VULNERABILITY SCANNING TOOL

Key Components:

1. **Port Scanning**: Detect open ports on a target system.
2. **Service Identification**: Identify services running on open ports.
3. **Service Identification**: Identify services running on open ports.
4. **Vulnerability Database**: Use a database to check known vulnerabilities against detected services and versions.
5. **Report Generation**: Create reports with findings and recommendations.

IMPLEMENTATION PLAN

1. **Set Up the Environment**
   - Set up a virtual environment for the project and run the following command on the bash shell:

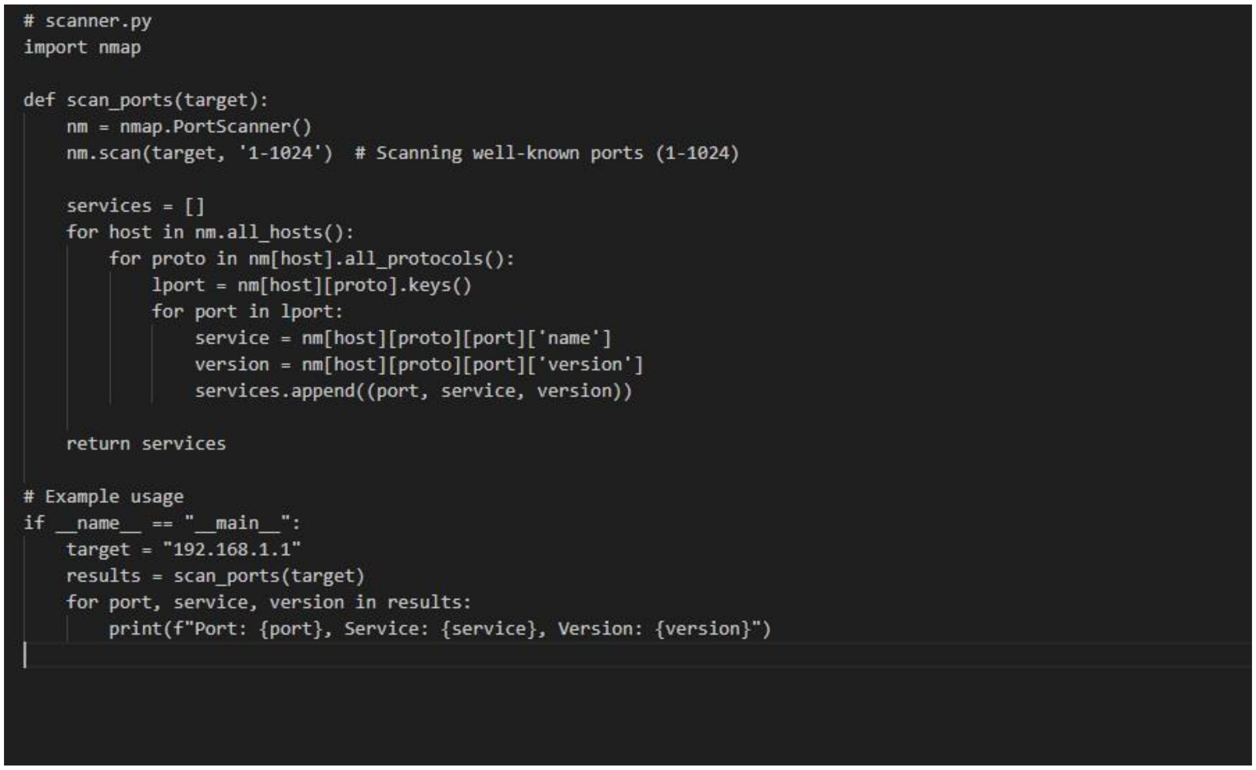
     ```
     mkdir vulnerability-scanner
     cd vulnerability-scanner
     python3 -m venv venv
     ```

   - Install necessary Python packages:

     ```
     pip install nmap requests
     ```

2. **Port Scanning and Service Identification**
   - Use the *python-nmap* library to perform port scanning and service identification

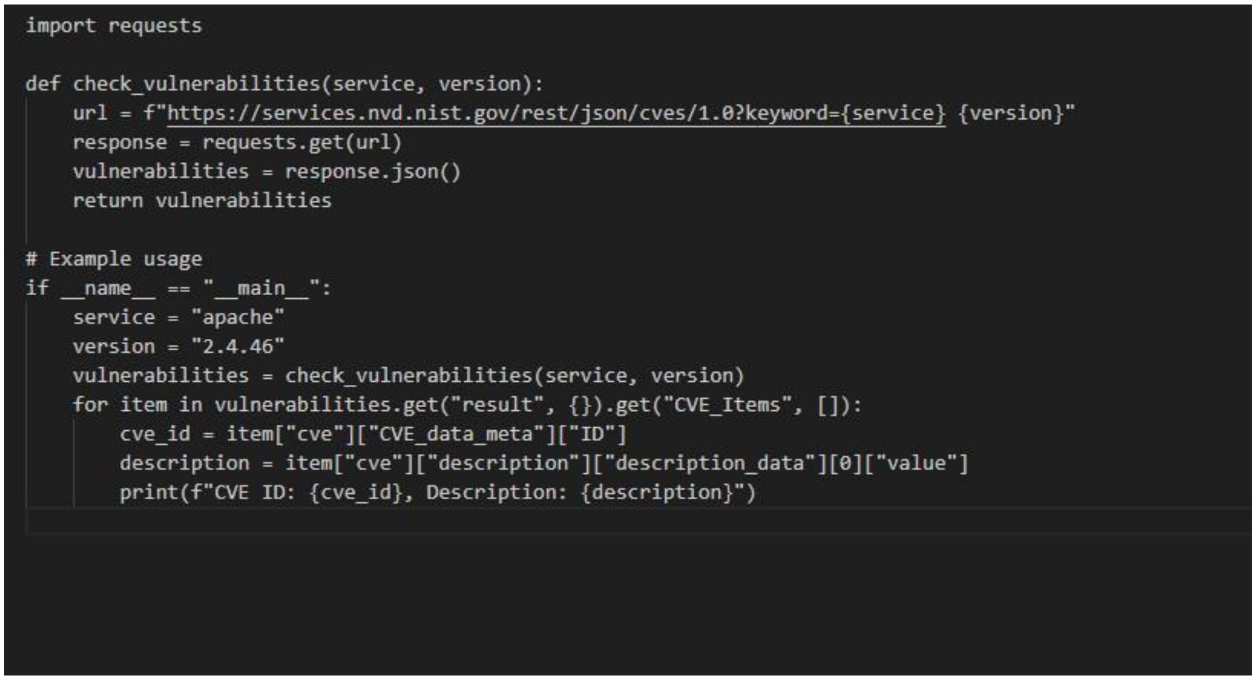
Figure 1. Python



```python
# scanner.py
import nmap

def scan_ports(target):
    nm = nmap.PortScanner()
    nm.scan(target, '1-1024')  # Scanning well-known ports (1-1024)

    services = []
    for host in nm.all_hosts():
        for proto in nm[host].all_protocols():
            lport = nm[host][proto].keys()
            for port in lport:
                service = nm[host][proto][port]['name']
                version = nm[host][proto][port]['version']
                services.append((port, service, version))

    return services

# Example usage
if __name__ == "__main__":
    target = "192.168.1.1"
    results = scan_ports(target)
    for port, service, version in results:
        print(f"Port: {port}, Service: {service}, Version: {version}")
```

3. **Vulnerability Database Integration:**
    - For simplicity, let's use the NVD (National Vulnerability Database) API to check for known vulnerabilities.

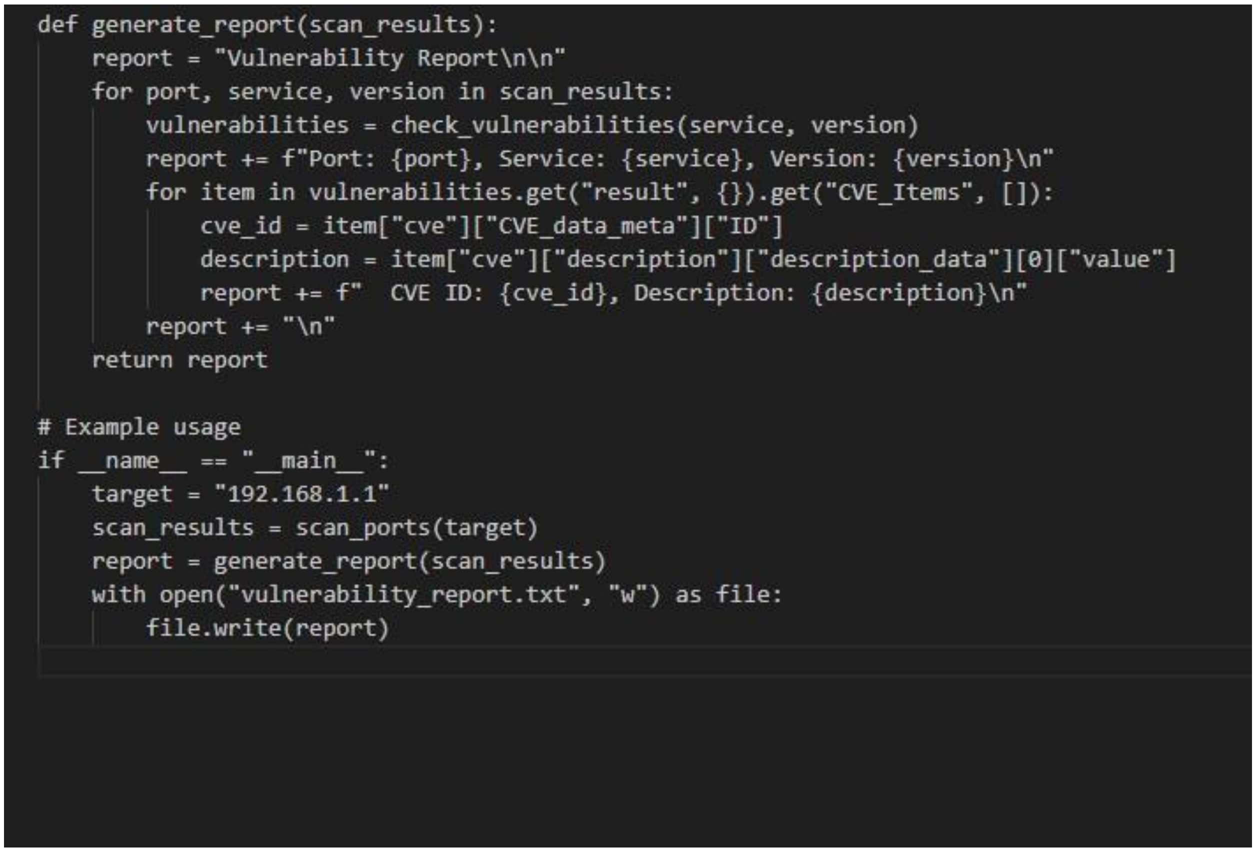
Figure 2. Python



```python
import requests

def check_vulnerabilities(service, version):
    url = f"https://services.nvd.nist.gov/rest/json/cves/1.0?keyword={service} {version}"
    response = requests.get(url)
    vulnerabilities = response.json()
    return vulnerabilities

# Example usage
if __name__ == "__main__":
    service = "apache"
    version = "2.4.46"
    vulnerabilities = check_vulnerabilities(service, version)
    for item in vulnerabilities.get("result", {}).get("CVE_Items", []):
        cve_id = item["cve"]["CVE_data_meta"]["ID"]
        description = item["cve"]["description"]["description_data"][0]["value"]
        print(f"CVE ID: {cve_id}, Description: {description}")
```

4. Generate Reports

- Create a report based on the identified vulnerabilities:



```python
def generate_report(scan_results):
    report = "Vulnerability Report\n\n"
    for port, service, version in scan_results:
        vulnerabilities = check_vulnerabilities(service, version)
        report += f"Port: {port}, Service: {service}, Version: {version}\n"
        for item in vulnerabilities.get("result", {}).get("CVE_Items", []):
            cve_id = item["cve"]["CVE_data_meta"]["ID"]
            description = item["cve"]["description"]["description_data"][0]["value"]
            report += f"  CVE ID: {cve_id}, Description: {description}\n"
        report += "\n"
    return report

# Example usage
if __name__ == "__main__":
    target = "192.168.1.1"
    scan_results = scan_ports(target)
    report = generate_report(scan_results)
    with open("vulnerability_report.txt", "w") as file:
        file.write(report)
```