Windows:

Step 1: Install Python and Dependencies

a. Install Python if it is not already installed
   i. Download from python.org
   ii. During installation, check "Add Python to PATH"
   iii. Verify the installation:

```
python --version
```

b. Install required Python libraries
   Open PowerShell or Command Prompt and run:

```
pip install python-nmap python-owasp-zap-v2.4 requests pandas jinja2
```

What we are installing:

1. Python-nmap: interacts with Nmap for network scanning
2. Zapv2: controls OWASP ZAP via API for web scanning
3. Requests: Fetches vulnerability details from external databases
4. Jinja2: Generates a report in HTML

Step 2: Scan Networks with Nmap

a. Install Nmap on Windows
   i. Download Nmap from nmap.org
   ii. Install Nmap and make sure that it is added to PATH
   iii. Verify the installation:

```
nmap --version
```

b. Create a Python script to scan networks

   See: netscan.py

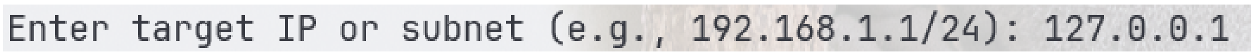
```
import nmap

def scan_network(target):
```

```python
    nm = nmap.PortScanner()
    nm.scan(hosts=target, arguments='-sS -p 1-1000')

    results = []
    for host in nm.all_hosts():
        for proto in nm[host].all_protocols():
            ports = nm[host][proto].keys()
            for port in ports:
                results.append({
                    "host": host,
                    "port": port,
                    "state": nm[host][proto][port]["state"]
                })

    return results

if __name__ == "__main__":
    target = input("Enter target IP or subnet: ")
    scan_results = scan_network(target)
    print(scan_results)
```

This script scans a target network using Nmap. It then returns open ports. Run the script (make sure you are in the correct directory):

```
python netscan.py
```

The script will prompt you to enter a target IP or local subnet.
Enter the target 127.0.0.1 to test the script.



```
Enter target IP or subnet (e.g., 192.168.1.1/24): 127.0.0.1
```

Step 3: Scan the Web with OWASP ZAP
Now, we can integrate OWASP ZAP to help scan for web vulnerabilities.

a. Install OWASP ZAP
    i. Download from [OWASP ZAP]
    ii. Install OWASP ZAP
    iii. Once installed, go to Tools > Options > API
        1. Make sure API is enabled. You can set your ow key if you wish

b. Create a Python script for web scanning

See: webscan.py

Make sure you connect your API key to the Python script.
OWASP ZAP should be started and running before running the script.

Run the script:

```
python webscan.py
```

When prompted, enter a test site such as: http://testphp.vulnweb.com

Note: users must have the ZAP proxy configured in "Local Proxy" mode for this to work, especially if they're running ZAP on a different machine.

Step 4: Integrate a Vulnerability Database

a. Fetch CVE data from an API
This code has been appended to *netscan.py* to check vulnerabilities

```
import requests

def get_cve_info(service_name):
    url =
f"https://services.nvd.nist.gov/rest/json/cves/1.0?keyword={service_n
ame}"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        cve_list = data.get("result", {}).get("CVE_Items", [])
        return [cve["cve"]["CVE_data_meta"]["ID"] for cve in
cve_list]
    return []

print(get_cve_info("Apache"))
```

This fetches CVEs related to scanned services (Apache, Open SSH). This information can be integrated into scan reports

Step 5: Create Reports
Results of scans can be output in an HTML report

a. Create a template in HTML
    i. See *template.html*
    ii. Generate the report in Python

        This code has been appended to *netscan.py* to save an HTML file with the scan results:

```
from jinja2 import Template

def generate_report(network_results, web_results):
    with open("template.html") as file:
        template = Template(file.read())

    report = template.render(network_results=network_results,
web_results=web_results)

    with open("scan_report.html", "w") as file:
        file.write(report)

    print("Report saved as scan_report.html")
```

Step 6: Run a Scan

Now, when the scan is run, you will get an automatically generated report on possible vulnerabilities (network and web). You can see an example of scan results in *scan_report.html*.

Start a scan by running *scan.py*

```
python scan.py
```

Windows Troubleshooting:

OWASP ZAP API not connecting properly

Issue: The OWASP ZAP script fails to connect to the ZAP API or hangs during scanning.
Solution: Ensure that OWASP ZAP is running and the API is enabled (Tools > Options > API). Also, verify the correct API key is set in the Python script. Double-check the proxy settings in the script match the ones configured in ZAP (default: http://127.0.0.1:8080).

Missing Dependencies after pip install

Issue: Some Python libraries may not be installed or accessible after running pip install.
Solution: Ensure that pip is installing the dependencies for the correct Python environment. If you are using a virtual environment, activate it before running the install command:

```
.\venv\Scripts\activate
```

If the issue persists, you may want to try upgrading pip:

```
pip install --upgrade pip
```

Incorrect Nmap scan output or no results

Issue: Nmap scan outputs no results or incorrect data.
Solution: Verify the target IP or subnet is correct and reachable. You can try running Nmap manually to check the results:

```
nmap -sS -p 1-1000 [target_ip]
```

Ensure the Nmap arguments in the script are appropriate for your scan (e.g., `-sS` for SYN scan)

CVE Fetching Issues

Issue: The CVE fetching function returns empty or no results.
Solution: Verify that the API URL is correctly formatted and that the service name (e.g., "Apache") is correct. You can test the API request manually using a browser or curl to check if it returns data. If there are connection issues, check your internet connection or firewall settings.