**👩‍💻 Developer:**

Anushree401

**📁 Repository:**

GitHub: [TCP-network-scanner](https://github.com/Anushree401/TCP-network-scanner)

**📌 Overview:**

This project implements a TCP port scanner using Python, leveraging the Nmap module (python-nmap) and the standard library tools like argparse, sys, and colorama. It allows users to scan a target IP address or hostname for open ports, displaying details about services running on those ports.

**⚙️ Features:**

* Accepts target hostname/IP and port range as input.
* Uses argparse for CLI interface.
* Integrates with nmap to perform efficient and detailed scanning.
* Displays service and port information in a readable format.
* Uses colorama for color-coded terminal output.
* Verbose mode to show closed or filtered ports (optional).
* Handles exceptions for invalid input or scanning errors.

**🧱 Technologies Used:**

|  |  |
| --- | --- |
| **Tool/Library** | **Purpose** |
| **Python** | Programming language |
| Argparse | Command-line argument parsing |
| Nmap | Network scanning backend |
| python-nmap | Python wrapper for Nmap |
| Colorama | Colored terminal output |
| Sys | System-level operations |

import argparse

import nmap

import sys

from colorama import Fore, Style

**🔧 Core Functionality**

It scans the TCP ports of a given host (IP or hostname), and:

* Tells you which ports are **open**, **filtered**, or **closed**.
* Tries to identify what **service** is running on each open port.
* Supports **verbose** mode to show all port states, not just open ones.
* Handles **single ports**, **comma-separated lists**, and **port ranges**.

**📌 Breakdown of Each Part:**

**✅ argument\_parser()**

Parses command-line arguments:

* --host (-H): IP address/hostname to scan (default: 127.0.0.1)
* --port (-p): Comma-separated list or range (default: 80)
* --verbose (-v): Shows all ports, not just open ones

Returns: A dictionary of arguments.

**✅ nmap\_scan(host\_id, port\_num, verbose=False)**

Handles scanning:

1. **Validates input ports**.
2. **Checks if the host is alive** (ping scan).
3. **Performs TCP scan** using -sT (connect scan) on the specified ports.
4. For each port or port range:
   * Calls process\_port() to extract info and format the result.

Returns: A list of formatted scan result strings.

**✅ process\_port(scanner, host\_id, port, result, verbose=False)**

Processes a **single port**:

* If open: adds detailed info about host, port, and service.
* If closed or filtered:
  + Only added if verbose mode is on.
* Catches KeyError in case no response is received for that port.

Adds results directly to the result list.

**✅ main()**

The main function:

* Parses arguments using argument\_parser().
* Calls nmap\_scan() and stores results.
* Prints all results.
* Summarizes:
  + Number of open ports.
  + Hidden ports if verbose was off.
* Handles exceptions like:
  + KeyboardInterrupt
  + AttributeError (missing/invalid args)
  + Any unexpected error

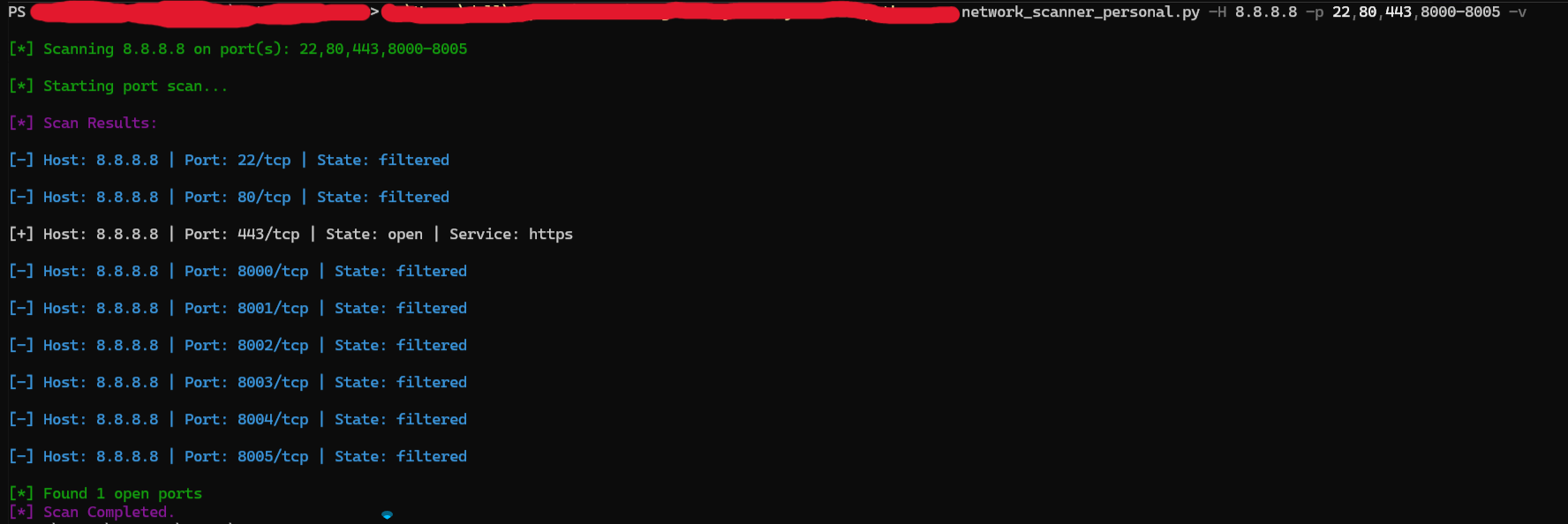
**🔍 Sample Usage:**

python network\_scanner\_personal.py -H 192.168.1.1 -p 22,80,443,8000-8005 -v

This will:

Scan host 192.168.1.1 on ports 22, 80, 443, and 8000 to 8005.

Show all ports including closed/filtered.



**🛡️ Error Handling:**

* Invalid IP/hostname
* Improper port formats
* Nmap not installed or permission denied
* Missing arguments

**📈 Applications:**

* Network auditing
* Penetration testing
* Learning and practicing port scanning concepts
* Custom security scripts in enterprise environments

**✅ Future Improvements:**

* Add support for UDP scans
* Export results to CSV/JSON
* Web GUI for user-friendly interaction
* Multi-threaded scanning for speed
* Scan entire subnet ranges

**📚 References:**

* Nmap Documentation
* [Python-nmap GitHub](https://github.com/alexxa/python-nmap)
* [Colorama Docs](https://pypi.org/project/colorama/)