**Project 2: Port Scanner Using Python**

**Problem Statement:** Network ports are entry points for services; open or vulnerable ports can be exploited.

**Objective:** Build a port scanner to detect open ports on a given IP address or website.

**Requirements:**

Python

socket and threading modules

Command-line interface

**Expected Outcome:** A script that lists open ports and helps in basic vulnerability assessment of a network.

**Solution:**

**Objective:** To detect open ports on a given IP address or website using Python. This helps identify potential entry points for attackers.

**Project Summary**

This tool scans for open ports on a given IP address or domain name using Python's socket module and multithreading to enhance speed. It's suitable for basic vulnerability assessment.

**Structure**

port\_scanner\_project/

port\_scanner.py # Main scanner script

README.md # Documentation

**Code:**

import socket

import threading

import sys

from queue import Queue

# Configuration

MAX\_THREADS = 100

print\_lock = threading.Lock()

queue = Queue()

# Port Scanner Function

def scan\_port(target, port):

try:

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

s.settimeout(1) # Timeout for connection attempt

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

if result == 0:

with print\_lock:

print(f"[+] Port {port} is OPEN")

except Exception:

pass # Ignore errors silently for clean output

# Worker Thread Function

def threader(target):

while True:

worker = queue.get()

scan\_port(target, worker)

queue.task\_done()

# Main Scanner Setup

def start\_scan(target, port\_range=(1, 1024)):

print(f"\nScanning Target: {target}")

print(f"Ports: {port\_range[0]} to {port\_range[1]}\n")

for \_ in range(MAX\_THREADS):

t = threading.Thread(target=threader, args=(target,), daemon=True)

t.start()

for port in range(port\_range[0], port\_range[1] + 1):

queue.put(port)

queue.join()

print("\n Scan Completed.")

# Domain Resolver

def resolve\_target(target):

try:

return socket.gethostbyname(target)

except socket.gaierror:

print(" Error: Invalid domain or IP.")

sys.exit()

# CLI Interface

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

print(" Python Port Scanner\n--------")

target = input("Enter IP or domain name: ").strip()

port\_choice = input(" Scan default (1-1024) ports? (y/n): ").strip().lower()

if port\_choice == 'n':

try:

start\_port = int(input("Start port: "))

end\_port = int(input("End port: "))

except ValueError:

print(" Invalid port number.")

sys.exit()

else:

start\_port, end\_port = 1, 1024

resolved\_ip = resolve\_target(target)

start\_scan(resolved\_ip, port\_range=(start\_port, end\_port))

**Features**

* Fast scanning using threading
* User-friendly command-line interface
* Scans top common ports or a custom range
* Automatically resolves domain to IP

**Tools Used**

- Python

- socket module

- threading module

- queue module

**How to Run**

python port\_scanner.py

**output:**

