A Project Report On

**Netsecanner**

Submitted in partial fulfillment of the requirement for the award of the degree

MASTER OF SCIENCE

(Cyber Security & Cyber Law)

from

Marwadi University

Academic Year 2023 – 24

**Nisarg Jhatakia 92200565013**

**Sourabh Purohit 92200565012**

|  |
| --- |
| **Internal Guide** |
| Dr. Pankaj Mudholkar |



Marwadi University

Rajkot-Morbi Road, At &PO: Gauridad, Rajkot 360 003. Gujarat. India.



**Faculty of Computer Applications (FoCA)**

****

**This is to certify that the project work entitled**

**Netsecanner**

**submitted in partial fulfillment of the requirement for**

**the award of the degree of**

**Master of Science (CS & CL)**

**of the**

**Marwadi University**

**is a result of the bonafide work carried out by**

**Nisarg Jhatakia 92200565013**

**Sourabh Purohit 92200565012**

**during the academic year 2023 – 2024**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Faculty Guide** |  |  |  | **HOD** |

**External Viva**

|  |  |  |
| --- | --- | --- |
| **Name of the Examiners** |  | **Signature with Date** |

**DECLARATION**

We hereby declare that this project work entitled ‘Netsecanner' is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place:

Date:

Nisarg Jhatakia 92200565013 Signature: \_\_\_\_\_\_\_\_\_\_\_

Sourabh Purohit 92200565012 Signature: \_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **Dr.** **Pankaj Mudholkar,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his /her guidance, suggestions and expertise at every stage. A part from that his/her valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the Dean of our department **Dr. R. Sridaran** for giving us an opportunity to work over this project and for their end-less and great support. And to all other people who directly or indirectly supported and help us to fulfil our task.

Nisarg Jhatakia 92200565013 Signature: \_\_\_\_\_\_\_\_\_\_\_

Sourabh Purohit 92200565012 Signature: \_\_\_\_\_\_\_\_\_\_\_

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapters** | **Particulars** | **Page No.** |
| 1 | **Introduction**   * 1. Problem Definition   2. Core Components   3. Project Profile   4. Assumptions and Constraints | 1 - 10 |
| 2 | **Requirement Determination & Analysis**   * 1. Requirement Determination   2. Targeted Users   3. Details of tools and techniques used / implemented   4. Advantages and Limitations of the used security tools | 11 - 16 |
| 3 | **System Design**   * 1. Flow chart   2. Dataset Details | 17-18 |
| 4 | **Development**  4.1 Script details / Source code  4.2. Test reports | 19- 29 |
| 5 | **Proposed Enhancements** | 30 -31 |
| 6 | **Conclusion** | 32 |
| 7 | **Bibliography** | 33-34 |

**FIGURE INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Figure No.** | **Particulars** | **Page No.** |
| 1 | Figure 1.1 | Diagram | 12 |
| 2 | Figure 2.1 | Flow chart of my project | 17 |
| 3 | Figure 3.1 | Requirements installation | 26 |
| 4 | Figure 3.2 | Opening main.py | 26 |
| 5 | Figure 3.3 | Arp scan | 27 |
| 6 | Figure 3.4 | Arp scan output | 27 |
| 7 | Figure 3.5 | Ranged port scan | 28 |
| 8 | Figure 3.6 | Port scan output | 28 |
| 9 | Figure 3.7 | Ping scan | 29 |

**CHAPTER-1**

**Introduction**

In an era of advancing cyber threats and security breaches, the need for comprehensive network scanning tools is more critical than ever. Our project delves into this vital domain, offering a powerful suite of scanning techniques: from precise ping scans to in-depth ARP scans, and a versatile port scanner. With the ability to dissect networks and gather MAC addresses, our tool equips users to navigate the digital landscape with confidence and precision.

This indispensable tool finds practical application in a wide array of scenarios, from fortifying enterprise networks against potential intrusions to empowering individuals to safeguard their personal connections. It serves as an invaluable resource for IT professionals, security experts, and everyday users alike, providing crucial insights for effective network management and security in an increasingly interconnected world.

**Key Objectives:**

1. ***Network Discovery*:** Swiftly identifying active hosts through ping and ARP scans.
2. ***Port Scanning Proficiency*:** Conducting in-depth port scans for effective vulnerability assessment.
3. ***MAC Address Enumeration*:** Retrieving and displaying MAC addresses for precise network inventory.
4. ***Customizable Parameters*:** Allowing users to define tailored scan parameters, optimizing results for specific environments.
5. ***User-Friendly Interface*:** Designing an intuitive interface for accessibility, regardless of technical expertise.
6. ***Optimized Performance*:** Streamlining tool efficiency for handling extensive network scans without compromise.
7. ***Detailed Reporting and Logging*:** Generating organized reports and implementing comprehensive logging for analysis and auditing.
8. ***Security Integration*:** Incorporating secure scanning practices to mitigate potential disruptions and security breaches.
9. ***Comprehensive Documentation:*** Developing user-friendly guides and manuals for effective tool utilization.

**Significance:**

This project holds paramount importance in the realm of network management and security. By seamlessly combining ping, ARP, and port scans, it provides users with a versatile tool for efficient network reconnaissance. This empowers businesses and individuals to proactively identify vulnerabilities, fortify their defenses, and safeguard sensitive information. Additionally, the ability to retrieve MAC addresses enhances network inventory management. In an era dominated by cyber threats, this tool stands as a crucial asset in maintaining the integrity and security of digital infrastructures.

* 1. **Problem Definition**

In our modern digital age, network vulnerabilities are more prevalent than ever, exposing individuals and organizations to potential breaches and cyber threats. A reliable network scanning tool is crucial in identifying and addressing these vulnerabilities, providing a proactive defense against potential intrusions.

**Problem Statement:**

In network management and security, there is a pressing need for a comprehensive scanning tool capable of identifying active hosts, retrieving MAC addresses, and conducting in-depth port scans. The current landscape lacks a unified solution that efficiently combines these essential functions.

**Challenges:**

**Efficient Network Discovery:** Swiftly identifying active hosts through ping and ARP scans.

**Port Scanning Proficiency:** Conducting in-depth port scans for effective vulnerability assessment.

**MAC Address Enumeration:** Retrieving and displaying MAC addresses for precise network inventory.

**Objectives:**

**Customizable Parameters:** Allowing users to define tailored scan parameters, optimizing results for specific environments.

**User-Friendly Interface:** Designing an intuitive interface for accessibility, regardless of technical expertise.

**Optimized Performance:** Streamlining tool efficiency for handling extensive network scans without compromise.

**Detailed Reporting and Logging:** Generating organized reports and implementing comprehensive logging for analysis and auditing.

**Security Integration:** Incorporating secure scanning practices to mitigate potential disruptions and security breaches.

**Comprehensive Documentation:** Developing user-friendly guides and manuals for effective tool utilization.

**Scope:**

This project focuses on the development of a network scanning tool designed to efficiently identify active hosts, retrieve MAC addresses, and conduct thorough port scans. The tool aims to address the gaps in existing solutions by providing a unified, user-friendly interface for network reconnaissance.

**Outcomes:**

The project will yield a versatile network scanning tool that empowers users to proactively identify vulnerabilities, strengthen network security, and enhance overall network management. It will offer a streamlined, user-centric solution for network reconnaissance, contributing to a more secure and resilient digital landscape.

**1.2. Core Components**

A network scanning tool can be composed of several core components to perform tasks like ping scan, ARP scan, and port scan.

1. **User Interface (UI)**:
   * The UI provides a graphical or command-line interface for users to interact with the tool. It should allow users to input target IP ranges or hostnames, select scan types, and view scan results.
2. **Input Parser**:
   * This component handles the parsing of user-provided input. It validates the input, extracts relevant information (e.g., target IP ranges), and passes it on to the scanning engine.
3. **Scan Engine**:
   * The scan engine is the heart of the tool. It’s responsible for conducting the various types of scans (ping, ARP, and port scans). Each scan type may have its own specialized module within the engine.
4. **Ping Scan Module**:
   * This module performs a ping sweep of a range of IP addresses to determine which hosts are live on the network.
5. **ARP Scan Module**:
   * The ARP scan module uses ARP requests to discover hosts on the local network. It’s particularly useful for discovering hosts that may not respond to ping requests.
6. **Port Scan Module**:
   * This module is responsible for scanning for open ports on the target hosts. It can use various techniques like TCP SYN, UDP, or other scan methods to determine open ports.
7. **Result Aggregator**:
   * This component collects and aggregates the results from the different scan modules. It compiles the information into a format that can be presented to the user.
8. **Output Formatter**:
   * The output formatter takes the aggregated results and formats them for presentation to the user. This could include displaying results in a user-friendly format on the python Gui menu
9. **Error Handling and Logging**:
   * This component is responsible for handling errors that may occur during the scanning process. It mentions important events, errors, and warnings to help with troubleshooting.
10. **Concurrency Manager**:
    * This component manages the allocation of resources and coordinates the scanning threads or processes.
11. **Configuration Management**:
    * This component allows users to configure various options such as timeout values, scan intensity, and other parameters that affect the behaviour of the scanning tool.

**1.3. Project Profile**

**Project Profile: Netsecanner**

**Overview:**

Netsecanner is a versatile network scanning tool designed to facilitate efficient and comprehensive network reconnaissance. It empowers administrators and security professionals to analyse and understand the topology of a network, identify live hosts, and discover open ports. With its user-friendly interface and robust scanning capabilities, Netsecanner is a valuable asset for both network administrators and security experts.

**Objectives:**

The primary objectives of Netsecanner are:

1. **Network Discovery**: Identify all active hosts within a specified IP range.
2. **Service Enumeration**: Determine the open ports and services running on each live host.
3. **Vulnerability Assessment**: Provide a foundation for further vulnerability assessments by identifying potential entry points.
4. **Security Auditing**: Support security audits by highlighting potential weaknesses in the network.
5. **Performance Optimization**: Optimize scan times and resource utilization to provide fast and accurate results.

**Core Components:**

1. **UI & Input Parser:** Provides an intuitive interface for inputting target IP ranges, scan types, and viewing results. Validates and extracts relevant information.
2. **Scan Engine & Modules:** Conducts ping, ARP, and port scans. Specialized modules for each scan type.
3. **Result Handling & Formatting:** Aggregates and formats scan results for user-friendly presentation.
4. **Error Handling & Logging:** Manages errors, events, and warnings for troubleshooting.
5. **Concurrency & Resource Management:** Optimizes resource allocation and coordinates scanning processes.
6. **Configuration Management:** Allows customization of scan parameters (timeout values, intensity, etc.).

**Key Features:**

* Timeout Configuration
* Scan Rate Limiting
* Network Topology Mapping
* Bandwidth Usage Monitoring
* Custom Scripts and Extensions
* Stealth Mode Options
* Reporting and Visualization

**Target Audience:**

1. **Network Administrators:** To analyse and understand network topology for effective management.
2. **Security Professionals:** For vulnerability assessments and security audits.
3. **Penetration Testers:** As a foundational tool for identifying potential entry points.
4. **IT Professionals:** To optimize network performance and identify potential vulnerabilities.

**Benefits:**

1. **Efficient Reconnaissance:** Quickly identifies live hosts and open ports for network assessment.
2. **User-Friendly Interface:** Intuitive UI simplifies the scanning process, even for less experienced users.
3. **Customization:** Allows users to adjust settings for specific scanning requirements.
4. **Optimized Resource Usage:** Efficiently allocates resources, minimizing scan times.
5. **Enhanced Security Measures:** Identifies potential vulnerabilities, allowing for proactive security measures.
6. **Faster Results:** Supports concurrent scanning for swift analysis of multiple hosts.
7. **Troubleshooting Support:** Logs events and errors for effective problem-solving during scanning.

**1.4 Assumptions and Constraints**

**Assumptions:**

1. **Stable Network Conditions:** The tool assumes a relatively stable network environment without frequent disruptions or high levels of packet loss.
2. **Accessible Target Networks:** The tool assumes that the networks being scanned are accessible and within reach of the scanning device.
3. **Permission and Authorization:** It assumes that the user has proper authorization and permission to conduct network scans, and that legal and ethical considerations are adhered to.
4. **Availability of Multithreading Support:** Assumes that the target system has the capability to support multithreading for efficient execution of scans.
5. **Properly Configured Firewalls and IDS:** Assumes that firewalls and intrusion detection systems (IDS) are properly configured and won’t excessively block or disrupt the scanning process.
6. **Available System Resources:** Assumes that the host system has sufficient resources (CPU, memory, etc.) to support the concurrent execution of multiple scanning threads.
7. **Accurate MAC Address Resolution:** Assumes that the Address Resolution Protocol (ARP) is functioning correctly and can accurately map IP addresses to MAC addresses within the local network segment.
8. **Target Hosts Are Online:** Assumes that the target hosts are powered on and connected to the network during the scan.

**Constraints:**

1. **Hardware Limitations:**

The tool’s performance may be constrained by the hardware capabilities of the host system, such as CPU speed and available memory.

1. **Network Bandwidth:**

The scanning process may be limited by the available network bandwidth, potentially affecting scan speed and accuracy.

1. **Firewall and Security Policies:**

The tool may encounter limitations imposed by firewalls, security policies, or IDS systems, potentially leading to incomplete or inaccurate scan results.

1. **Network Segment Boundaries:**

ARP scans are limited to the local network segment and may not be able to discover hosts on different subnets without proper routing and permissions.

1. **Legal and Compliance Considerations:**

The tool’s usage may be subject to legal and compliance constraints, and users must ensure they have proper authorization and adhere to applicable laws and regulations.

1. **Potential False Positives/Negatives:**

Due to the dynamic nature of networks, there may be cases of false positives (indicating a port is open when it’s not) or false negatives (not detecting a live host or open port).

1. **Dependence on Network Responses:**

The tool’s accuracy is contingent on the responsiveness of the network and the hosts being scanned. Network congestion or unresponsive hosts may affect results.

1. **Limited Operating System Compatibility:**

The tool’s compatibility may be limited to specific operating systems, potentially excluding some environments.

**CHAPTER-2**

**Requirement Determination & Analysis**

**2.1. Requirement Determination**

**Programming Language and Frameworks:**

* **Python 3.x:** The tool will primarily be developed in Python 3.x due to its versatility, extensive libraries, and compatibility with both Windows and Unix-based systems.
* **Bash Scripting:** Bash will be utilized for executing shell commands and automating certain aspects of the tool, enhancing compatibility with Unix-based systems.
* **Multithreading Libraries (e.g., threading in Python):** Multithreading support is essential for concurrent execution of scans, ensuring efficient resource utilization.

**Operating Systems:**

* **Linux (e.g., Ubuntu, CentOS):** The tool supports various Linux distributions, including Ubuntu and CentOS, to accommodate users in enterprise and server environments.

**Timeout Handling:**

Incorporate timeout mechanisms for network operations to prevent scans from waiting indefinitely for responses.

**Unit Testing and Quality Assurance:**

Robust suite of unit tests to verify the correctness and reliability of the tool’s functionality.

**Documentation and User Guides:**

Comprehensive documentation and user guides to facilitate ease of use, installation, and troubleshooting.

**2.2 Targeted Users**

1. **Network Administrators:** To analyse and understand network topology for effective management.
2. **Security Professionals:** For vulnerability assessments and security audits.
3. **Penetration Testers:** As a foundational tool for identifying potential entry points.
4. **IT Professionals:** To optimize network performance and identify potential vulnerabilities.

With the increasing amount of data breaches small and mid-sized enterprises (SMEs) have spent the second highest on securing their network security

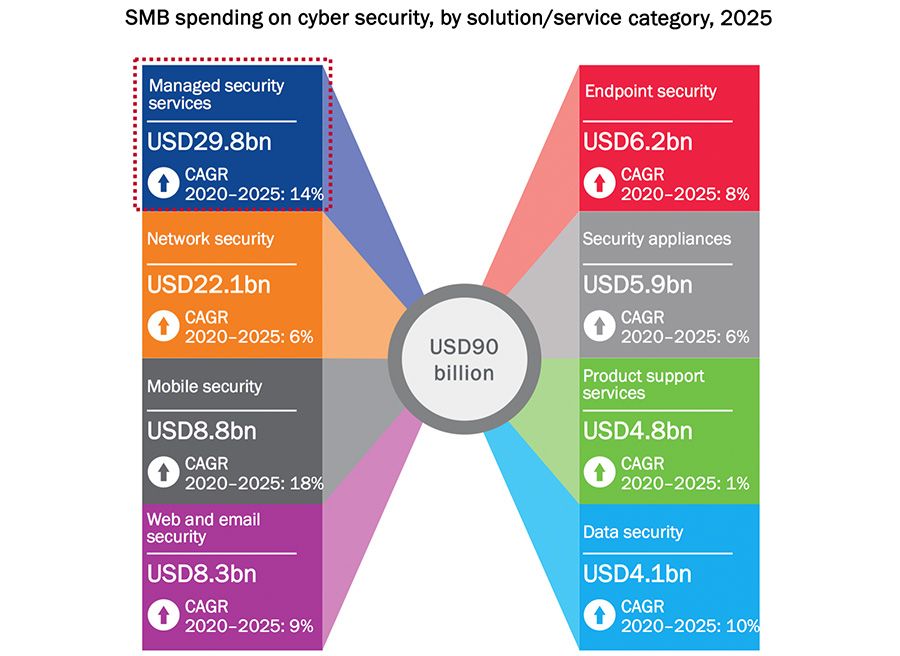


Figure 1.1 Diagram

**2.3 Details of Tools and Techniques Used / Implemented**

1. **Development Environment:**
   * **Visual Studio Code (VSCode):** The primary integrated development environment (IDE) used for writing and managing the Python and Bash code. VSCode provides a user-friendly interface, extensive extension support, and version control integration.
2. **Programming Languages:**
   * **Python 3.x:** Python serves as the core programming language for developing the Multithreaded Network Scanning Tool. Its versatility, rich standard library, and support for multithreading make it an ideal choice for this project.
   * **Bash Scripting:** Bash is employed for executing shell commands and automating certain aspects of the tool, enhancing compatibility with Unix-based systems.
3. **Version Control:**
   * **Git and GitHub:** Git is used for version control, allowing for collaborative development, code management, and tracking of changes. GitHub serves as the remote repository for the project.
4. **Multithreading Techniques:**
   * **Threading Module in Python:** The threading module is utilized in Python to implement multithreading support. This allows for concurrent execution of scans, optimizing resource utilization and enhancing scanning speed.
5. **Socket Programming:**
   * **Python’s Socket Module:** Socket programming techniques are implemented using Python’s socket module. This facilitates the establishment of network connections for communication with hosts during the scanning process.
6. **Port Scanning Algorithms:**
   * **SYN, UDP, and TCP Connect Scans:** Various port scanning algorithms are implemented to accurately identify open ports on target hosts. This includes SYN scans, UDP scans, and TCP connect scans.
7. **Ping Scanning Techniques:**
   * **ICMP Echo and TCP Ping Scans:** Different ping scanning techniques are employed, such as ICMP Echo (ping) and TCP Ping, to detect live hosts on the network.
8. **ARP Cache Interaction:**
   * The tool interacts with the ARP cache to facilitate ARP scans. This allows for the discovery of devices on the local network segment.
9. **Concurrency Management:**
   * Techniques such as thread synchronization and coordination are implemented to efficiently manage concurrency, ensuring optimal use of system resources during the scanning process.
10. **Timeout Handling:**
    * Timeout mechanisms are incorporated to manage network operations, preventing scans from waiting indefinitely for responses and ensuring timely progress.
11. **Error Handling and Logging:**
    * Comprehensive error handling routines and logging mechanisms are implemented to capture and report any unexpected events or errors that may occur during scans.

**2.4 Advantages and Limitations of the Used Security Tools**

**Advantages:**

1. **Efficient Multithreading:**

The tool’s multithreading implementation allows for concurrent execution of scans, which significantly improves scanning speed and efficiency.

1. **Utilization of Standard Libraries:**

Python’s extensive standard library provides a rich set of modules for network programming, enabling robust and feature-rich functionality.

1. **Ease of Scripting and Automation:**

Python’s scripting capabilities enable the automation of network scanning tasks, streamlining the scanning process and enhancing overall efficiency.

1. **Community and Ecosystem:**

Python has a large and active community, ensuring a wealth of resources, libraries, and support for security practitioners.

**Limitations and Considerations:**

1. **Dependency on Python Installation:**

Users must have Python installed on their system for the tool to run. Additionally, the tool may require specific Python packages or modules, which need to be installed or included in the environment.

1. **Network Configuration and Connectivity:**

The tool relies on proper network configuration, including the availability of connected devices and their IP addresses. Any misconfiguration or disconnected devices may lead to incomplete or inaccurate scan results.

1. **Resource Utilization:**

The tool’s multithreading approach can be resource-intensive, particularly on systems with limited CPU or memory resources. Users need to ensure their system can support the level of concurrency selected.

1. **False Positives/Negatives:**

Due to the dynamic nature of networks, there may be cases of false positives (indicating a port is open when it’s not) or false negatives (not detecting a live host or open port).

1. **Limited to Local Network Segment:**

ARP scans are limited to the local network segment and may not discover hosts on different subnets without proper routing and permissions.

1. **Potential Impact on Network Performance:**

Intensive scanning activities may generate additional network traffic and could potentially impact the performance of the target network. This should be considered, especially in production environments.

1. **Security Considerations:**

Users must ensure that they have proper authorization and adhere to legal and ethical guidelines when conducting network scans, to avoid any potential security or privacy breaches.

**CHAPTER-3**

**Project Design**

**3.1. Flow chart**

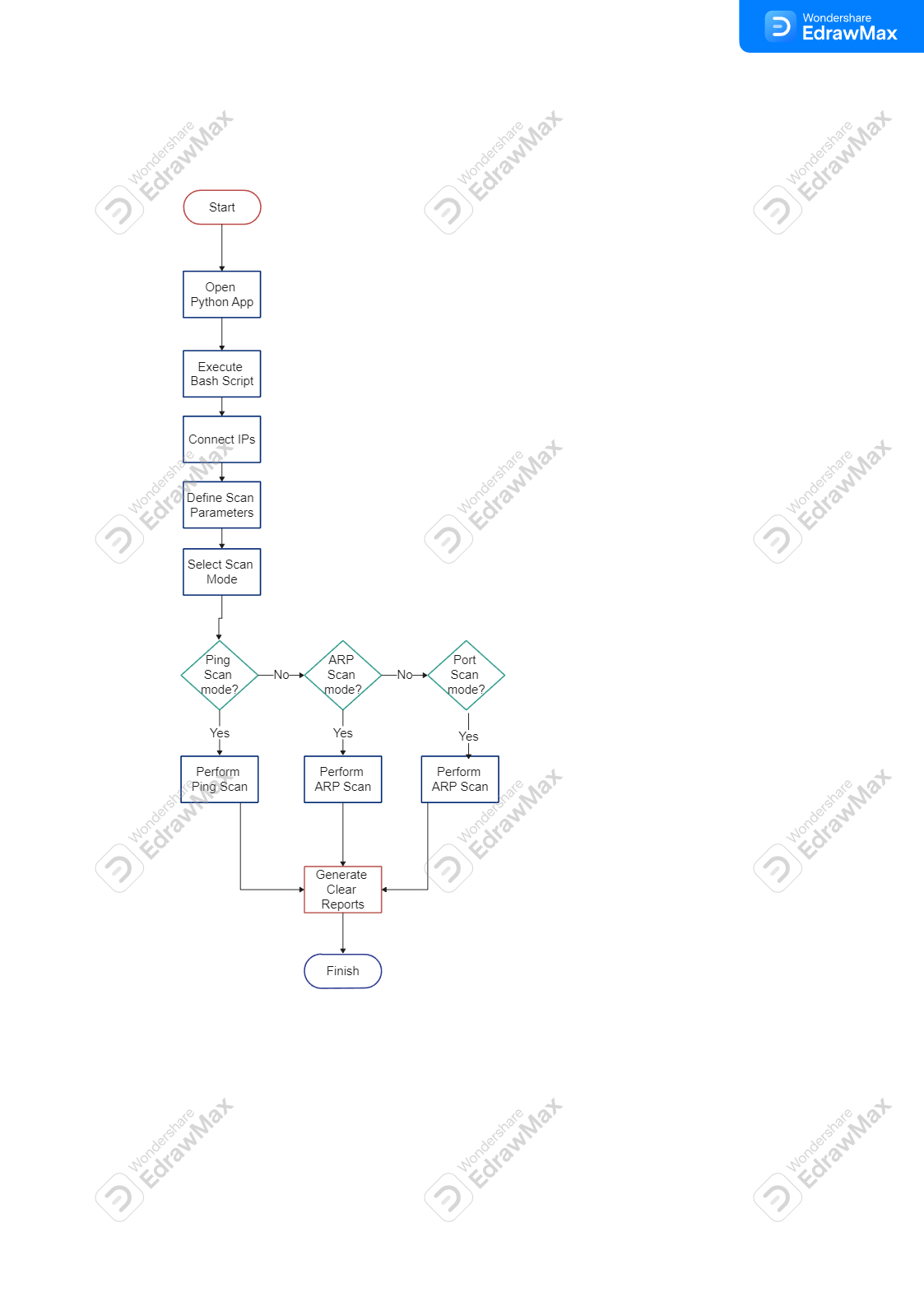
****

Figure 2.1 Flow chart of my project

**3.2 Dataset details**

**Dataset Details:**

**Source:** The data is retrieved directly from the Command-Line Interface (CLI) in real-time, capturing live network information.

**Size:** The dataset is of manageable size, as it requires the installation of Python and a few plugins. This ensures efficient handling and processing.

**Format:** The data is temporarily stored as output displayed in a Tkinter interface. This temporary storage is cleared upon the initiation of the next functionality in the application.

**Attributes:** The dataset contains information on both single and multiple IP addresses, MAC addresses, and open ports. This detailed information forms the core of the network scanning tool's capabilities.

**Variability:** The tool demonstrates versatility by collecting data on both Wi-Fi and Ethernet connections. It is designed to work seamlessly in various network environments, whether in an office or a home network, adapting to the specified IP range or port range.

**Anomalies:** While the dataset does not contain specific anomalies, it does provide feedback if there are input errors, such as mistyped IP addresses or range entries. This feature ensures accurate data collection.

**Privacy Concerns:** The dataset is processed locally on the user's computer. As the tool is not designed for server or cloud usage, there are no privacy concerns associated with the data.

**Real-time Handling:** The tool operates in real-time, providing immediate feedback and data updates as the network scan progresses.

**Accessibility:** There are no restrictions on the port numbers or the range of IP addresses that can be entered, providing users with flexibility and customization options.

**CHAPTER-4**

**Development**

**4.1 Script Details / Source Code**

**4.1.1 main.py -User Interface Script**

* **Description:** This script provides a graphical user interface (GUI) for interacting with the network scanning tools.
* **Dependencies:**
  + tkinter: Python library for GUI.
  + ttkthemes: Python library for themed Tkinter windows.
  + subprocess: Used to run command-line scripts.
  + threading: Used to run commands in the background.
  + simpledialog: Provides dialogs for user input.

**4.1.2. portscan.py – Port Scanning Script**

* **Description:** A Python script for multi-threaded port scanning to identify open ports on a target.
* **Dependencies:**
  + socket: Used for network communication and port scanning.
  + argparse: Enables command-line argument parsing for specifying the target and port(s).

**4.1.3 pingscan.py – Ping Scanning Script**

* **Description:** A Python script for discovering live hosts in a network using ICMP ping requests.
* **Dependencies:**
  + argparse: Enables command-line argument parsing for specifying the target (IP range, IP list, CIDR notation, or single IP).
  + re: Regular expressions for parsing target input.
  + ipaddress: Used to parse IP ranges in CIDR notation.
  + threading: Used for concurrent ping requests.
  + subprocess: Invoked for executing ping commands.
  + platform: Used to determine the system platform (Linux or Windows).

**4.1.4 getmac – Bash Script**

* **Description:** A Bash script for obtaining MAC addresses of devices within a local network.
* **Dependencies:**
  + getipeth and getipwlan: Python scripts for fetching IP addresses of the Ethernet and Wi-Fi interfaces.
* **Input Parameters:**
  + iface: User specifies the network interface (Ethernet or Wi-Fi) for MAC address retrieval.

**4.1.5 getipeth and getipwlan – Python Scripts**

* **Description:** These Python scripts fetch the IP addresses of the Ethernet (eth0) and Wi-Fi (wlan0) interfaces, respectively.
* **Dependencies:**
  + netifaces: Python library used to retrieve network interface information.

These scripts provide necessary IP address information for the getmac script to function effectively.

Please include this consolidated script details section in your project documentation or README file to provide comprehensive information about the included scripts and how to interact with the user interface provided by main.py.

In addition to the script details provided in the previous section, this section includes further information about how to use the network scanning tools and the user interface. It also highlights important considerations for using these tools effectively.

**4.1.6 Using the User Interface (main.py)**

To run the graphical user interface (GUI) provided by main.py:

1. Ensure you have the required Python libraries installed. You can use pip to install missing libraries:

 Run the main.py script using Python:

1. The GUI window will appear, presenting the following options:
   * **ARP Scan (ARP MAC Address Discovery):** Click the “ARP Scan” button to initiate the network scanning process. You will be prompted to select an interface (Ethernet or Wi-Fi) for scanning.
   * **Port Scan (Port Scanning):** Click the “Port Scan” button to scan a specific IP address or hostname for open ports within a specified range. You will be prompted to provide the target and port range.
   * **Ping Scan (Host Discovery):** Click the “Ping Scan” button to discover live hosts in a network. You can input an IP range, a list of IP addresses, CIDR notation, or a single IP address.
2. Follow the on-screen prompts and instructions to perform the desired network scan.

**4.1.7 Important Notes and Considerations**

* **ARP Scan Limitations:** The ARP scan (getmac) script is designed for /24 subnets (common home network sizes). Using it in different subnet configurations may lead to errors or incorrect output.
* **Interface Selection:** When running the ARP scan, you’ll need to select an interface (Ethernet or Wi-Fi) that is currently up and operational. If both interfaces are down, the script will terminate with an error message.
* **Port Scan and Ping Scan:** The portscan.py and pingscan.py scripts offer command-line functionality. You can use these scripts directly from the command line as well, providing your own input and target parameters.
* **Multithreading:** Both the port scan and ping scan scripts are multithreaded for increased efficiency. The number of threads is configurable and can be adjusted in the script.
* **Error Handling:** Ensure you handle errors gracefully, especially if providing custom inputs to the command-line scripts. Invalid input may result in unexpected behavior.

**4.1.8 Output and Results**

The main.py GUI will display the results of the network scans in real-time within the GUI window. The results may include MAC addresses from the ARP scan, open ports from the port scan, and live hosts from the ping scan.

**Code Architecture Overview**

**Main Script (main.py):**

1. **Imports and Dependencies:**
   * The script imports modules from the tkinter library to create a graphical user interface (GUI).
   * It utilizes the ttkthemes library for theming the GUI.
   * The subprocess, threading, and simpledialog modules are used for executing shell commands, handling threads, and collecting user input.
2. **Interface Selection and Command Execution:**
   * The script provides a graphical interface for the user to select an interface (Ethernet or WiFi) for ARP scanning.
   * Upon interface selection, it constructs a relevant command (e.g., "bash getmac eth" or "bash getmac wifi").
   * It initiates command execution in a separate thread to prevent freezing the GUI.
   * Real-time output from the executed command is displayed in the GUI.
3. **User Dialog for Interface Selection:**
   * A user-friendly dialog is created to prompt the user to select an interface (Ethernet or WiFi) for ARP scanning.
   * Upon interface selection, the dialog is closed after a short delay.
4. **Additional Network Scanning Options:**
   * The script provides options for port scanning and ping scanning.
   * Buttons for these operations invoke separate functions when clicked: open\_port\_scan\_dialog and open\_ping\_scan\_dialog.
5. **Display Elements:**
   * The script creates a GUI window with labels, buttons, and a text widget for displaying the results.
   * It applies a custom style for buttons with specific fonts, colors, and background styles.
   * The text widget allows real-time output to be displayed to the user as the commands execute.
6. **Main Execution:**
   * The main function serves as the entry point for the script.
   * It checks command-line arguments (though not used extensively in the script) and initializes the GUI.
   * The main GUI loop (root.mainloop()) runs to handle user interactions and execute network scanning commands.

**Port Scan (portscan.py):**

1. **Imports and Dependencies:**
   * The script uses the socket, threading, and argparse modules for port scanning.
   * It employs multithreading to improve scanning speed and utilizes a set to store open ports.
2. **Port Scanning Functions:**
   * The script defines probe\_port to probe a single port on a target IP.
   * The scan\_ports function scans a list of ports and updates the set of open ports.
   * It splits the list of ports among multiple threads for parallel scanning.
3. **Argument Parsing:**
   * The script uses argparse to parse command-line arguments, including the target IP address or hostname and the port range to scan.
4. **Main Function:**
   * The main function serves as the entry point for the script.
   * It parses command-line arguments, initiates the port scanning process, and reports the open ports.

**ARP Scan (getmac):**

1. **Imports and Dependencies:**
   * The script is primarily a Bash script with embedded Python code for IP address retrieval.
   * It relies on netifaces to fetch IP addresses of Ethernet and Wi-Fi interfaces.
2. **Interface and IP Address Detection:**
   * The script checks the availability of Ethernet and Wi-Fi interfaces using the getipeth and getipwlan Python scripts.
   * It validates the chosen interface (Ethernet or Wi-Fi) and retrieves the corresponding IP address.
3. **Neighborhood Scanning:**
   * The script clears the ARP cache using the ip command to ensure accurate results.
   * It performs a ping sweep across the local network to discover active hosts.
   * The results are stored in the ~/.macoutput.txt file, and the script displays the MAC addresses and the number of alive hosts.

**Ping Scan (pingscan.py):**

1. **Imports and Dependencies:**
   * The script relies on argparse, re, ipaddress, and threading for IP range parsing and parallel ping scans.
   * It also uses the subprocess module to execute the ping commands.
2. **Ping Host Function:**
   * The ping\_host function pings a single IP address using the appropriate command based on the operating system (Linux or Windows).
   * It collects results and updates a list of alive hosts.
3. **IP Range Parsing:**
   * The script can handle various input formats, including single IPs, IP ranges, CIDR notation, and comma-separated lists.
   * It uses regular expressions to parse and validate the input.
4. **Parallel Ping Scanning:**
   * The script leverages threading to scan multiple IP addresses in parallel, improving efficiency.
   * It reports progress during the scan and ultimately displays the list of alive hosts.

**Overall:** The code architecture for all three scripts follows a clear pattern of imports, function definitions, argument parsing, and main execution. Each script is focused on its specific task: port scanning, ARP scanning, or ping scanning. They are well-structured, modular, and use external libraries and tools where necessary to achieve their respective goals. Additionally, they offer flexibility in terms of input formats and provide valuable results for network analysis.

**4.2. Test reports / POC (proof of concept)**

**Installing the requirements**

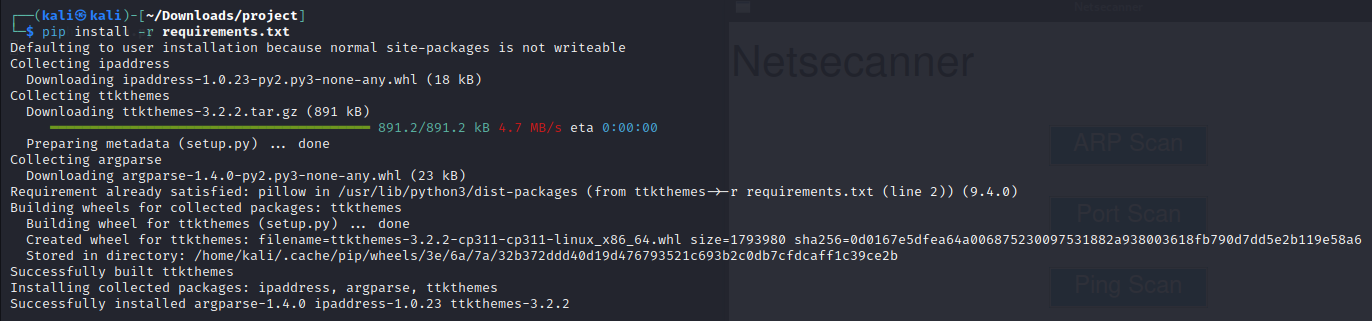
****

Figure 3.1 requirements installation

**To execute the program simply open ‘main.py’**

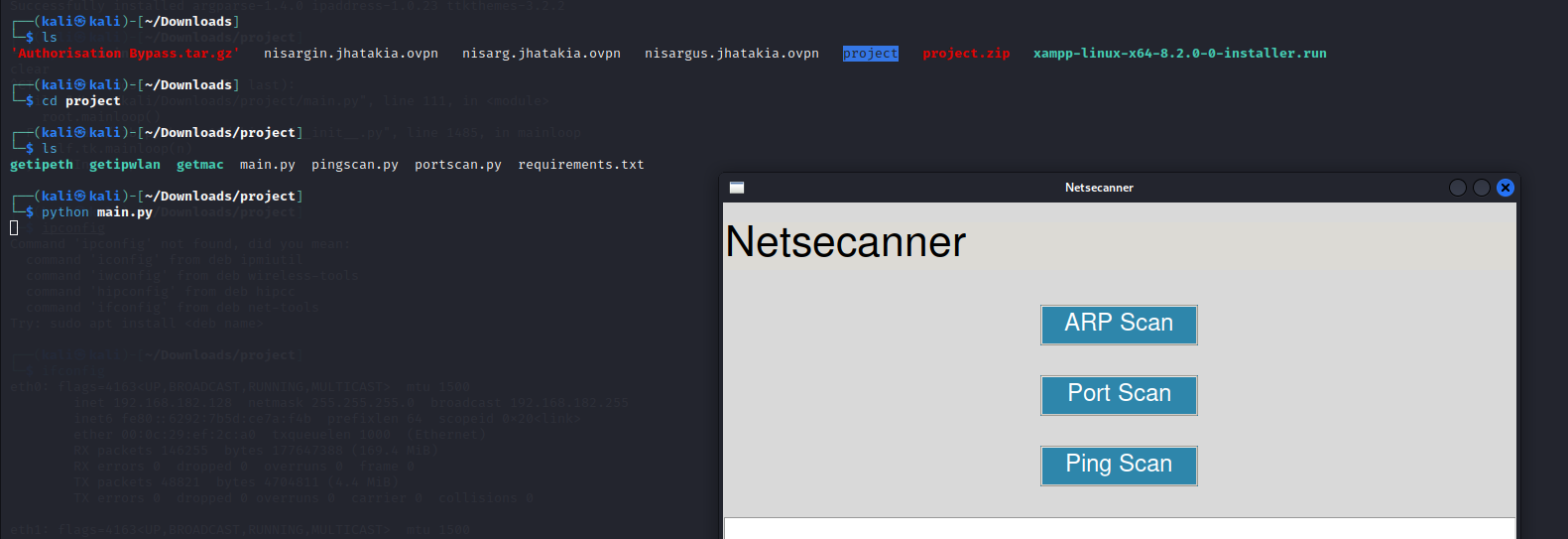
****

Figure 3.2 opening main.py

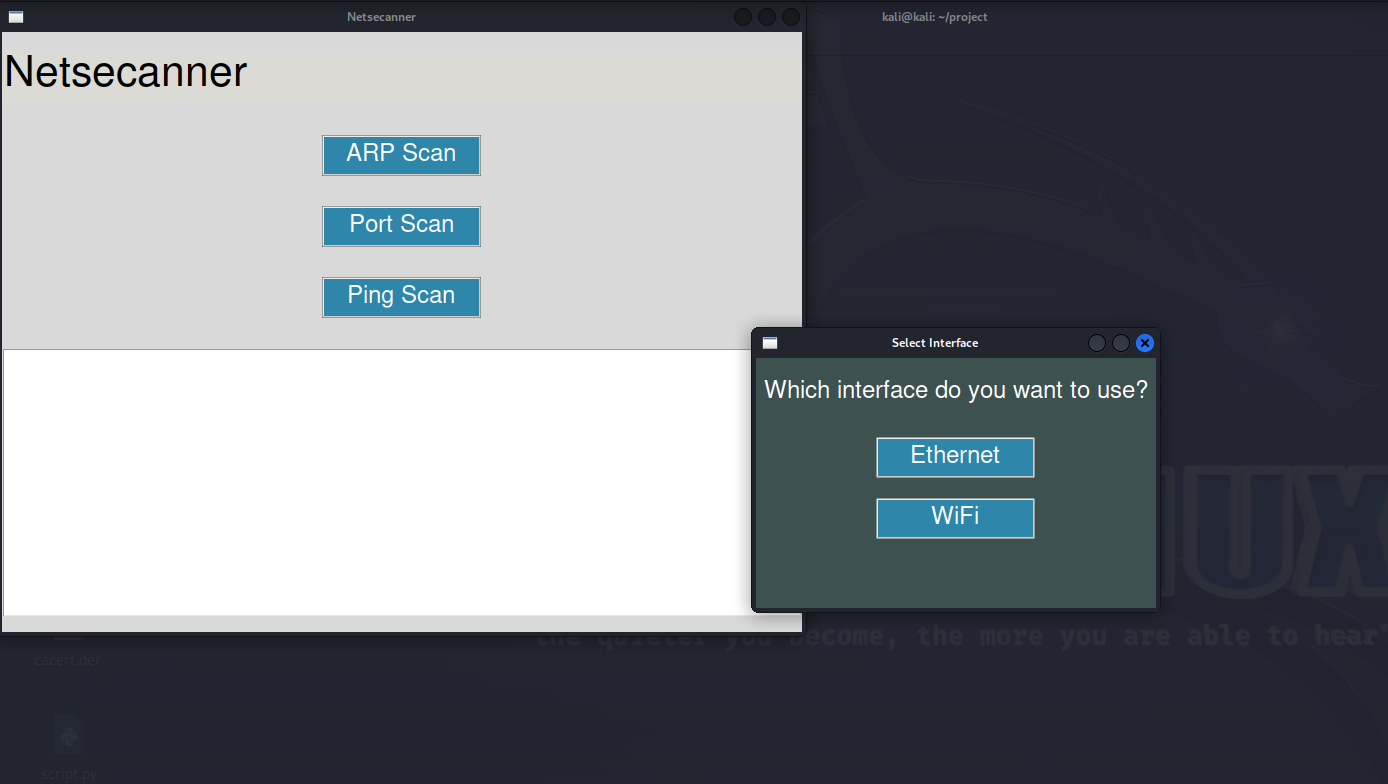
**Click on ARP Scan button and it will ask you to select the interface and it will perform ARP scan on the local network and give IP addresses and MAC addresses as output:  
  
**

Figure 3.3 arp scan

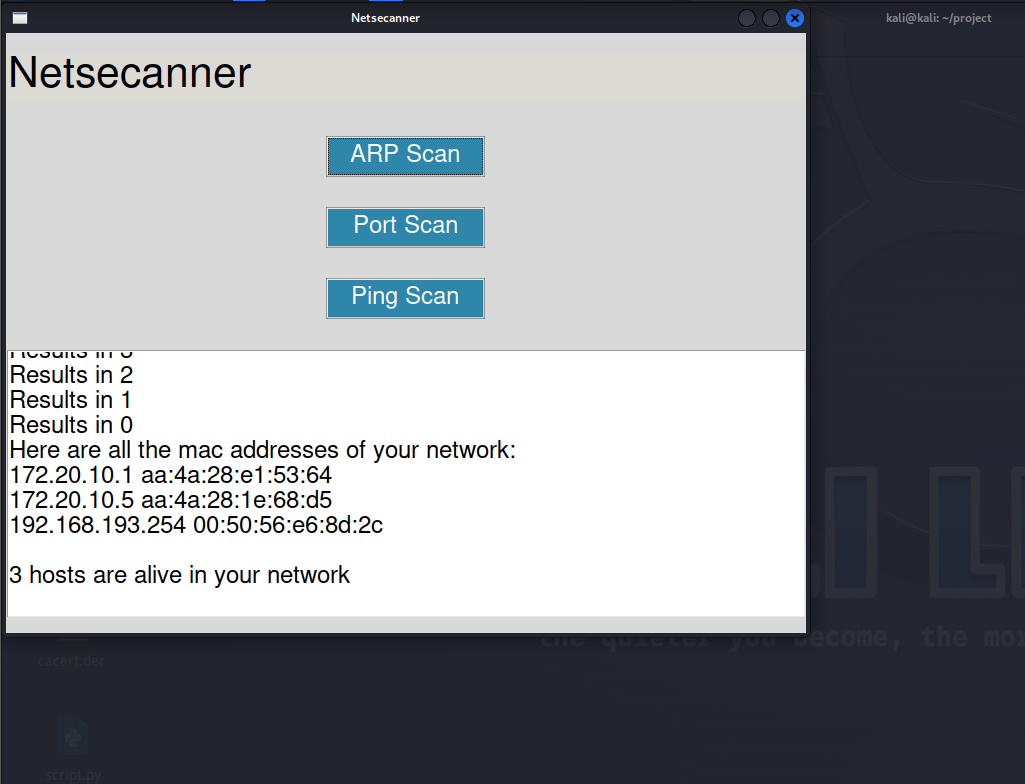
****

Figure 3.4 arp scan output

**Click on Port scan and you will be asked IP address and port range separated by space then it will scan the target IP and give you the open ports as output:**

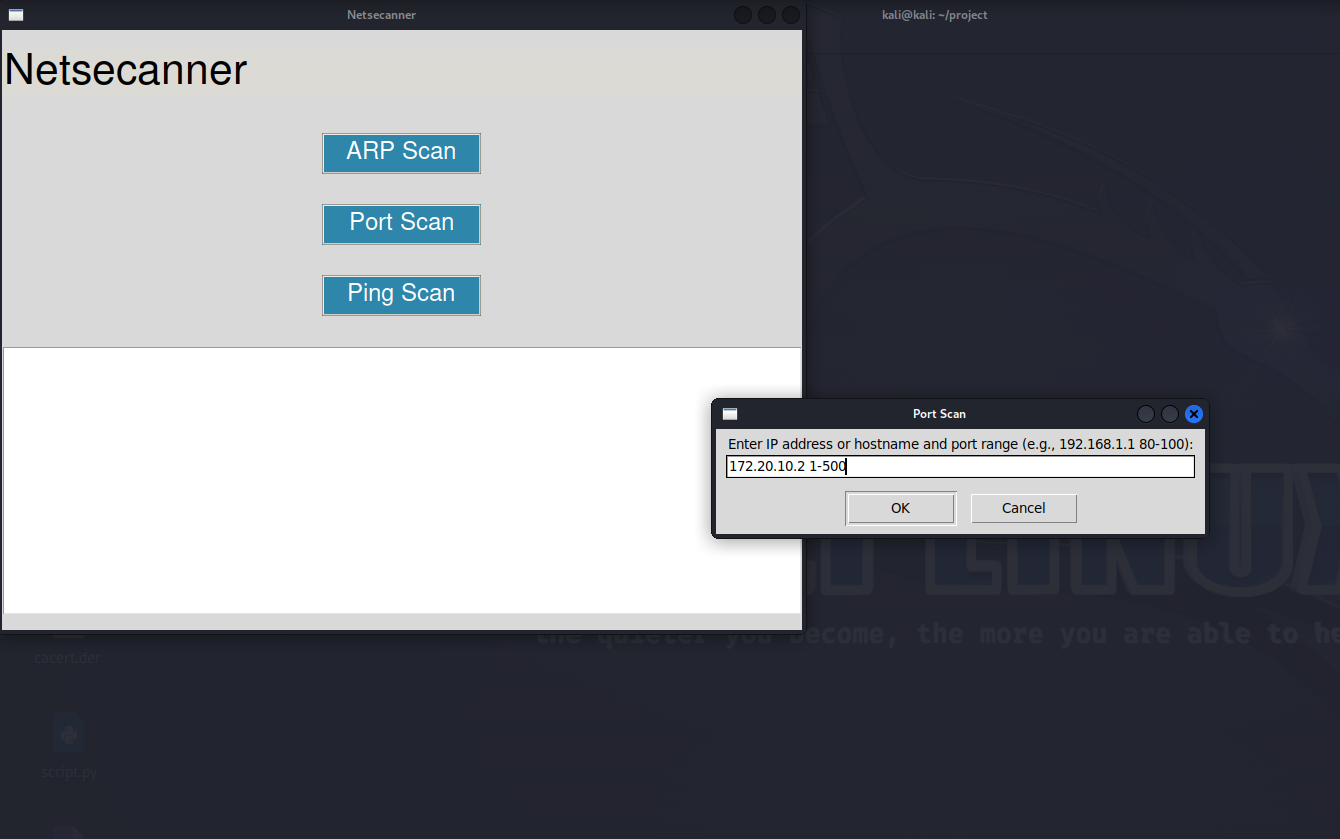
****

Figure 3.5 ranged port scan

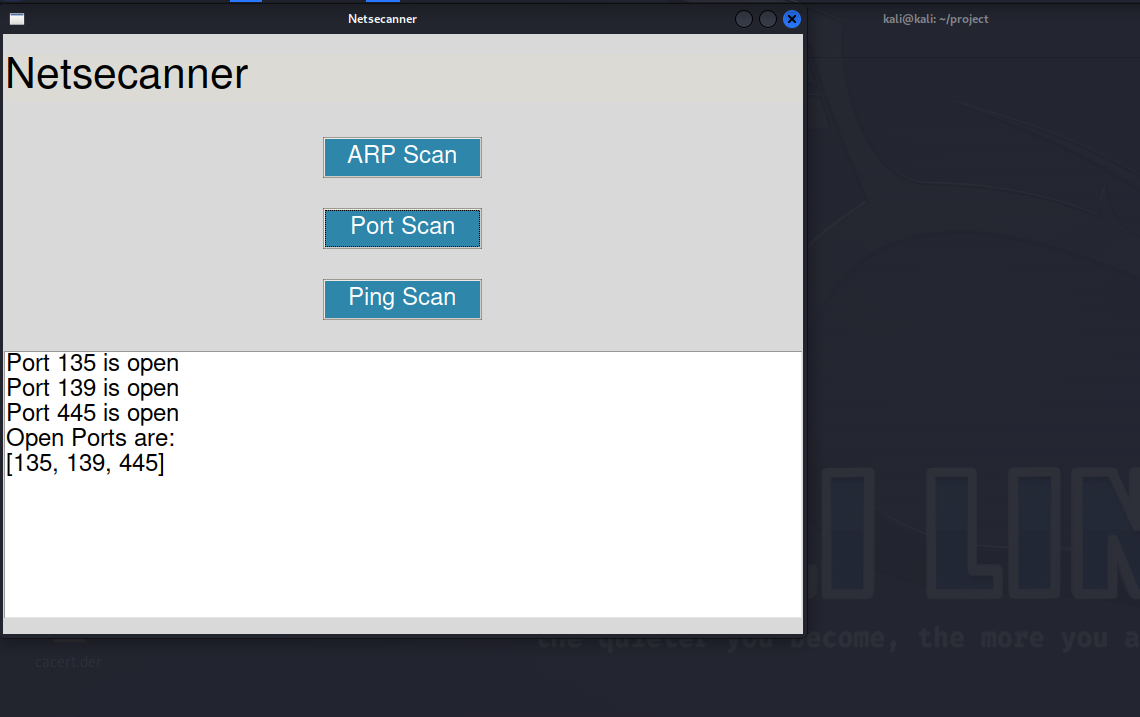
****

Figure 3.6 port scan output

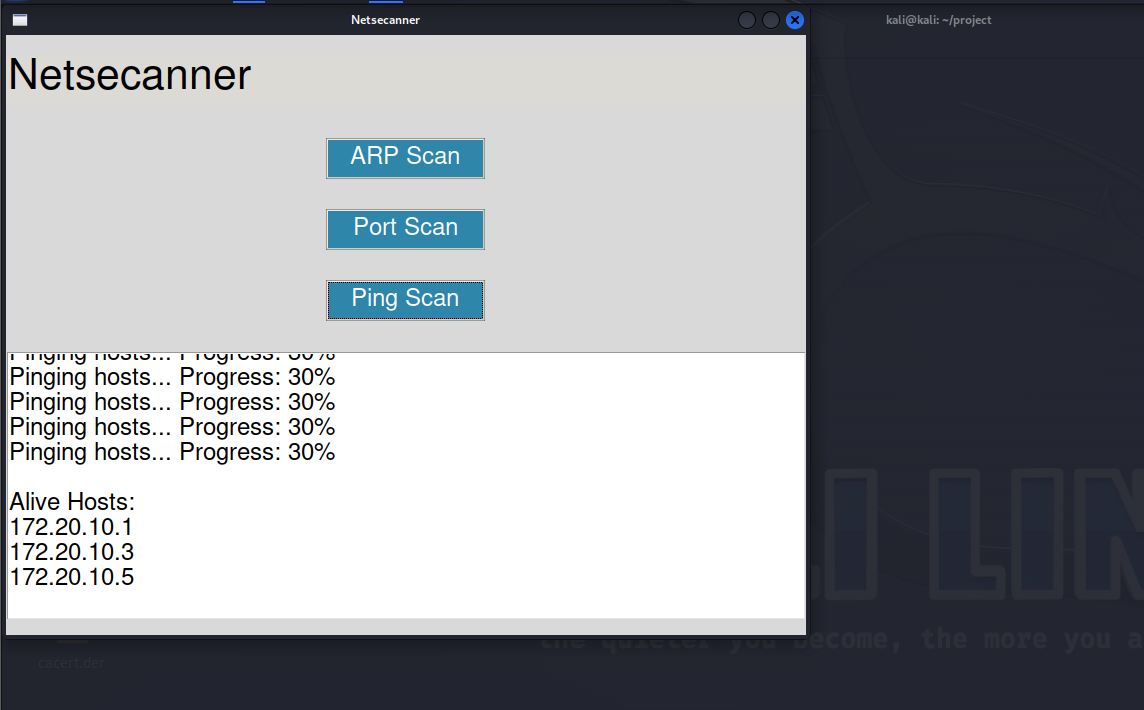
**Click on Ping scan and then write a range of IP addresses and then you will see alive hosts in the network:  
  
**

Figure 3.7 ping scan

**CHAPTER-5**

**Proposed Enhancements**

1. **Cross-Platform Support:**
   * Ensure seamless execution of the networking script on various operating systems (Windows, Linux, macOS) to cater to a broader user base.
2. **Intensive Network Scan Options:**
   * Implement advanced scanning techniques like SYN, FIN, or XMAS scans to provide more detailed information about the target hosts.
3. **Bandwidth Throttling:**
   * Introduce an option to limit the bandwidth used during scans to prevent overwhelming networks, ensuring responsible and controlled scanning.
4. **Customizable Output Formats:**
   * Allow users to specify the format in which scan results are presented, such as JSON, XML, or plain text, for compatibility with different analysis tools.
5. **Scan, Resume and Pause:**
   * Provide the ability to pause and resume scans, allowing users to manage long-running scans and pick up where they left off.
6. **Network Range Expansion:**
   * Allow users to input larger IP ranges or CIDR notations for broader network coverage, accommodating larger and more complex network environments.
7. **Configurable Timeout Values:**
   * Enable users to customize timeout values for different scan types, providing flexibility to adapt to varying network conditions.

**Note:**

These proposed enhancements are part of an ongoing effort to refine and expand our network scanning tool. The overarching project aims to create a versatile and efficient network security toolset that empowers users to conduct thorough assessments of their network environments. Our goal is to provide college students and aspiring network professionals with a robust tool that offers practical insights into network configurations and potential vulnerabilities.

The larger project aims to provide cross-platform support, enabling users to perform network scans on various operating systems, including Windows, Linux, and macOS. Additionally, it aims to provides more intensive scan options, allowing for in-depth analysis of target hosts. Committed to ensuring that this tool remains user-friendly, with clear documentation and interactive prompts for beginners. By incorporating these enhancements, the aim is to create a valuable resource for students to gain hands-on experience in network scanning and security assessment.

**CHAPTER-6**

**Conclusion**

In an era dominated by digital connectivity, network security has become paramount. The focus on enhancing this network scanning tool is a direct response to the escalating need for robust cybersecurity measures. By providing cross-platform support and intensified scanning options, the tool equips users, particularly college students, with an effective means of understanding and assessing network environments.

The commitment to user-friendliness, coupled with comprehensive documentation, ensures accessibility for individuals at all skill levels. Through these enhancements, the aim is to foster a hands-on learning experience in network security, empowering the next generation of cybersecurity professionals.

As technology evolves, so do cyber threats. The continuous refinement of network scanning techniques aligns with the ever-changing landscape of network security. It serves as a foundation for future innovations and developments in network security tools, reinforcing our collective defence against cyber threats.

Ultimately, this project seeks to fortify the digital realm by arming individuals, students, and professionals with the knowledge and tools they need to navigate and secure today's interconnected world. Through these efforts, there is an endeavour to create a safer and more resilient digital ecosystem for all.

**CHAPTER-7**

**Bibliography**

1. **"Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning"** by Gordon Fyodor Lyon
   * This book provides in-depth information on Nmap, a powerful open-source network scanning tool used for a wide range of network discovery and security auditing tasks.
2. **"Hacking: The Art of Exploitation"** by Jon Erickson
   * This book covers various aspects of network scanning, including port scanning and vulnerability assessment. It's a valuable resource for understanding network security from an offensive perspective.
3. **"Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems"** by Chris Sanders
   * This book focuses on network analysis and packet capture, which is essential for understanding network behavior, troubleshooting, and scanning techniques.
4. **Nmap Official Documentation:**
   * [Nmap Official Website](https://nmap.org/book/)
   * The official Nmap documentation provides comprehensive information about network scanning techniques, Nmap usage, and network discovery.
5. **Wireshark Official Documentation:**
   * [Wireshark User's Guide](https://www.wireshark.org/docs/wsug_html_chunked/)
   * Wireshark is a powerful network packet analyzer. Its official documentation covers the use of Wireshark for network analysis and scanning.
6. **Cybrary (Online Training):**
   * [Cybrary Network Scanning Course](https://www.cybrary.it/course/network-scanning/)
   * Cybrary offers various online courses related to network scanning and ethical hacking. It's a valuable resource for learning network scanning techniques.
7. **Consultancy UK, SMEs to spend $90 billion on cyber security in 2025.**

<https://www.consultancy.uk/news/28470/smes-to-spend-90-billion-on-cyber-security-in-2025>

1. **Python.org, tkinter — Python interface to Tcl/Tk.**

<https://docs.python.org/3/library/tkinter.html>

1. **Cisco, DHCP: Dynamic Host Configuration Protocol**

<https://www.cisco.com/c/en/us/support/docs/ip/dynamic-address-allocation-resolution/13718-5.html>

1. **LearnPython.com, Using a Requirements File in Python.**

<https://learnpython.com/blog/python-requirements-file/>