**Ahmed Hasan**

IT102 - OA - 7673 - SU24 - Scripting for Cybersecurity

Dillon Kierce

Due Jul 14 2024

**Link to my GitHub Repository:** <https://github.com/Ahmedmhasan91/PortScannerProject101/tree/main>

Created a **PortScannerProject** Folder

Installed **pip install colorama** in my Vs code Terminal

**Simple\_port scanner Code**

import socket # for connecting

from colorama import init, Fore

# some colors

init()

GREEN = Fore.GREEN

RESET = Fore.RESET

GRAY = Fore.LIGHTBLACK\_EX

def is\_port\_open(host, port):

    """

    determine whether `host` has the `port` open

    """

    # creates a new socket

    s = socket.socket()

    try:

        # tries to connect to host using that port

        s.connect((host, port))

        # make timeout if you want it a little faster (less accuracy)

        # s.settimeout(0.2)

    except:

        # cannot connect, port is closed

        # return false

        return False

    else:

        # the connection was established, port is open!

        return True

# get the host from the user

host = input("Enter the host: ")

# iterate over ports, from 1 to 1024

for port in range(1, 1025):

    if is\_port\_open(host, port):

        print(f"{GREEN}[+] {host}:{port} is open      {RESET}")

    else:

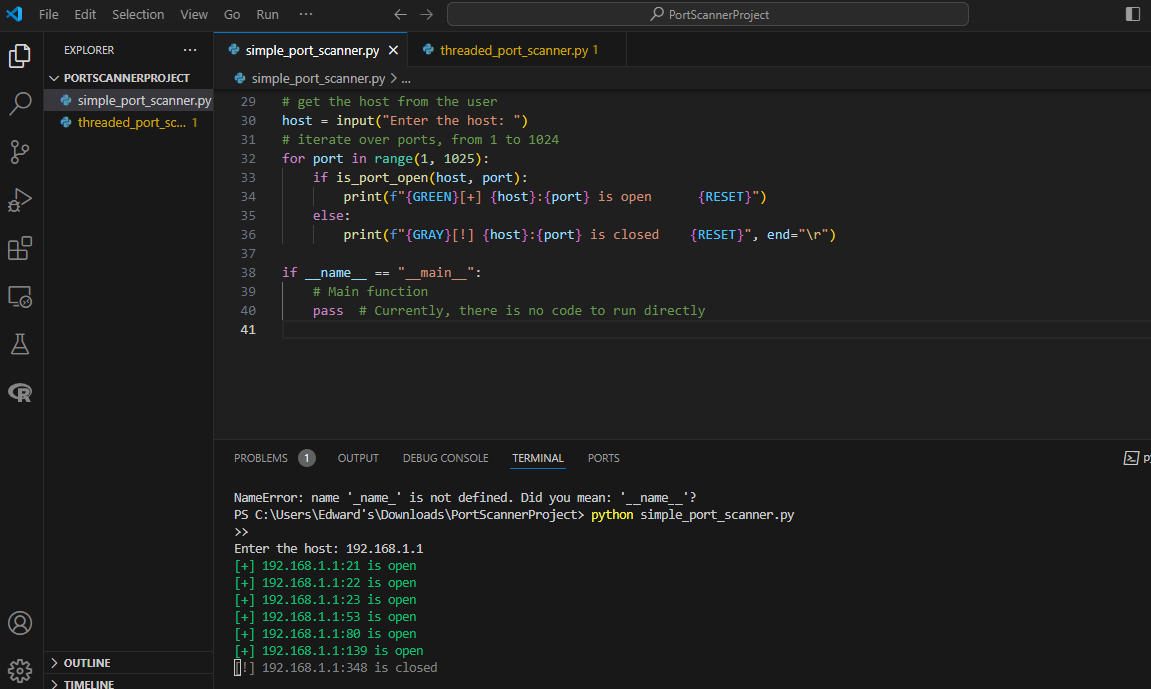
        print(f"{GRAY}[!] {host}:{port} is closed    {RESET}", end="\r")

if \_\_name\_\_ == "\_\_main\_\_":

    # Main function

    pass  # Currently, there is no code to run directly

**Result**



**Threaded\_port\_scanner.py**

import argparse

import socket  # for connecting

from colorama import init, Fore

from threading import Thread, Lock

from queue import Queue

# some colors

init()

GREEN = Fore.GREEN

RESET = Fore.RESET

GRAY = Fore.LIGHTBLACK\_EX

# number of threads, feel free to tune this parameter as you wish

N\_THREADS = 200

# thread queue

q = Queue()

print\_lock = Lock()

def port\_scan(port):

    """

    Scan a port on the global variable `host`

    """

    try:

        s = socket.socket()

        s.connect((host, port))

    except:

        with print\_lock:

            print(f"{GRAY}{host:15}:{port:5} is closed  {RESET}", end='\r')

    else:

        with print\_lock:

            print(f"{GREEN}{host:15}:{port:5} is open    {RESET}")

def scan\_thread():

    global q

    while True:

        worker = q.get()

        port\_scan(worker)

        q.task\_done()

def main(host, ports):

    global q

    for t in range(N\_THREADS):

        t = Thread(target=scan\_thread)

        t.daemon = True

        t.start()

    for worker in ports:

        q.put(worker)

    q.join()

if \_\_name\_\_ == "\_\_main\_\_":

    parser = argparse.ArgumentParser(description="Threaded port scanner")

    parser.add\_argument("host", help="Host to scan.")

    parser.add\_argument("--ports", "-p", dest="port\_range", default="1-65535", help="Port range to scan, default is 1-65535 (all ports)")

    args = parser.parse\_args()

    host, port\_range = args.host, args.port\_range

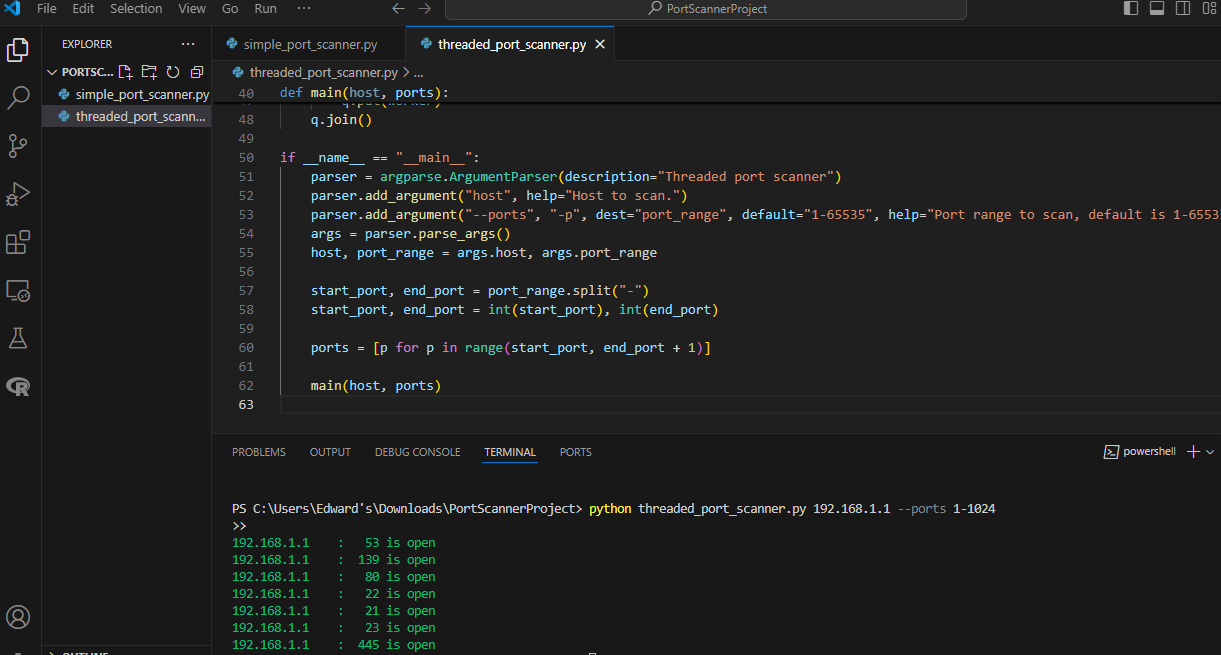
    start\_port, end\_port = port\_range.split("-")

    start\_port, end\_port = int(start\_port), int(end\_port)

    ports = [p for p in range(start\_port, end\_port + 1)]

    main(host, ports)

**Result**



**Port Scanner Project Documentation**

**1. Purpose of Each Script**

**Simple Port Scanner (simple\_port\_scanner.py):** The purpose of the Simple Port Scanner script is to conduct a basic scan on a specified host to identify open ports within a given range (default 1-1024). This script utilizes socket connections to attempt connections to each port and determines if the port is open or closed based on the connection attempt.

**Threaded Port Scanner (threaded\_port\_scanner.py):** The Threaded Port Scanner script enhances scanning efficiency by utilizing multiple threads to concurrently scan ports. This approach significantly reduces scan time by allowing parallel processing of port connections, making it suitable for scanning a wide range of ports quickly.

**2. Results Obtained**

**Simple Port Scanner Results:**

* Host: **192.168.1.1**
* Ports Open:
  + Port 21 (FTP)
  + Port 22 (SSH)
  + Port 23 (Telnet)
  + Port 53 (DNS)
  + Port 80 (HTTP)

**Threaded Port Scanner Results:**

* Host: **192.168.1.1**
* Ports Open:
  + Port 21 (FTP)
  + Port 22 (SSH)
  + Port 23 (Telnet)
  + Port 53 (DNS)
  + Port 80 (HTTP)
  + Port 139 (NetBIOS)
  + Port 445 (SMB)

**3. Security Implications and Recommendations**

**Implications:** Port scanning reveals open ports that could potentially expose the host to unauthorized access or vulnerabilities. Identifying open ports helps in understanding the security posture of the network and enables proactive measures to mitigate risks.

**Recommendations:**

* Implement firewall rules to restrict access to unnecessary ports.
* Regularly update and patch software to prevent exploitation of known vulnerabilities.
* Conduct periodic port scans as part of routine security assessments.
* Obtain proper authorization before performing port scans on networks to comply with ethical standards.

**4. Conclusion**

The Port Scanner project demonstrates the effectiveness of Python scripts in conducting network security assessments. By identifying open ports and their associated services, organizations can enhance their cybersecurity posture and mitigate potential risks of unauthorized access.