PROJECT TITLE : Comprehensive Networking Mapping with Nmap

PROBLEM STATEMENT:

In today's network environments, ensuring the security and integrity of network infrastructure is crucial. Existing tools for network mapping and vulnerability scanning can be complex or lack certain functionalities. This project aims to provide a comprehensive and user-friendly solution for network mapping using Nmap, enabling users to perform various types of scans effectively.

INTRODUCTION:

Network mapping is a critical aspect of network administration and security. Nmap (Network Mapper) is a powerful and versatile tool used for network discovery and security auditing. However, its command-line interface and wide array of features can be overwhelming for some users. This project aims to harness the capabilities of Nmap and present them through an intuitive Python-based command-line interface. By providing a streamlined approach to performing various types of scans—such as ping scans, port scans, host scans, and OS scans—the project seeks to empower network administrators and security professionals with a more accessible and efficient tool for identifying vulnerabilities and managing network resources.

EXISTING SYSTEM:

Current network mapping tools, while effective in their own right, often pose significant usability challenges. Tools like Nmap, although highly functional, require users to be proficient in command-line operations and scripting. The complexity of configuring different types of scans and interpreting their results can be a barrier for less experienced users. Moreover, many existing tools do not offer an integrated interface that consolidates various scanning functionalities, leading to a fragmented user experience. These challenges highlight the need for a more user-friendly solution that simplifies the scanning process and makes comprehensive network assessment more accessible.

PROPOSED SYSTEM:

The proposed system aims to address the limitations of existing tools by providing a cohesive and user-friendly interface for network mapping using Nmap. The system is developed as a Python-based command-line tool that integrates multiple Nmap functionalities into a single interface. This integration allows users to perform a variety of scans—including ping scans, port scans, host scans, and OS scans—through a unified and simplified command-line interface. The proposed solution not only streamlines the scanning process but also enhances the overall efficiency and effectiveness of network vulnerability assessments. By automating and consolidating scan functionalities, the tool provides a more intuitive experience for users, reducing the learning curve associated with traditional network scanning tools.

SOFTWARE AND HARDWARE REQUIREMENTS:

Software:

- **Python 3.x:** The programming language used to develop the tool.
- **python-nmap library:** A Python wrapper for the Nmap tool that simplifies interaction with Nmap.
- **Nmap:** The network scanning tool that performs the actual scans (must be installed on the system).

Hardware:

- **Computer with Windows OS:** Required for path configurations and running the tool.
- **Network connection:** Necessary for performing scans on target systems.

MODULES INVOLVED IN THE PROJECT:

Main Module (main.py): Manages user interactions and coordinates different scan functions. This module serves as the entry point for the application, guiding the user through various scan options and executing the selected scans.

Scanner Module (scanner.py): Contains the core functions for executing different types of scans using Nmap. This module leverages the python-nmap library to interface with Nmap and retrieve scan results.

README.md: Provides comprehensive documentation on setting up and running the project, including installation instructions and usage guidelines.

ARCHITECTURE:

The architecture of the project is designed for simplicity and efficiency. It consists of:

- **Main Script (main.py):** Acts as the user interface, allowing users to choose and initiate different types of scans. It handles input from the user, calls the appropriate functions in the scanner module, and displays the results.
  - The main program file.
  - Handles user interface, menu, and interaction.

- **Scanner Module (scanner.py):** Implements functions for performing various scans. It uses the python-nmap library to interact with Nmap, execute scans, and process the results.
  - Contains the functionalities for network scanning using Nmap.

    - Conatains different functions like as follows:
        - Ping Scan
        - Port Scan
        - Host Scan
        - OS Scan
        - Exit

- **Output Handling:** Results from the scans are processed and formatted for clear presentation. This may include generating reports or displaying results in a structured format.

Description of functions used in my project:

Here is a brief description of each function in your project:

1. **Ping Scan:**
    - A Ping Scan is used to check if a host is alive or reachable. It sends ICMP echo requests to the target and waits for a response. If the target responds, it is considered up and reachable.
2. **Port Scan:**
    - A Port Scan identifies open ports on a target machine. It scans a range of specified ports to determine which are open, closed, or filtered. Open ports indicate services running on the target that might be exploitable.
3. **Host Scan:**
    - A Host Scan gathers information about the specified target hosts. It can identify various details like hostnames, IP addresses, MAC addresses, and the status of the host (up or down).
4. **OS Scan:**
    - An OS Scan attempts to determine the operating system running on the target machine. It uses various techniques to identify the OS type, vendor, and version based on responses to network probes.
5. **Exit:**

  - Exits the program. This option allows the user to quit the application gracefully.

Dependencies:

- **python-nmap**: Used for network scanning and obtaining detailed information about hosts, open ports, and operating systems.

Instructions to Run the Code:

To run your project, ensure the following steps are followed:

1. **Installation of Dependencies**:
   - Install the python-nmap library using pip:

   Command used: pip install python-nmap

2. **Execution**:
   - Ensure that Nmap is installed on your system and the path to nmap.exe is correctly set
   - Run your main.py script to interact with the command-line interface for performing various network scans (Ping Scan, Port Scan, Host Scan, OS Scan, etc.).

PROJECT CODE:

Main.py:

```
import scanner

def main():

    print("Welcome to Comprehensive Networking Mapping with Nmap!\n")


    while True:

        print_menu()

        choice = input("Enter your choice (1-5): ")


        if choice == '1':
```

```
    target = input("Enter target IP or range (e.g., 192.168.1.1 or 192.168.1.0/24): ")
    result = scanner.ping_scan(target)
    print_ping_scan_result(result)

elif choice == '2':
    target = input("Enter target IP: ")
    ports = input("Enter ports (e.g., 22-1000, 80, 443): ")
    result = scanner.port_scan(target, ports)
    print_port_scan_result(result)

elif choice == '3':
    target = input("Enter target IP: ")
    result = scanner.host_scan(target)
    if result:  # Check if hosts were found
        print_host_scan_result(result)
    else:
        print(f"No hosts found for {target}")

elif choice == '4':
    target = input("Enter target IP: ")
    result = scanner.os_scan(target)
    print_os_scan_result(result)

elif choice == '5':
    print("Exiting...")
    break
```

```
        else:
            print("Invalid choice. Please enter a number from 1 to 7.")

def print_menu():
    print("Choose an option:")
    print("1. Ping Scan")
    print("2. Port Scan")
    print("3. Host Scan")
    print("4. OS Scan")
    print("5. Exit")

def print_ping_scan_result(result):
    for host in result:
        print(f"Host: {host['host']}")
        print(f"Status: {host['status']}")
        print(f"MAC Address: {host['mac_address']}")
        print(f"Vendor: {host['vendor']}")
        print("Open Ports:")
        for port_info in host['open_ports']:
            print(f"  Port: {port_info['port']}")
            print(f"  State: {port_info['state']}")
            print(f"  Service: {port_info['service']}")
            print()

def print_port_scan_result(result):
    for port_info in result:
        print(f"Host: {port_info['host']}")
        print(f"Port: {port_info['port']}")
        print(f"State: {port_info['state']}")
        print(f"Service: {port_info['service']}")
        print()

def print_host_scan_result(result):
    for host in result:
        print(f"Host: {host['host']}")
        print(f"Status: {host['status']}")
        print(f"MAC Address: {host['mac_address']}")
        if 'hostnames' in host:
            print(f"Hostnames: {host['hostnames']}")
        print(f"Addresses: {host['addresses']}")
        print()

def print_os_scan_result(result):
    for os_info in result:
        print(f"Host: {os_info['host']}")
        print(f"OS Type: {os_info['os_type']}")
        print(f"OS Vendor: {os_info['os_vendor']}")
        print(f"OS Family: {os_info['os_family']}")
        print()


if __name__ == "__main__":
    main()
```

Scanner.py:

```
 import nmap
nmap_path ='C:\\Program Files (x86)\\Nmap\\nmap.exe'
nmap.PortScanner.nmap_path = nmap_path
def ping_scan(target):
    nm = nmap.PortScanner(nmap_search_path=('C:\\Program Files (x86)\\Nmap\\nmap.exe',))
    nm.scan(target, arguments='-sV')  # Use -sV for service version detection
    hosts_list = []

    for host in nm.all_hosts():
        if nm[host].state() == 'up':
            host_details = {
                'host': host,
                'status': nm[host].state(),
                'mac_address': get_mac_address(nm[host]['addresses']),
                'vendor': nm[host]['vendor'],
                'open_ports': []
            }

            for proto in nm[host].all_protocols():
                lport = nm[host][proto].keys()
                for port in lport:
                    port_info = {
                        'port': port,
                        'state': nm[host][proto][port]['state'],
                        'service': nm[host][proto][port]['name']
                    }
                    host_details['open_ports'].append(port_info)

            hosts_list.append(host_details)

    return hosts_list

def get_mac_address(addresses):
    try:
        return addresses['mac']
    except KeyError:
        return "N/A"

def host_scan(target):
    nm = nmap.PortScanner(nmap_search_path=('C:\\Program Files (x86)\\Nmap\\nmap.exe',))
    nm.scan(target)
    hosts_list = []

    if nm.all_hosts():
        for host in nm.all_hosts():
            if nm[host].state() == 'up':
                host_details = {
                    'host': host,
                    'status': nm[host].state(),
                    'hostnames': nm[host]['hostnames'],
                    'addresses': nm[host]['addresses'],
                    'mac_address': get_mac_address(nm[host]['addresses'])
                }
                hosts_list.append(host_details)
    else:
        print(f"No hosts found for {target}")

    return hosts_list

def port_scan(target, ports):
    nm = nmap.PortScanner(nmap_search_path=('C:\\Program Files (x86)\\Nmap\\nmap.exe',))
    nm.scan(target, arguments='-sV -p {}'.format(ports))

    open_ports = []
    try:
        for host in nm.all_hosts():
            for proto in nm[host].all_protocols():
                if proto == 'tcp':
                    for port in nm[host][proto].keys():
                        port_info = {
                            'host': host,
                            'port': port,
                            'state': nm[host][proto][port]['state'],
                            'service': nm[host][proto][port]['name']
                        }
                        open_ports.append(port_info)
    except KeyError as e:
        print(f"No results found for {target}:{ports}")
    return open_ports

def host_scan(target):
    nm = nmap.PortScanner(nmap_search_path=('C:\\Program Files (x86)\\Nmap\\nmap.exe',))
    nm.scan(target)
    hosts_list = []

    if nm.all_hosts():
        for host in nm.all_hosts():
            if nm[host].state() == 'up':
                host_details = {
                    'host': host,
                    'status': nm[host].state(),
                    'hostnames': nm[host]['hostnames'],
                    'addresses': nm[host]['addresses'],
                    'mac_address': get_mac_address(nm[host]['addresses'])
                }
                hosts_list.append(host_details)
    else:
        print(f"No hosts found for {target}")

    return hosts_list

def os_scan(target):
    nm = nmap.PortScanner(nmap_search_path=('C:\\Program Files (x86)\\Nmap\\nmap.exe',))
    nm.scan(target, arguments='-O')
    os_info_list = []

    try:
        if target in nm.all_hosts() and 'osmatch' in nm[target]:
            os_matches = nm[target]['osmatch']
            for os_match in os_matches:
                os_info = {
                    'host': target,
                    'os_type': os_match['osclass'][0]['osfamily'],
                    'os_vendor': os_match['osclass'][0]['vendor'],
                    'os_family': os_match['osclass'][0]['osfamily']
                }
                os_info_list.append(os_info)
        else:
            print(f"No OS information found for {target}")
    except KeyError as e:
        print(f"Error: {e}")

    return os_info_list


def disable_dns_resolution_scan(target):
    nm = nmap.PortScanner(nmap_search_path=('C:\\Program Files (x86)\\Nmap\\nmap.exe',))
    nm.scan(target, arguments='-n')
    return nm[target]['hostnames'], nm[target]['addresses']
```

TESTING:

The project underwent rigorous testing to ensure accuracy and reliability. The testing scenarios included:

- **Ping Scan on Local Network:** Verified the ability to detect active hosts within a local network.
- **Port Scan on Specific IP Addresses and Port Ranges:** Ensured that the tool accurately identifies open ports and services on target systems.
- **Host Scan to Identify Active Hosts:** Tested the tool's capability to discover active hosts and gather basic information such as MAC addresses and hostnames.
- **OS Scan to Detect Operating Systems:** Validated the accuracy of OS detection features by comparing results with known configurations.

Test results were cross-referenced with manual Nmap command outputs to ensure consistency and accuracy.

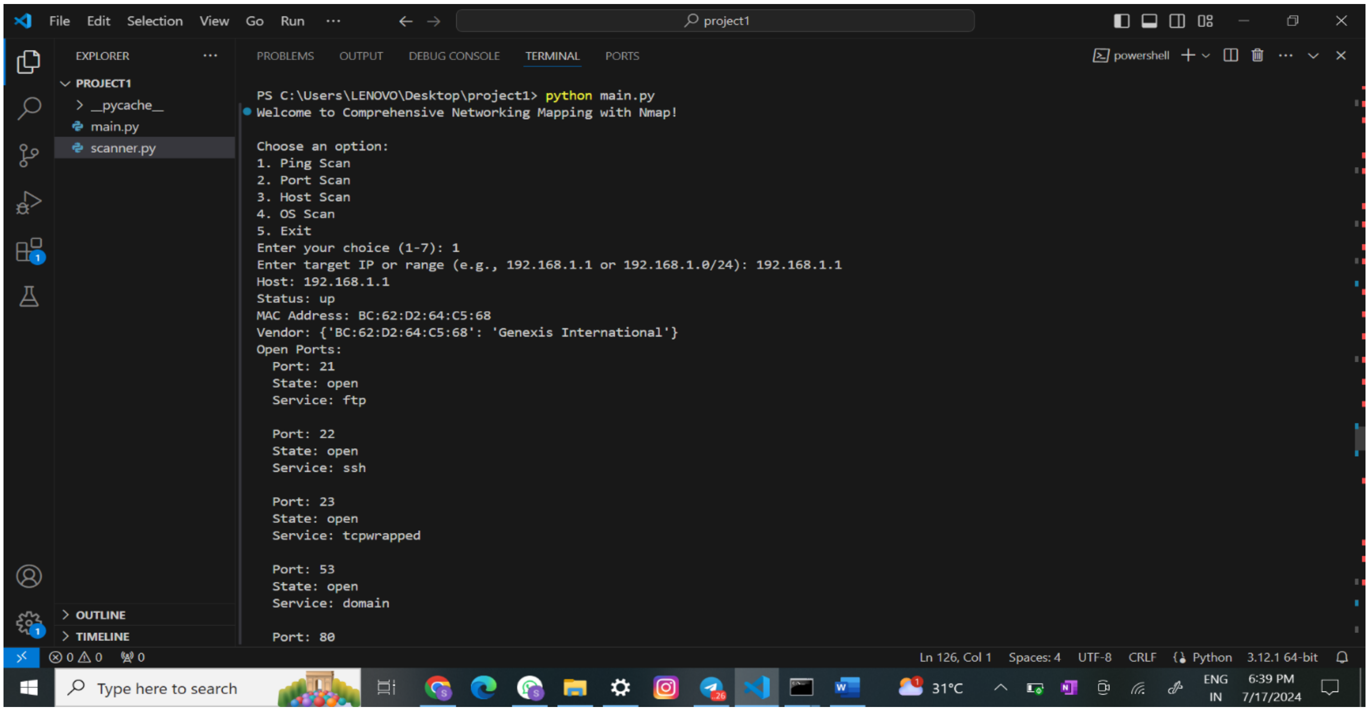
Sample inputs and outputs :

```
PS C:\Users\LENOVO\Desktop\project1> python main.py

Welcome to Comprehensive Networking Mapping with Nmap!


Choose an option:

1. Ping Scan

2. Port Scan

3. Host Scan

4. OS Scan

5. Exit

Enter your choice (1-5): 1

Enter target IP or range (e.g., 192.168.1.1 or 192.168.1.0/24): 192.168.1.1

Host: 192.168.1.1
```

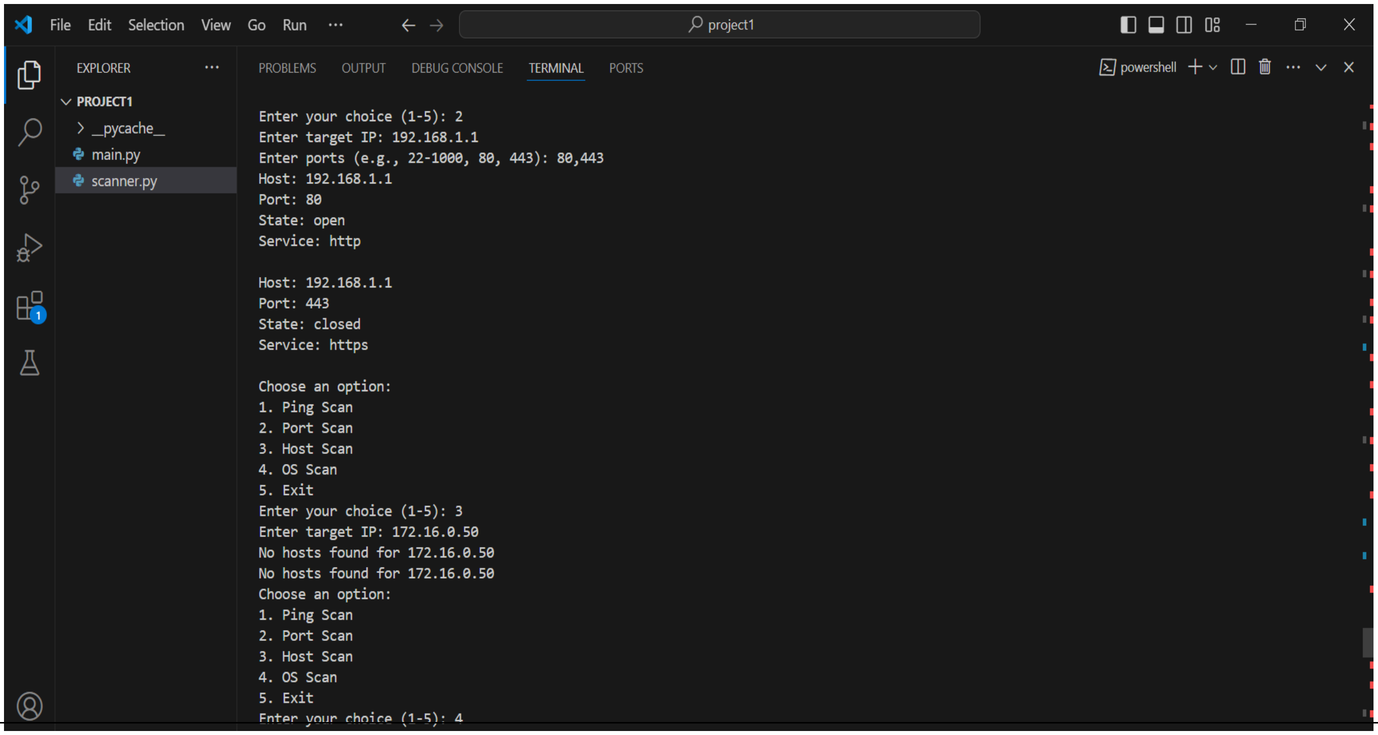
```
Status: up
MAC Address: BC:62:D2:64:C5:68
Vendor: {'BC:62:D2:64:C5:68': 'Genexis International'}
Open Ports:
  Port: 21
  State: open
  Service: ftp

  Port: 22
  State: open
  Service: ssh

  Port: 23
  State: open
  Service: tcpwrapped

  Port: 53
  State: open
  Service: domain

  Port: 80
  State: open
  Service: http

Choose an option:
1. Ping Scan
2. Port Scan
```

```
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-5): 2
Enter target IP: 192.168.1.1
Enter ports (e.g., 22-1000, 80, 443): 80,443
Host: 192.168.1.1
Port: 80
State: open
Service: http

Host: 192.168.1.1
Port: 443
State: closed
Service: https

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 4
Enter target IP: 192.168.1.1
Host: 192.168.1.1
OS Type: Linux
OS Vendor: Linux
OS Family: Linux


Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 3
Enter target IP: 8.8.8.8
Host: 8.8.8.8
Status: up
MAC Address: N/A
Hostnames: [{'name': 'dns.google', 'type': 'PTR'}]
Addresses: {'ipv4': '8.8.8.8'}


Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 5
Exiting...
```

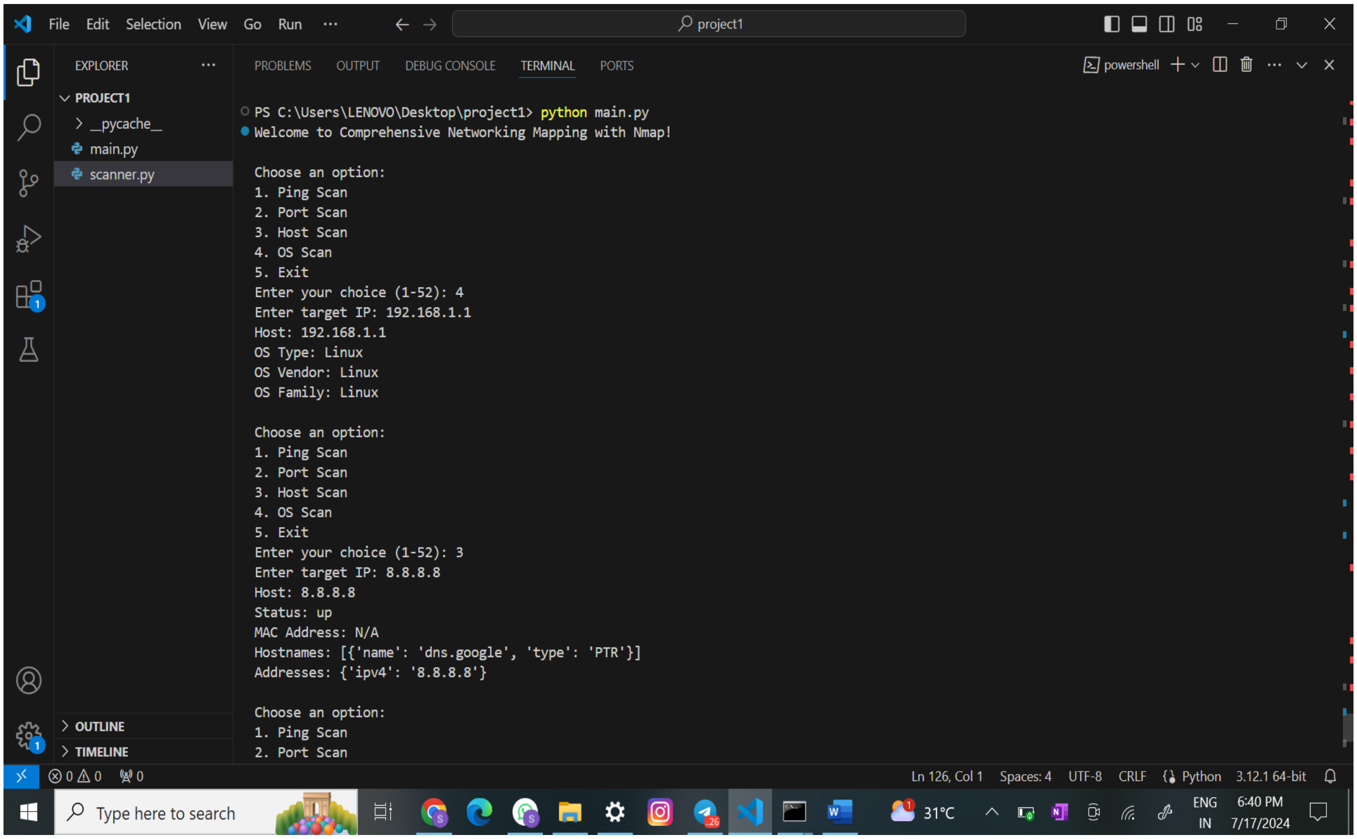
Some Screen shots of executed outputs:



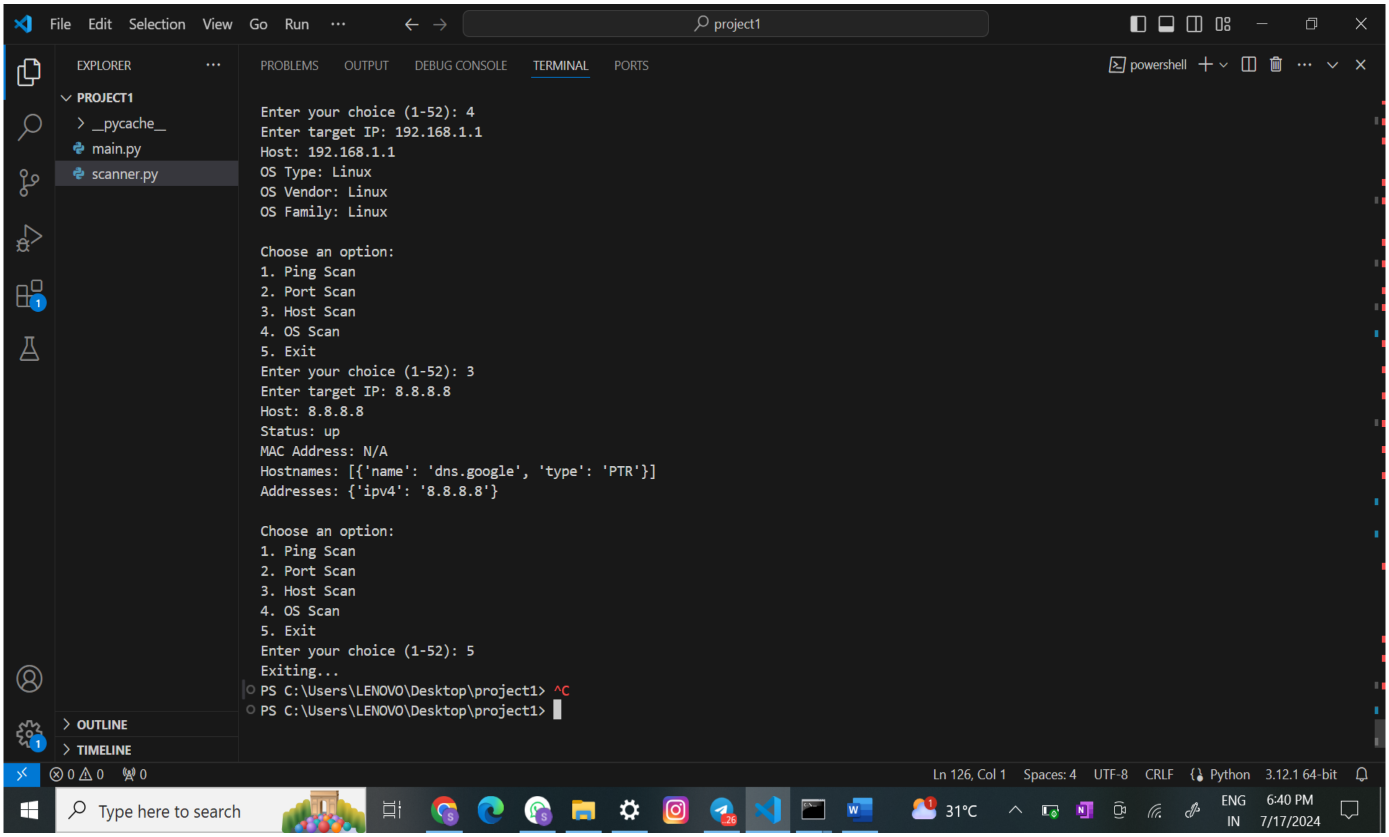
```
PS C:\Users\LENOVO\Desktop\project1> python main.py
Welcome to Comprehensive Networking Mapping with Nmap!

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-7): 1
Enter target IP or range (e.g., 192.168.1.1 or 192.168.1.0/24): 192.168.1.1
Host: 192.168.1.1
Status: up
MAC Address: BC:62:D2:64:C5:68
Vendor: {'BC:62:D2:64:C5:68': 'Genexis International'}
Open Ports:
  Port: 21
  State: open
  Service: ftp

  Port: 22
  State: open
  Service: ssh

  Port: 23
  State: open
  Service: tcpwrapped

  Port: 53
  State: open
  Service: domain

  Port: 80
```



```
Enter your choice (1-5): 2
Enter target IP: 192.168.1.1
Enter ports (e.g., 22-1000, 80, 443): 80,443
Host: 192.168.1.1
Port: 80
State: open
Service: http

Host: 192.168.1.1
Port: 443
State: closed
Service: https

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-5): 3
Enter target IP: 172.16.0.50
No hosts found for 172.16.0.50
No hosts found for 172.16.0.50
Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-5): 4
```


File Edit Selection View Go Run ···
project1
EXPLORER
PROJECT1
__pycache__
main.py
scanner.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Ln 126, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.1 64-bit
Type here to search
31°C
ENG IN 6:40 PM 7/17/2024

```
PS C:\Users\LENOVO\Desktop\project1> python main.py
Welcome to Comprehensive Networking Mapping with Nmap!

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 4
Enter target IP: 192.168.1.1
Host: 192.168.1.1
OS Type: Linux
OS Vendor: Linux
OS Family: Linux

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 3
Enter target IP: 8.8.8.8
Host: 8.8.8.8
Status: up
MAC Address: N/A
Hostnames: [{'name': 'dns.google', 'type': 'PTR'}]
Addresses: {'ipv4': '8.8.8.8'}

Choose an option:
1. Ping Scan
2. Port Scan
```


File Edit Selection View Go Run ···
project1
EXPLORER
PROJECT1
__pycache__
main.py
scanner.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Ln 126, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.1 64-bit
Type here to search
31°C
ENG IN 6:40 PM 7/17/2024

```
Enter your choice (1-52): 4
Enter target IP: 192.168.1.1
Host: 192.168.1.1
OS Type: Linux
OS Vendor: Linux
OS Family: Linux

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 3
Enter target IP: 8.8.8.8
Host: 8.8.8.8
Status: up
MAC Address: N/A
Hostnames: [{'name': 'dns.google', 'type': 'PTR'}]
Addresses: {'ipv4': '8.8.8.8'}

Choose an option:
1. Ping Scan
2. Port Scan
3. Host Scan
4. OS Scan
5. Exit
Enter your choice (1-52): 5
Exiting...
PS C:\Users\LENOVO\Desktop\project1> ^C
PS C:\Users\LENOVO\Desktop\project1>
```

CONCLUSION:

The 'Comprehensive Networking Mapping with Nmap' project successfully addresses the need for a more accessible and user-friendly network scanning tool. By integrating various Nmap functionalities into a cohesive Python-based interface, the project enhances the usability and effectiveness of network mapping and vulnerability assessments. The tool simplifies the process of performing complex network scans, making it a valuable asset for network administrators and security professionals. Through its streamlined approach and comprehensive feature set, the project contributes to more effective network management and security practices.