Journal of Science and Technology

ISSN: 2456-5660 Volume 8, Issue 12 (Dec -2023)

www.jst.org.in *DOI:https://doi.org/10.46243/jst.2023.v8.i12.pp17-22*

OPEN PORT SCANNER TO IDENTIFY OPEN PORTS USING PYTHON

Marni Rohit[1], Madgul Udaykiran Reddy[2],Koyyala Chandra Teja[3], V Bala[4]

[1,2,3]B.Tech Student,Department Of CSE(Cyber Security),Malla Reddy College Of Engineering and Technology,Hyderabad,India.

[4]Associate Professor, Department Of CSE (Cyber Security), Malla Reddy College Of Engineering and Technology, Hyderabad,India.

To Cite this Article

Marni Rohit, Madgul Udaykiran Reddy, Koyyala Chandra Teja, V Bala , " OPEN PORT SCANNER TO IDENTIFY OPEN PORTS USING PYTHON " *Journal of Science and Technology, Vol. 08, Issue 12,- Dec 2023, pp17-22*

Article Info

Received: 12-11-2023 Revised: 22-11-2023 Accepted: 02-12-2023 Published: 12-12-2023

ABSTRACT:

An Open Port Scanner is a vital tool for identifying open network ports on a target system, which is crucial for network security assessment and troubleshooting. This abstract introduces a Python-based Open Port Scanner designed to help network administrators and security professionals quickly and efficiently identify open ports on a target host. The tool leverages Python's socket library to establish connections with various network services on the target system, and it can be customized to scan specific port ranges or even conduct comprehensive scans on multiple hosts. The Open Port Scanner operates by systematically probing a range of TCP or UDP ports on a target host to determine their open or closed status. It provides real-time feedback on the ports' accessibility, making it a valuable asset for identifying potential vulnerabilities and unauthorized services. The tool also includes features for customizing scan parameters, such as setting timeout values and specifying multiple target hosts. The results of the scan are presented in a user-friendly format, enabling users to easily identify open ports and assess the network's security posture. This Python-based Open Port Scanner serves as a powerful and flexible solution for network administrators and security experts seeking to secure their systems and networks by proactively identifying and addressing potential vulnerabilities. Its open-source nature allows for further customization and integration into existing network security workflows, making it a valuable addition to the toolkit of those responsible for maintaining the integrity of network infrastructure.

I INTRODUCTION

In the realm of network security and diagnostics, understanding the state of ports and vulnerabilities is essential.

The presented Python script offers a multifaceted toolset, combining port scanning and UDP flooding capabilities. Developed by Rohit, UdayKiran and Chandra Tej, the script enables users to scan a range of ports on a target IP address and flood a selected port with UDP packets, simulating a load and testing the network's resilience. The script begins by setting parameters such as delay, port scan precision, and port scan

ISSN: 2456-5660 Volume 8, Issue 12 (Dec -2023)
www.jst.org.in *DOI:https://doi.org/10.46243/jst.2023.v8.i12.pp17-22*

interval. It initializes a UDP socket to craft and send UDP packets of varying sizes to the chosen port on the specified IP address. On the other hand, it employs threads to concurrently scan ports, identifying those in a 'Listening' state. Users are prompted to input necessary details, ensuring flexibility and user-interaction. The script provides real-time updates, displaying progress and results during the scanning and flooding processes.

The Open Port Scanner revolves around the creation of a sophisticated tool designed to assess the security status of network systems comprehensively. In today's interconnected digital landscape, ensuring the integrity of network infrastructure is paramount. The scanner will serve as an indispensable asset for cybersecurity professionals, enabling them to identify open ports and potential vulnerabilities that could be exploited by malicious actors. By delivering an intuitive user interface, supporting multiple scan techniques, and generating detailed reports, the project aims to provide an efficient and user-friendly solution for network security assessment. Ethical considerations and responsible use will remain at the forefront of our project, ensuring that users are empowered to conduct scans responsibly and within the bounds of legal and ethical frameworks.

The tool is designed with a disclaimer, emphasizing responsible usage and non-endorsement of any misuse. By facilitating network analysis and stress testing, it empowers users to enhance network security and performance.

II LITERATURE SURVEY

According to Authors RajniRanjan Singh and Deepak Singh Tomar, Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, M.P., India., Port scanning is a process of probing networks, finding vulnerabilities and then infiltrate IT recourses. It is often the fundamental method utilized by intruder prior to initiate a targeted cyber-attack. Port scan attack traffic does not contain any specific signature, therefore IDS based detection may suffer by generating many/false alerts. Manual examination is an error prone, labor intensive and time consuming process. This work presented an approach to detect port scanning attack based on the entropy and failed connection attempt made by each host. To analyze and prioritize the observed evidence, Dempster-Shafer theory is utilized to calculate combined belief of each host in support of the proposed hypothesis.

TCP port scanners are specialized programs used to determine what TCP ports of a host have processes listening on them for possible connections. Since these ports characterize, in part, the amount of exposure of the hosts to potential external attacks, knowing their existence is a fundamental matter for network and/or security administrators. Moreover, as scanners are also used by hackers, administrators need to know how they work and what possible weakness they exploit to be able to prevent unwanted scanning or at least to record each scanning attempt.

III WORKING METHODOLOGY

The working of an open port scanner to identify open ports using Python involves establishing connections to specific ports on a target system to determine whether they are open, closed, or filtered. Here's a step-by-step explanation of how such a scanner operates:

Input Parameters:

The user provides input parameters to the scanner, including the target IP address or hostname and a range of ports to scan. Additional options may include the choice of scanning technique (e.g., TCP, UDP), timeout values, and output preferences.

Socket Creation:

The Python program uses the socket library to create a socket, which is the endpoint for network communication. The scanner creates a socket for each port to be scanned.

Port Loop:

The scanner enters a loop to iterate over the specified range of ports. For each port in the range, the following steps are performed:

Socket Connection Attempt:

The scanner attempts to establish a connection to the target host's IP address and the current port using the created socket. The specific method used (e.g., TCP connect, SYN, or UDP) depends on the scanning technique chosen.

Handling Connection Result:

Depending on the outcome of the connection attempt, the scanner records the status of the port:

If the connection is successful, the port is marked as "open."

If the connection is refused, the port is marked as "closed."

If the connection times out or encounters an error, the port status may be marked as "filtered" or "unreachable."

The purpose of this methodology thesis is to elucidate the methods and techniques employed in the development of the "Port Scanner and DDoS Attack Script" written in Python. The script's primary functions include scanning ports on a specified IP address and conducting a Distributed Denial of Service (DDoS) attack. The methodology covers key aspects such as code structure, socket programming, threading, and packet sending strategies.

Code Structure

The script is structured in a modular fashion, employing functions to encapsulate specific functionalities. The main components of the code include port scanning, UDP packet generation, user input handling, and display mechanisms. The code is organized to enhance readability and maintainability.

Socket Programming

The script utilizes the socket module in Python for network communication. Sockets are used to create UDP packets and establish connections for port scanning. The socket.AF_INET family is employed for Internet Protocol v4 addresses, and socket.SOCK_DGRAM is used for UDP communication.

Threading

Threading is employed to improve the efficiency of port scanning. Multiple threads are utilized to concurrently check a range of ports. This approach accelerates the scanning process by allowing parallel execution of port checks.

UDP Packet Generation

Random UDP packets of size 1490 bytes are generated using the `random._urandom` function. These packets serve as the payload for the DDoS attack. The size of the packets is chosen to optimize network traffic and resource utilization.

User Input Handling

User input is obtained using the `input` function, allowing users to specify the target IP address, timeout, port scan precision, and other parameters. Input validation is performed to ensure appropriate user-defined values.

Display Mechanisms

ISSN: 2456-5660 Volume 8, Issue 12 (Dec -2023)
www.jst.org.in *DOI:https://doi.org/10.46243/jst.2023.v8.i12.pp17-22*

The script employs display functions to present information to the user in a formatted manner. The `type` function simulates typewriter-style printing for a more interactive user experience. Additionally, various print statements are used to output information regarding the scan results and attack progress.

ALGORITHM

Initialization:
Initialize variables for delay, port scan precision, port scan interval, IP address, and other necessary values.

Setup Socket and Random Bytes:
Create a UDP socket using `socket. Generate random bytes for UDP packets using `random._urandom`.

Console Setup:
Customize console appearance using OS-specific commands (`os.system`) for Windows and other platforms.

Define Helper Functions:
Define functions like `cls()` to clear the console and `type()` to display text character by character.

User Input:
Prompt the user to input the target IP address for port scanning.

TCP Connection Function (`TCP_connect`):
Define a function to attempt a TCP connection to a specified port on the target IP address.Update the `output` dictionary with the result of the connection attempt.

Port Scanning Function (`scan_ports`):
Create threads to scan ports concurrently using the `TCP_connect` function. Start and join the threads to perform the port scanning.

Main Function (`main`):
Prompt the user for timeout, port scan precision, and port scan interval.Print information about the scanning process and phases. Call the `scan_ports` function to initiate port scanning.

Display Results:
Display the open ports found during the scan.

User Interaction:
Prompt the user to choose a port and specify package size.Continuously send UDP packets to the chosen port with the specified package size. Display the sent packets and their IDs.

IV RESULTS

Whenwerunthescanner, firstly in the terminal it displays the author's names and the undertaking of the port scanner. Then we have to enter the target IP of the system on which we want to perform the scanning. Now, we have to enter the time- out seconds as input. Timeout seconds is used to calculate the time interval of between each phasing module. Next step is to provide input for the port scan precision for each interval. We can adjust the port scan precision according to our requirement or the default recommended precision is three. Then, we have to enter scan interval of the ports. We can select the scan interval from 0 – 65535 ports. Then recommended is 5000 ports, if the timeout is 20 and precision is 3. Then, the scanner will scan for all

Journal of Science and Technology

ISSN: 2456-5660 Volume 8, Issue 12 (Dec -2023)

www.jst.org.in *DOI:https://doi.org/10.46243/jst.2023.v8.i12.pp17-22*

the open ports in the system within a displayed time interval in different phases of the scanning and finally displays the open ports.

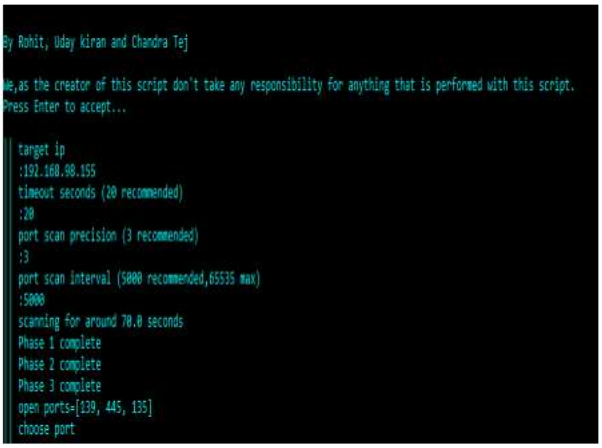
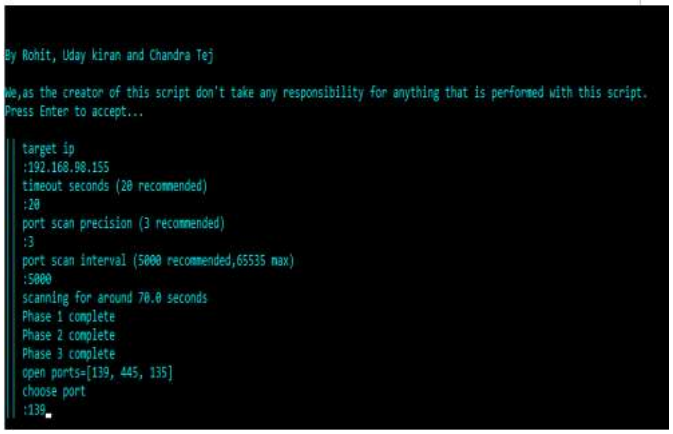
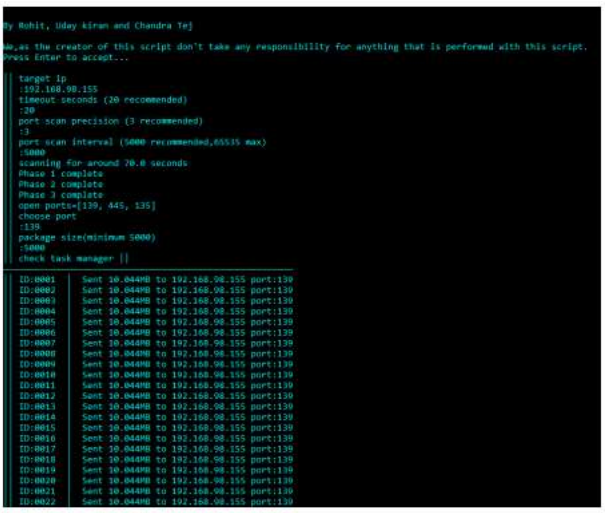


Fig4:Open ports present in the system

Now we can the port on which we want to perform the DoS attack. DoS attack is called as Denial of Service where it floods the target server with multiple packets in order to stall the server. We need to set the package size before launching the attack. The recommended size is 5000, we can set how much ever we want.



Finally, after performing the attack we can see that the port on which the attack was performed is compromised.



V CONCLUSION

The Open Port Scanner is an innovative endeavor aimed at providing a robust and user-friendly tool for network security assessment. It addresses this imperative by offering a comprehensive solution that empowers users to identify open ports, assess potential vulnerabilities, and fortify their networks against potential threats. The core objective is to deliver an open port scanner tool that conducts scans with high efficiency and accuracy. By employing advanced scanning techniques, multithreading, and smart reporting, we aim to provide users with reliable and actionable insights into their network's security posture. The interface enables users to easily configure scans, interpret results, and make informed decisions about network security measures. It Includes Versatile scan techniques including TCP SYN, UDP, and Connect scans to cater to various network environments and requirements. It has a repository of well-known ports and services for accurate identification and reporting of open ports. The Scanner gives a detailed reporting that summarizes scan results, provides insights into potential vulnerabilities, and aids in prioritizing security measures. Also ensures Logging and monitoring capabilities to track scan activities, troubleshoot issues, and maintain an audit trail for compliance. The script also prompts the user to choose a port and specify a package size. It then begins flooding the chosen port with UDP packets in an infinite loop. It provides instructions to check task manager or traffic based on theoperatingsystem, ensuring the user is aware of the
impact on system resources.

VI REFERENCES

1. Fyodor, "The Art of Port Scanning", Phrack Magazine, Volume 7, Issue 51, September 01 1997, Article 11 of 17.
2. Shaun Jamieson, "The Ethics and Legality of Port Scanning", October8,2001.http://www.sans.org/reading-room/whitepapers/legal/71.php.
3. Kocher, J.E.; Gilliam, D.P., "Self port scanning tool: providing a more secure computing environment through the use of proactive port scanning", 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, 13-15 June 2005.
4. Roger Christopher, "Port Scanning Techniques and the Defense Against Them", October 5, 2001. http://www.sans.org/reading-room/whitepapers/ auditing/70.php
5. Kimberly Graves, "Official Certified Ethical Hacker Review Guide", Wiley Publishing, pp.15-65, February 2007.
6. Fyodor, "Remote OS Detection using TCP/IP Fingerprinting(2ndGeneration)",Jan2007.
http://nmap.org/osdetect/index.html#id287961.