Setup and Run Guide for Ethical IP Scanning Agent

This guide provides comprehensive, step-by-step instructions to get your ethical IP scanning agent up and running on your macOS M2 device.

Important: Read the `README.md` file first for ethical disclaimers and project overview.

1. Project Setup

First, let's create your project directory and set up a Python virtual environment.

1. **Open your Terminal application.**
2. **Create Project Directory:**

```
mkdir ip_scanner_agent
```

3.
4. **Navigate into the New Directory:**

```
cd ip_scanner_agent
```

5.
6. Create a Python Virtual Environment:
   This isolates your project's Python dependencies.

```
python3 -m venv venv
```

7.
8. Activate the Virtual Environment:
   You should see (venv) appear at the beginning of your terminal prompt.

```
source venv/bin/activate
```

9.
10. Install Python Dependencies:
    These libraries are required for Nmap interaction, XML parsing, and API calls.

```
pip install python-nmap requests beautifulsoup4 lxml
```

11.

2. Tool Installation & API Key

Next, install Nmap and prepare your LLM access.

1. Install Homebrew (if not already installed):
   Homebrew is a package manager for macOS, used to install Nmap.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2.
3. Follow any on-screen prompts (e.g., for your password).
4. **Install Nmap:**

```
brew install nmap
```

5.
6. **Verify Nmap Installation:**

```
nmap --version
```

7.
8. You should see Nmap version information.
9. Obtain a Together AI API Key:
   Your agent will use Together AI for LLM analysis.
   - Go to the Together AI website: https://www.together.ai/
   - Sign up or log in to your account.
   - Navigate to your **API Keys** section (usually in your dashboard or settings).
   - Generate a new API key (it will typically start with `sk-`).
   - **Copy this key and keep it secure.** You will paste it into your Python code in the next step.

3. Code Setup (`agent.py`)

Now, let's create the main Python script for your agent.

1. Create the agent.py file:
   Ensure you are in the ip_scanner_agent directory with your virtual environment activated.

```
nano agent.py
```

2.
3. Paste the Complete Python Code:
   Copy the entire Python code block below and paste it into the nano editor.
   IMPORTANT:
   - **Replace** `"YOUR_TOGETHER_API_KEY"` with the actual API key you obtained from Together AI.

- You can optionally change `TOGETHER_MODEL` to another model available on Together AI (e.g., `"mistralai/Mixtral-8x7B-Instruct-v0.1"`), but ensure it's a chat-optimized model.

```
# agent.py - Ethical IP Scanning Agent (Command-Line)

import nmap
import subprocess
import json
import time
import os
from bs4 import BeautifulSoup
import requests

# --- Configuration ---
# TOGETHER AI API Configuration
TOGETHER_API_URL = "https://api.together.ai/v1/chat/completions"
# Choose a model from Together AI. Examples:
# "meta-llama/Llama-3-8b-chat-hf"
# "mistralai/Mixtral-8x7B-Instruct-v0.1"
# "mistralai/Mistral-7B-Instruct-v0.2"
TOGETHER_MODEL = "meta-llama/Llama-3-8b-chat-hf" # Or your preferred Together AI model

# IMPORTANT: Replace "YOUR_TOGETHER_API_KEY" with your actual API key from together.ai
TOGETHER_API_KEY = "YOUR_TOGETHER_API_KEY"

# --- Helper Functions ---

def run_nmap_scan(target_ip, scan_type="-sV -O -p-"):
    """
    Executes an Nmap scan and returns the XML output.
    -sV: Service version detection
    -O: OS detection
    -p-: Scan all ports (1-65535)
    -oX -: Output in XML format to stdout
    """
    print(f"[*] Starting Nmap scan on {target_ip} with options: {scan_type}...")
    try:
        scan_options = scan_type.split()

        # IMPORTANT: Adding 'sudo' here for root privileges required by Nmap
        command = ["sudo", "nmap"] + scan_options + ["-oX", "-", target_ip]

        process = subprocess.run(command, capture_output=True, text=True, check=True)
        print("[+] Nmap scan completed.")
```

```
        return process.stdout
    except subprocess.CalledProcessError as e:
        print(f"[-] Nmap scan failed: {e}")
        print(f"    Stderr: {e.stderr}")
        return None
    except FileNotFoundError:
        print("[-] Nmap command not found. Please ensure Nmap is installed and in your PATH.")
        return None

def parse_nmap_xml(xml_output):
    """
    Parses Nmap XML output to extract relevant information.
    Returns a dictionary of host information.
    """
    if not xml_output:
        return {}

    # Use 'lxml-xml' for proper XML parsing to avoid warnings
    soup = BeautifulSoup(xml_output, 'lxml-xml')
    hosts_info = {}

    for host_tag in soup.find_all('host'):
        ip_address = host_tag.find('address', addrtype='ipv4')['addr'] if host_tag.find('address',
addrtype='ipv4') else 'N/A'
        hostname = host_tag.find('hostname')['name'] if host_tag.find('hostname') else 'N/A'
        status = host_tag.find('status')['state'] if host_tag.find('status') else 'N/A'

        ports_info = []
        for port_tag in host_tag.find_all('port'):
            port_id = port_tag['portid']
            protocol = port_tag['protocol']
            state = port_tag.find('state')['state'] if port_tag.find('state') else 'N/A'
            service_name = port_tag.find('service')['name'] if port_tag.find('service') else 'N/A'
            service_product = port_tag.find('service')['product'] if port_tag.find('service') and 'product' in
port_tag.find('service').attrs else 'N/A'
            service_version = port_tag.find('service')['version'] if port_tag.find('service') and 'version' in
port_tag.find('service').attrs else 'N/A'

            ports_info.append({
                'port': port_id,
                'protocol': protocol,
                'state': state,
                'service': service_name,
                'product': service_product,
                'version': service_version
            })
```

```
        os_match = host_tag.find('osmatch')['name'] if host_tag.find('osmatch') else 'N/A'

        hosts_info[ip_address] = {
            'hostname': hostname,
            'status': status,
            'os_match': os_match,
            'ports': ports_info
        }
    return hosts_info

def query_together_ai(messages):
    """
    Sends a list of messages to the Together AI LLM and returns the response.
    """
    if not TOGETHER_API_KEY or TOGETHER_API_KEY == "YOUR_TOGETHER_API_KEY":
        print("[-] Error: TOGETHER_API_KEY is not set. Please get an API key from together.ai.")
        return "Error: Together AI API key not configured."

    print(f"[*] Querying Together AI with {len(messages)} messages...")
    try:
        headers = {
            'Content-Type': 'application/json',
            'Authorization': f'Bearer {TOGETHER_API_KEY}'
        }
        data = {
            "model": TOGETHER_MODEL,
            "messages": messages, # Together AI expects a list of messages
            "max_tokens": 1024, # Adjust as needed
            "temperature": 0.7, # Adjust creativity
            "top_p": 0.7,
            "top_k": 50,
            "repetition_penalty": 1 # Adjust to avoid repetition
        }
        response = requests.post(TOGETHER_API_URL, headers=headers, data=json.dumps(data))
        response.raise_for_status() # Raise an HTTPError for bad responses (4xx or 5xx)
        result = response.json()
        print("[+] Together AI response received.")

        # Extract the content from the response
        if result and result.get('choices') and result['choices'][0].get('message') and
result['choices'][0]['message'].get('content'):
            return result['choices'][0]['message']['content'].strip()
        else:
            print(f"[-] Unexpected response structure from Together AI: {result}")
            return "Error: Unexpected response structure from Together AI."
```

```
    except requests.exceptions.ConnectionError:
        print(f"[-] Could not connect to Together AI. Check your internet connection or API URL.")
        return "Error: Could not connect to Together AI."
    except requests.exceptions.RequestException as e:
        print(f"[-] Error querying Together AI: {e}")
        return f"Error: {e}"

def generate_analysis_prompt_messages(scan_results):
    """
    Generates a list of messages for the LLM based on scan results,
    suitable for chat completion APIs like Together AI.
    """
    if not scan_results:
        return [{"role": "user", "content": "Nmap scan yielded no results. Please provide a summary of
an empty scan."}]

    prompt_content = [
        "Analyze the following Nmap scan results. Focus on identifying potentially interesting open
ports, services, and operating systems. Suggest next *legitimate* steps for further reconnaissance or
vulnerability assessment, such as looking for public exploits for identified services, or performing
more detailed scans on specific ports. Do not suggest any illegal or unethical actions like exploiting
vulnerabilities without permission, or generating attack payloads."
    ]

    for ip, info in scan_results.items():
        prompt_content.append(f"\n--- Host: {ip} ---")
        prompt_content.append(f"  Hostname: {info['hostname']}")
        prompt_content.append(f"  Status: {info['status']}")
        prompt_content.append(f"  OS Match: {info['os_match']}")
        prompt_content.append("  Open Ports:")
        if info['ports']:
            for port in info['ports']:
                if port['state'] == 'open':
                    prompt_content.append(
                        f"    - Port: {port['port']}/{port['protocol']} | Service: {port['service']} "
                        f"| Product: {port['product']} | Version: {port['version']}"
                    )
        else:
            prompt_content.append("    No open ports found.")

    return [{"role": "user", "content": "\n".join(prompt_content)}]

def main():
    """
    Main function to orchestrate the scanning and analysis process.
```

```
    """
    target_ip = input("Enter the target IP address or range (e.g., 192.168.1.1 or 192.168.1.0/24): ")

    # Initialize chat history for the session
    chat_history = []

    # Step 1: Run Initial Nmap Scan
    nmap_xml_output = run_nmap_scan(target_ip)

    if nmap_xml_output:
        # Step 2: Parse Nmap XML Output
        parsed_results = parse_nmap_xml(nmap_xml_output)
        print("\n--- Parsed Nmap Results ---")
        print(json.dumps(parsed_results, indent=2))

        if parsed_results:
            # Step 3: Generate Analysis Prompt for LLM (as messages)
            llm_prompt_messages = generate_analysis_prompt_messages(parsed_results)
            chat_history.extend(llm_prompt_messages) # Add user's initial prompt to history

            # Step 4: Query Together AI for Analysis
            ollama_analysis = query_together_ai(chat_history)
            print("\n--- AI Analysis and Suggestions (from Together AI) ---")
            print(ollama_analysis)
            chat_history.append({"role": "assistant", "content": ollama_analysis}) # Add AI's response to
history

            # --- Interactive Follow-up Scans (CLI) ---
            print("\n--- Follow-up Scan Interface ---")
            while True:
                follow_up_command_str = input(
                    "Enter a specific Nmap command (e.g., '-p 3306 --script mysql-info') "
                    "for the target, or type 'quit' to exit: "
                )
                if follow_up_command_str.lower() == 'quit':
                    break
                if not follow_up_command_str.strip():
                    print("No command entered. Please try again or type 'quit'.")
                    continue

                # Execute the follow-up Nmap command
                print(f"\n[*] Running follow-up Nmap scan with options: {follow_up_command_str}...")

                follow_up_xml_output = run_nmap_scan(target_ip, scan_type=follow_up_command_str)

                if follow_up_xml_output:
```

```
                follow_up_parsed_results = parse_nmap_xml(follow_up_xml_output)
                print("\n--- Follow-up Scan Parsed Results ---")
                print(json.dumps(follow_up_parsed_results, indent=2))

                # OPTIONAL: Feed these follow-up results back to LLM for new analysis
                # Generate a new prompt including the latest scan results and previous history
                follow_up_llm_prompt_content = f"Given our previous conversation and the new scan results for {target_ip}:\n{json.dumps(follow_up_parsed_results, indent=2)}\n\nPlease provide further analysis and suggestions based on this new information."
                chat_history.append({"role": "user", "content": follow_up_llm_prompt_content})

                follow_up_llm_analysis = query_together_ai(chat_history) # Pass updated history
                print("\n--- AI Analysis of Follow-up Scan (from Together AI) ---")
                print(follow_up_llm_analysis)
                chat_history.append({"role": "assistant", "content": follow_up_llm_analysis}) # Add AI's response to history

            else:
                print("[-] Follow-up Nmap scan failed.")

        else:
            print("\nNo hosts found or no open ports in the Nmap scan results to analyze.")
    else:
        print("\nFailed to get Nmap scan output. Exiting.")

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

4.
5. **Save and Exit** `nano`**:**
    - Press `Ctrl + X`
    - Press `Y` (for Yes to save)
    - Press `Enter` (to confirm the filename `agent.py`)

4. Running the Agent

Now you're ready to run your agent from the command line.

1. **Ensure your virtual environment is active:**

```
source venv/bin/activate
```

2.
3. (You should see `(venv)` at the start of your prompt.)
4. Run the Python script with sudo:
   Nmap requires root privileges for certain scan types.

```
sudo python agent.py
```

5.
6. You will be prompted for your computer's password. Type it (characters won't appear) and press Enter.
7. **Interact with the Agent:**
    - When prompted, **enter the target IP address or range** (e.g., `192.168.1.1` for your router, or another device on your local network that you have permission to scan).
    - The agent will perform an initial comprehensive Nmap scan.
    - It will then display the parsed Nmap results and the AI's analysis and suggestions (from Together AI).
    - You will then enter the "Follow-up Scan Interface" prompt:

```
Enter a specific Nmap command (e.g., '-p 3306 --script mysql-info') for the target, or type 'quit' to exit:
```

    -
    - To run a suggested scan (e.g., for web vulnerabilities):
      Type the Nmap options (without nmap or the IP) and press Enter. For example:

```
-p 80,443,8080,8443 --script http-enum,http-title,http-headers,http-methods,http-sitemap-generator,http-vuln*
```

    -
    - The agent will execute this, show you the new results, and the AI will analyze them in context.
    - To quit the agent:
      Type quit and press Enter.

5. Exiting the Virtual Environment

When you are done working on the project and want to exit the virtual environment:

```
deactivate
```

Your terminal prompt will return to normal.

6. For GitHub

To add this to your GitHub repository:

1. Initialize a Git repository in your `ip_scanner_agent` directory (if you haven't already):

```
git init
```

2.
3. Create a `.gitignore` file to prevent unnecessary files (like your virtual environment) from being committed:

```
nano .gitignore
```

4.
5. Paste the following into `.gitignore` and save:

```
venv/
__pycache__/
*.pyc
.DS_Store
```

6.
7. Add your files to the staging area:

```
git add README.md SETUP_AND_RUN.md agent.py .gitignore
```

8.
9. Commit your changes:

```
git commit -m "Initial commit: Ethical IP Scanning Agent (CLI)"
```

10.
11. Set up your remote repository on GitHub and push your code (replace with your actual GitHub repository URL):

```
git remote add origin https://github.com/yourusername/your-repo-name.git
git branch -M main
git push -u origin main
```

12.