AUTHOR
MEZZOUR OMAR

PROFESSOR
AIT TCHAKOUCHT TAHA

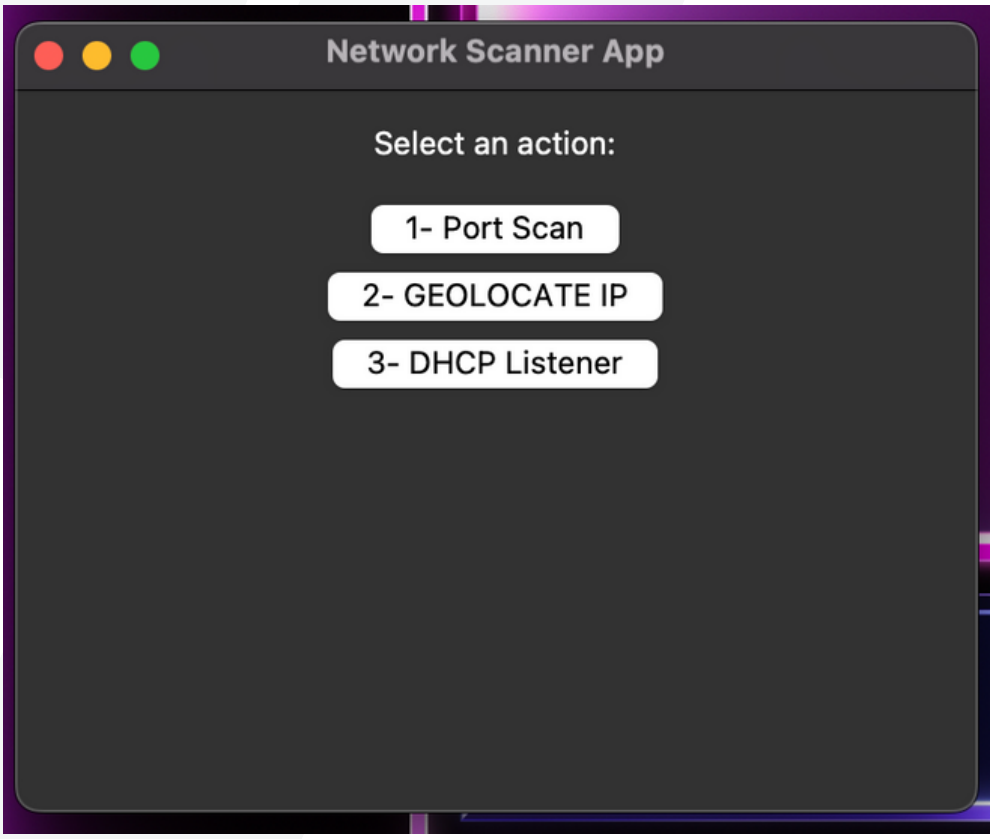
NETWORK SCANNER APP **REPORT**

JANVIER. 2024

INDEX

Overview

The Network Scanner App is a Python-based application using the Tkinter GUI toolkit for its user interface. The application provides three main functionalities: Port Scanning, IP Geolocation, and DHCP Listening. It leverages various libraries, including Scapy for packet sniffing, socket for port scanning, and ipinfo for IP geolocation.

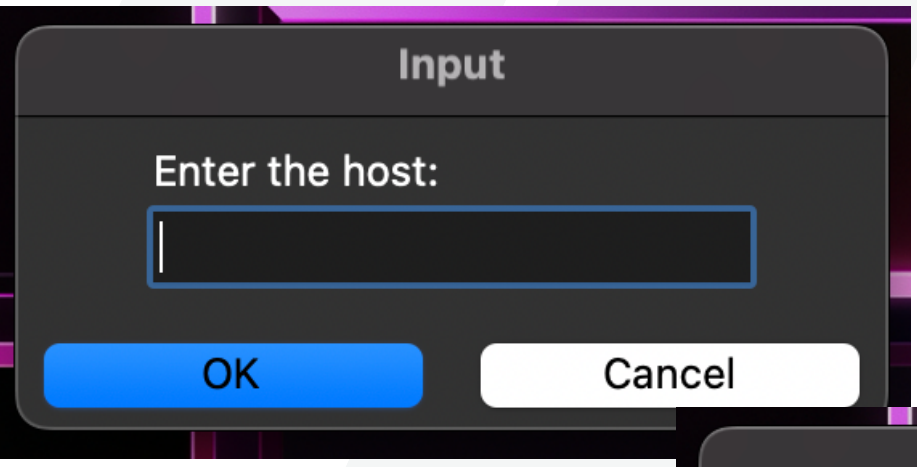


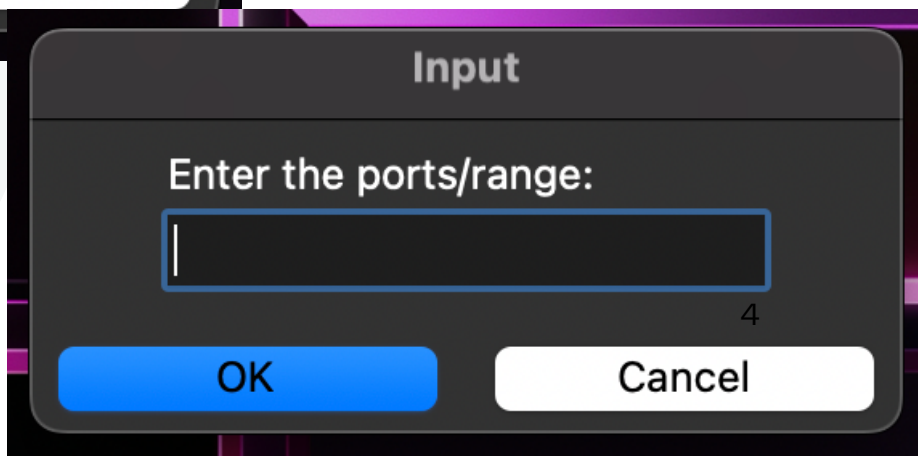
Functionalities

1. Port Scan

User Interaction:

- The user is prompted to enter a target host and a range of ports.
- The application then performs a port scan on the specified range.
- Results are displayed in a pop-up messagebox.





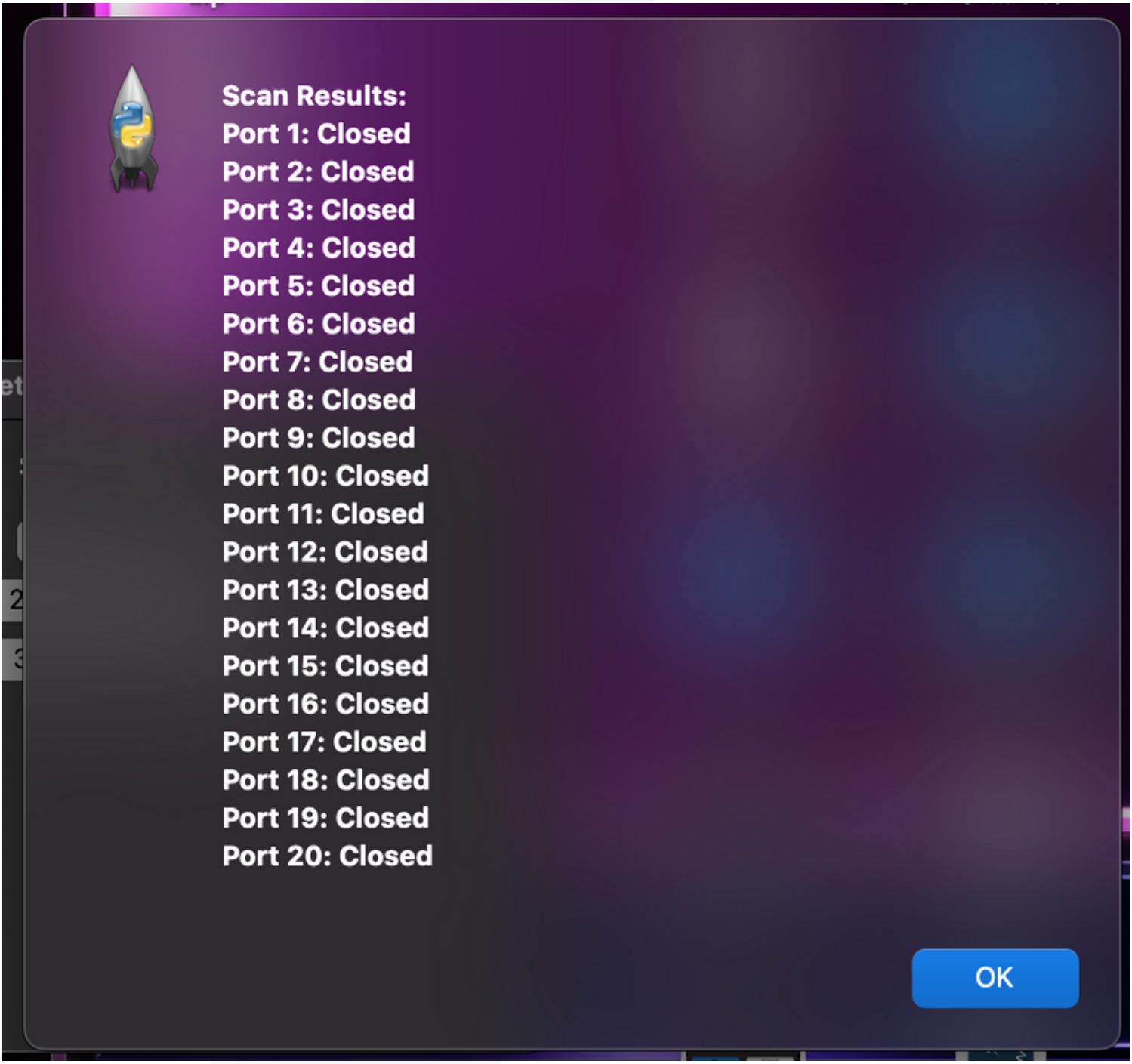
Functionalities

1. Port Scan

Implementation:

- Uses the socket library to attempt connections to each port in the specified range.
- Provides information on whether each port is open or closed.

RESULT

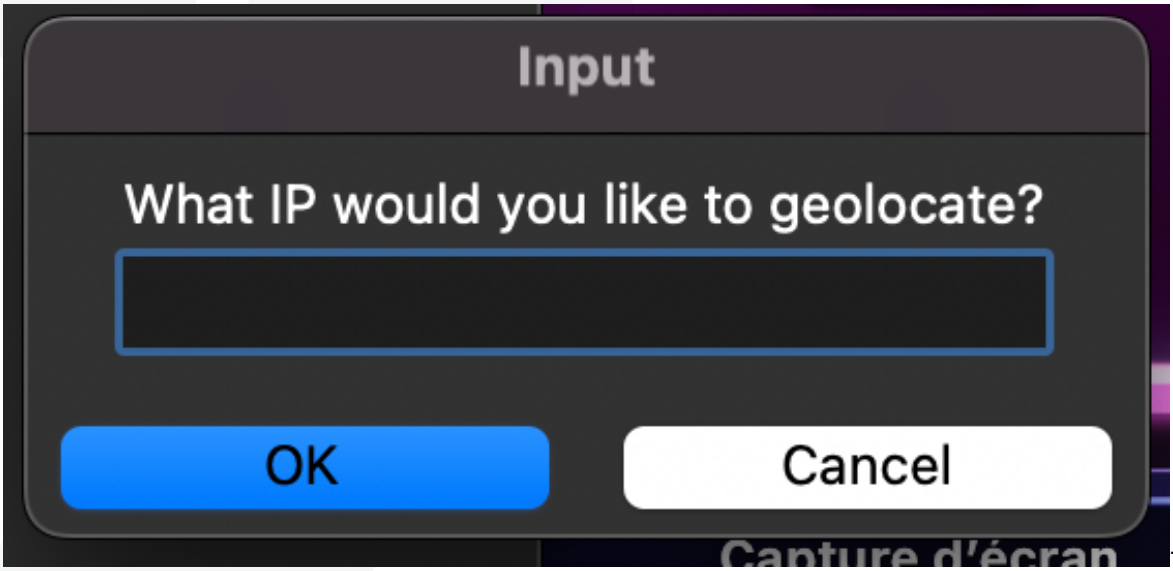

Scan Results:
Port 1: Closed
Port 2: Closed
Port 3: Closed
Port 4: Closed
Port 5: Closed
Port 6: Closed
Port 7: Closed
Port 8: Closed
Port 9: Closed
Port 10: Closed
Port 11: Closed
Port 12: Closed
Port 13: Closed
Port 14: Closed
Port 15: Closed
Port 16: Closed
Port 17: Closed
Port 18: Closed
Port 19: Closed
Port 20: Closed
OK

Functionalities

2. IP Geolocation

User Interaction:

- The user is prompted to enter an IP address for geolocation.
- The application queries the ipinfo service to retrieve geolocation details.
- Results are displayed in a pop-up messagebox.



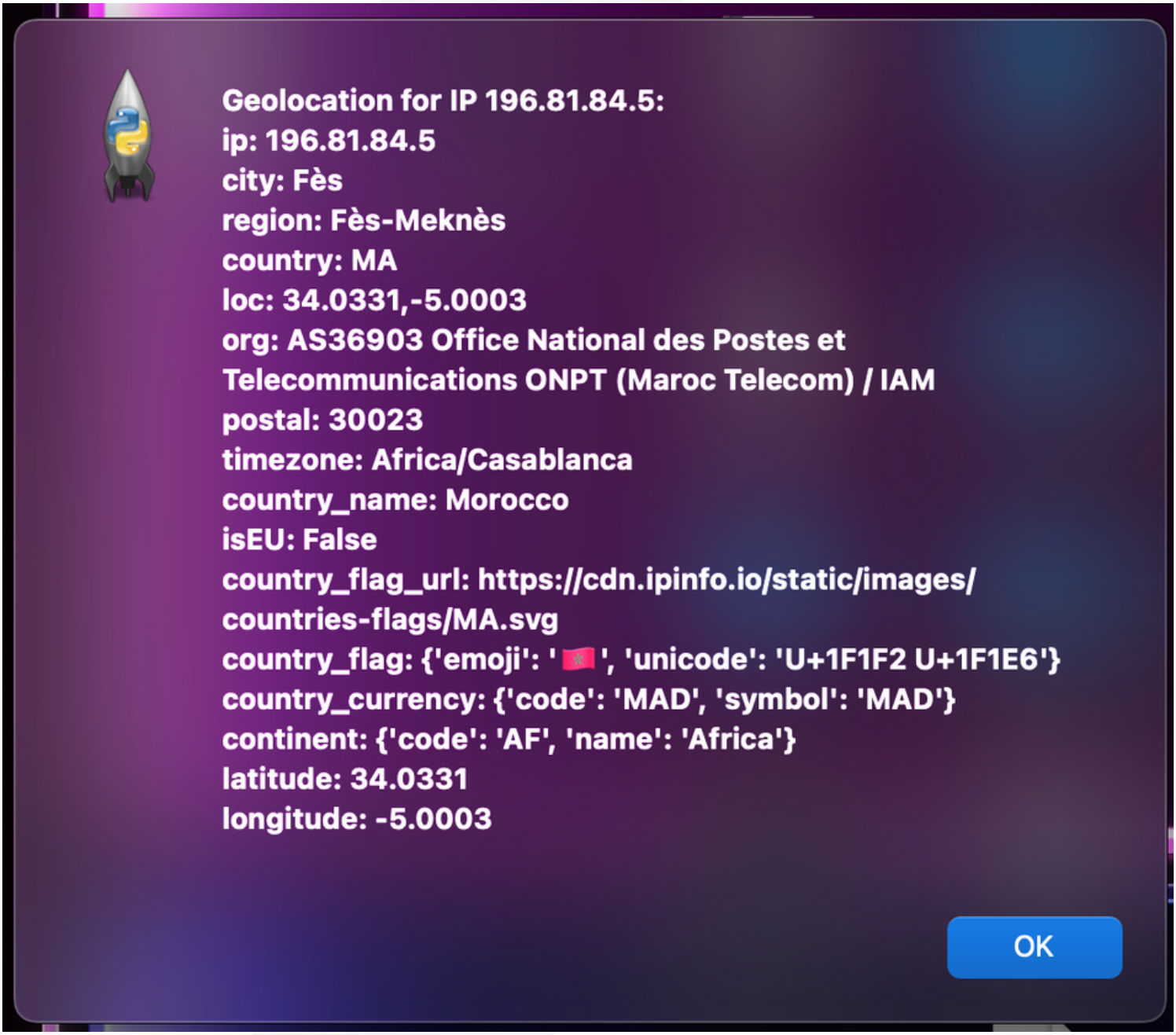
Functionalities

2. IP Geolocation

Implementation:

- Utilizes the ipinfo library to obtain detailed information about the provided IP address.
- Displays various geolocation details, such as city, country, and organization.

RESULT

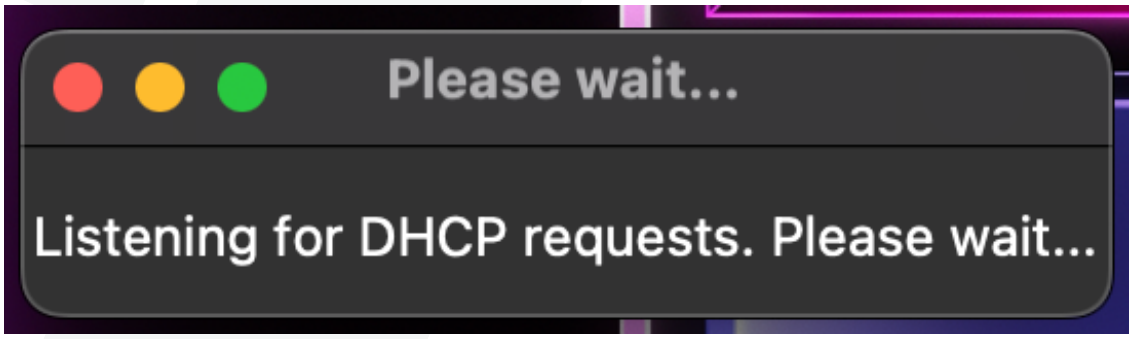

Geolocation for IP 196.81.84.5:
ip: 196.81.84.5
city: Fès
region: Fès-Meknès
country: MA
loc: 34.0331,-5.0003
org: AS36903 Office National des Postes et Telecommunications ONPT (Maroc Telecom) / IAM
postal: 30023
timezone: Africa/Casablanca
country_name: Morocco
isEU: False
country_flag_url: https://cdn.ipinfo.io/static/images/countries-flags/MA.svg
country_flag: {'emoji': '🇲🇦', 'unicode': 'U+1F1F2 U+1F1E6'}
country_currency: {'code': 'MAD', 'symbol': 'MAD'}
continent: {'code': 'AF', 'name': 'Africa'}
latitude: 34.0331
longitude: -5.0003
OK

Functionalities

3. DHCP Listener

User Interaction:

- Clicking the "DHCP Listener" button initiates the DHCP listening functionality.
- A waiting window is shown during the listening process.
- Detected DHCP requests are displayed in a pop-up messagebox.



Functionalities

3. DHCP Listener

Implementation:

- Utilizes the Scapy library to sniff DHCP packets on UDP ports 67 and 68.
- Extracts relevant information from DHCP packets, such as MAC address, requested IP, hostname, and vendor ID.
- Displays DHCP request details in a messagebox after stopping the listening.

RESULT


[2024-01-21 - 23:38:19]:
0a:c0:c 3:9c :c 2:7c - Tab-S7-de-
Jamal / android-dhcp-13
requested 192.168.11.100
OK

Code Structure

Classes

- **NetworkScannerApp:**
  - Manages the main application and GUI.
  - Contains methods for each functionality: port_scan, geo_locate, dhcp_listener.
  - Utilizes separate threads for DHCP listening to avoid freezing the GUI during packet sniffing.

```
class NetworkScannerApp:
```

Code Structure

Methods

- **create_widgets:**
  - Creates and organizes GUI elements.

```
def create_widgets(self):
```

- **port_scan:**
  - Prompts the user for a target host and port range.
  - Performs a port scan and displays the results.

```
def port_scan(self):
```

- **scan_ports:**
  - Performs a port scan using socket connections and returns the results.

```
def scan_ports(self, host, ports):
```

Code Structure

Methods

- **geo_locate:**
  - Prompts the user for an IP address.
  - Uses the ipinfo service to retrieve and display geolocation details.

```
def geo_locate(self):
```

- **dhcp_listener:**
  - Initiates DHCP packet sniffing in a separate thread.
  - Shows a waiting window during the listening process.

```
def dhcp_listener(self):
```

Code Structure

Methods

- **start_dhcp_sniffing:**
  - Sets up a BPF filter and starts DHCP packet sniffing.

```
def start_dhcp_sniffing(self, callback):
```

- **show_dhcp_result:**
  - Destroys the waiting window and displays DHCP results in a pop-up messagebox.

```
def show_dhcp_result(self, result):
```

Dependencies

```
import tkinter as tk
from tkinter import messagebox
from tkinter import simpledialog
from scapy.all import Ether, DHCP, sniff
import socket
from threading import Thread
import ipinfo
import time
```

The application relies on the following external libraries:

- **tkinter:** For GUI development.
- **scapy:** For packet sniffing and DHCP listening.
- **ipinfo:** For IP geolocation.
- **socket:** For port scanning.
- **threading:** For managing threads.
- **time:** For timestamping DHCP results.

Compilation and Execution

Compilation

1- File Location:

- The Network Scanner App is located in the file named "Network_Scanner.py".

2- Navigate to the Directory:

- Open a terminal and navigate to the directory where "Network_Scanner.py" is located.

```
~ (0.026s)
cd Desktop/Network_Scanner_Off/
```

Compilation and Execution

Execution:

1- Run the Application:

- Execute the following command to run the Network Scanner App.

```
~/Desktop/Network_Scanner_Off
python3 Network_Scanner.py
```

2- User Interface:

- The application's user interface will appear, presenting you with options for port scanning, IP geolocation, and DHCP listening.

Compilation and Execution

Execution:

3- Follow On-Screen Prompts:

- Follow the on-screen prompts to input necessary details for each functionality, such as target host, port range, or IP address.

4- Review Results:

- After completing each operation, the application will display results in pop-up message boxes.

Compilation and Execution

Execution:

5- Terminate the Application:

- Close the application's main window to terminate the program.

6- View DHCP Results:

- For DHCP listening, a waiting window will be displayed during the listening process. Detected DHCP requests will be shown in a pop-up messagebox once the listening is complete.

Ideas for Future Enhancements

- Graphical Representation.
- Persistent User Settings.
- Database Integration.
- Network Security Checks.
- User Authentication for DHCP Listener.
- Customizable BPF Filters.

Conclusion

The Network Scanner App effectively combines various network-related functionalities into a user-friendly GUI application. By addressing the recommended improvements, the application can enhance its reliability, user experience, and maintainability.