, textvariable=thread\_count, width=10)

thread\_entry.grid(row=4, column=1, sticky=tk.W, pady=5)

# Buttons

scan\_button = ttk.Button(frame, text="Start Scan", command=start\_scan)

scan\_button.grid(row=5, column=0, pady=10, sticky=tk.W)

common\_ports\_button = ttk.Button(frame, text="Scan Common Ports", command=scan\_common\_ports)

common\_ports\_button.grid(row=5, column=1, pady=10, sticky=tk.E)

nmap\_button = ttk.Button(frame, text="Run Nmap", command=run\_nmap)

nmap\_button.grid(row=5, column=2, pady=10, padx=10)

theme\_button = ttk.Button(frame, text="Toggle Theme", command=toggle\_theme)

theme\_button.grid(row=5, column=3, pady=10, padx=10)

# Output

progress\_bar = ttk.Progressbar(frame, orient='horizontal', length=700, mode='determinate')

progress\_bar.grid(row=6, column=0, columnspan=4, pady=5)

results\_text = scrolledtext.ScrolledText(frame, wrap=tk.WORD, width=95, height=20, state='disabled')

results\_text.grid(row=7, column=0, columnspan=4, pady=10)

# Extra Buttons

button\_frame = ttk.Frame(frame)

button\_frame.grid(row=8, column=0, columnspan=4, pady=5)

save\_button = ttk.Button(button\_frame, text="Save to TXT", command=save\_results)

save\_button.grid(row=0, column=0, padx=5)

csv\_button = ttk.Button(button\_frame, text="Export to CSV", command=export\_to\_csv)

csv\_button.grid(row=0, column=1, padx=5)

clipboard\_button = ttk.Button(button\_frame, text="Copy to Clipboard", command=copy\_to\_clipboard)

clipboard\_button.grid(row=0, column=2, padx=5)

root.mainloop()

4. How It Works

1. **Input Target and Port Range:**  
   The user provides the target IP address/domain and a range of ports to scan.
2. **Initialize Socket Connection:**  
   Python’s socket module is used to create connections to the specified ports.
3. **Scan Ports:**  
   The program attempts to connect to each port within the specified range. If the connection is successful, the port is marked as open.
4. **Display Results:**  
   The scanner outputs which ports are open and which are closed.
5. **Optional Features:**
   * Multithreaded scanning for speed
   * Logging with timestamps
   * Host resolution

5. Execution

Step-by-Step Execution:

Save the Python script as port\_scanner.py.

Open a terminal and navigate to the script’s directory.

Run the script using:

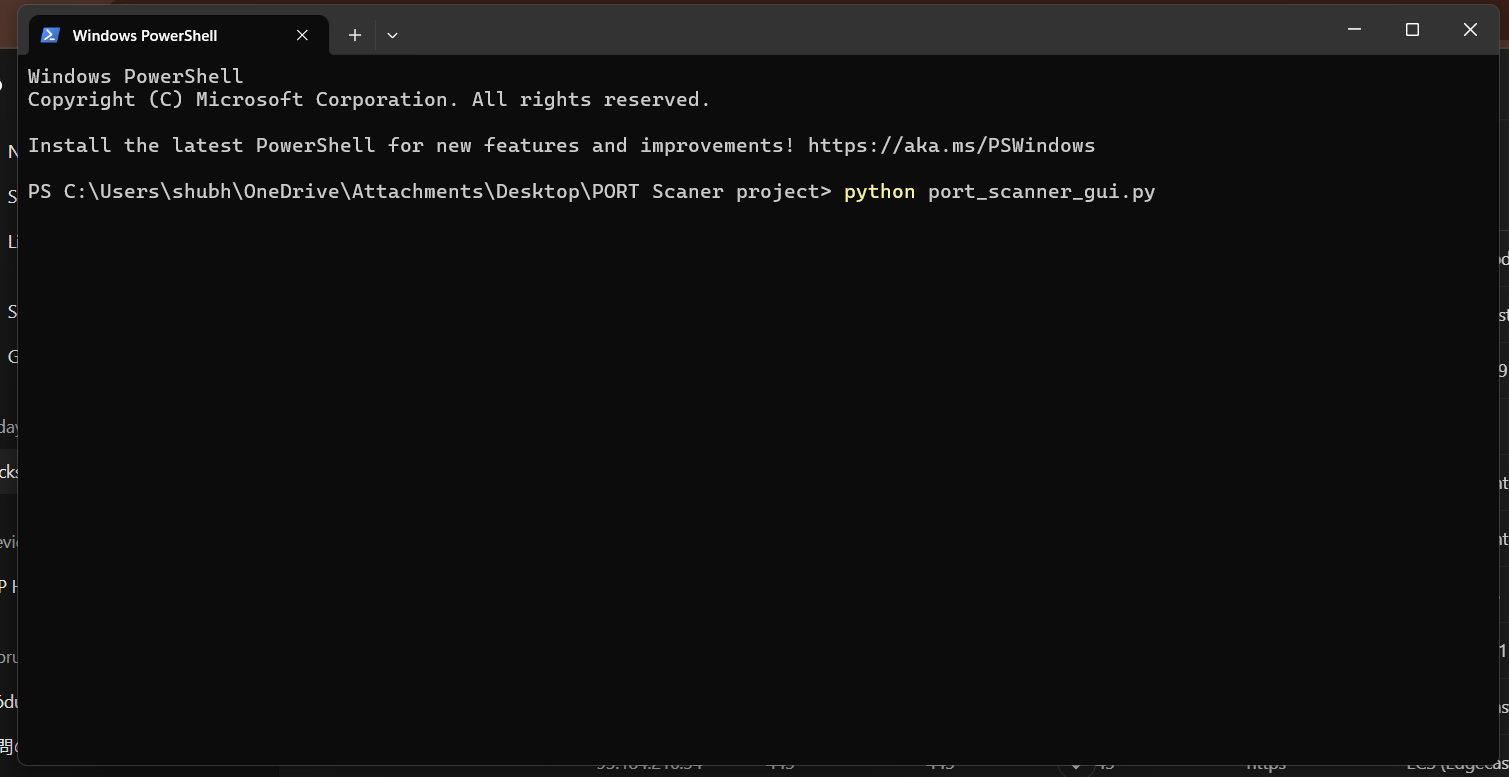
nginx

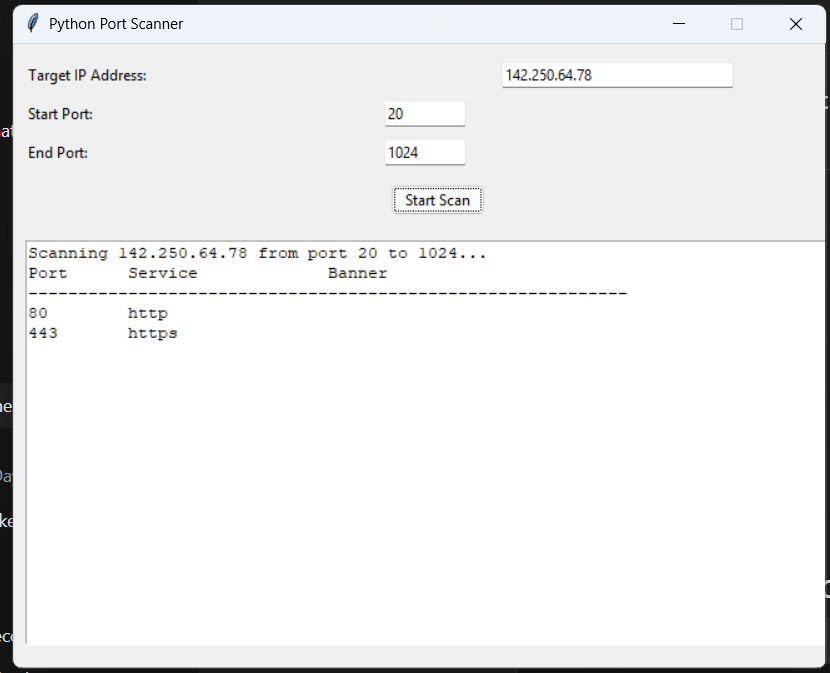
python port\_scanner.py

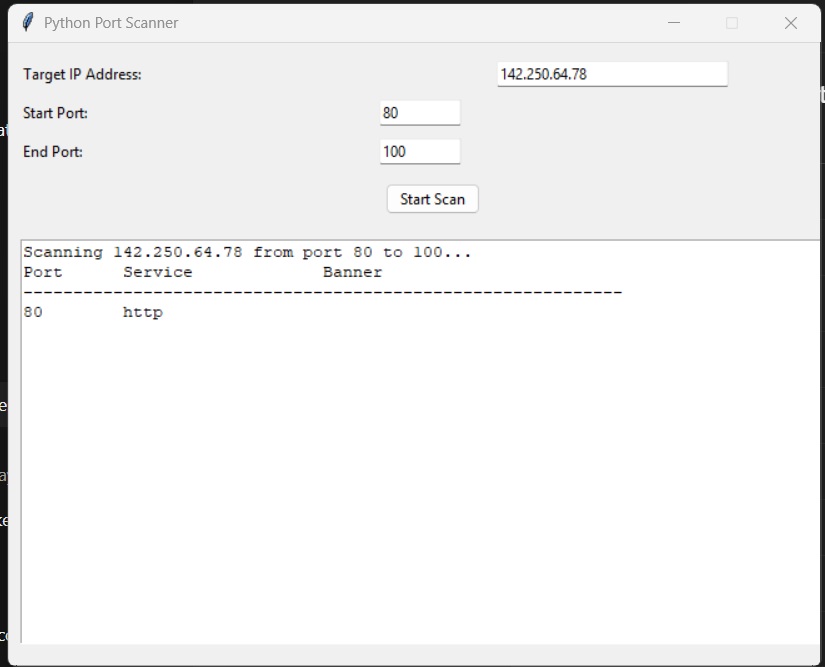
Enter the target host and port range as prompted.

View the scan results directly in the terminal.

6. Screenshots







7. Use Cases

Network Security Auditing:

Identify open ports on devices and ensure that unnecessary ports are closed.

Educational Tool:

Help students and beginners learn about network scanning and socket programming.

System Administration:

Check for reachable services on internal servers

8. Conclusion

This project demonstrates how Python can be used to build practical network tools like a port scanner. By leveraging built-in libraries like socket and threading, the scanner offers insight into how port scanning works under the hood. It provides a valuable learning experience for those interested in cybersecurity, networking, or Python programming.