

Detailed Steps for Building a Web Application Vulnerability Scanner

Objective:

Develop a Python-based scanner that identifies common web vulnerabilities, including Cross-Site Scripting (XSS), **SQL Injection (SQLi)**, and **Cross-Site Request Forgery (CSRF)**, by crawling web pages, injecting attack payloads, analyzing responses, and presenting a Flask web interface for viewing scan results and reports.

Part 1: Setting Up the Project Directory

Step 1: Open Terminal and Create a Project Folder

Command: `mkdir ~/Vulnerability_scanner`

`cd ~/Vulnerability_scanner`

Step 2: Set Up a Virtual Environment (Optional but Recommended)

Command: `sudo apt install python3-venv -y`

`python3 -m venv venv`

`source venv/bin/activate`

Part 2: Install Required Python Packages

Command: `pip install requests beautifulsoup4 flask`

You can save this in a requirements.txt file:

`echo -e "requests\nbeautifulsoup4\nflask" > requirements.txt`

`pip install -r requirements.txt`

Part 3: Create Project Files

Step 1: File Structure

Vulnerability_scanner/

├── app.py

├── scanner.py

├── templates/

│ ├── index.html

├── static/

│ ├── style.css (optional)

└── logs/

└── scan_results.json

Step 2: Create scanner.py

Command: nano scanner.py

Paste this **sample scanning logic**:

```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urlparse
import re

xss_payloads = ["<script>alert(1)</script>", "\" onerror=\"alert(1)\", \"<img src=x onerror=alert(1)>"]
sqli_payloads = ["' OR '1'='1", "' OR 1=1 --", "'; DROP TABLE users; --"]

scanned_urls = set()

def crawl_and_scan(base_url):
    found_vulns = []
    to_visit = [base_url]

    while to_visit:
        url = to_visit.pop()
        if url in scanned_urls or not url.startswith(base_url):
            continue
        try:
            resp = requests.get(url, timeout=5)
            scanned_urls.add(url)

            soup = BeautifulSoup(resp.text, 'html.parser')
            links = [urljoin(url, a.get("href")) for a in soup.find_all("a", href=True)]

            to_visit.extend(links)

            forms = soup.find_all("form")
            for form in forms:
                action = urljoin(url, form.get("action"))
                method = form.get("method", "get").lower()
                inputs = form.find_all("input")

                for payload in xss_payloads + sqli_payloads:
                    data = {}
                    for input in inputs:
                        name = input.get("name")
                        if name:
                            data[name] = payload
```

for input in inputs:

 name = input.get("name")

 if name:

 data[name] = payload

if method == "post":

 res = requests.post(action, data=data)

else:

 res = requests.get(action, params=data)

if payload in res.text:

 vuln_type = "XSS" if payload in xss_payloads else "SQL Injection"

 found_vulns.append({

 "url": action,

 "payload": payload,

 "type": vuln_type,

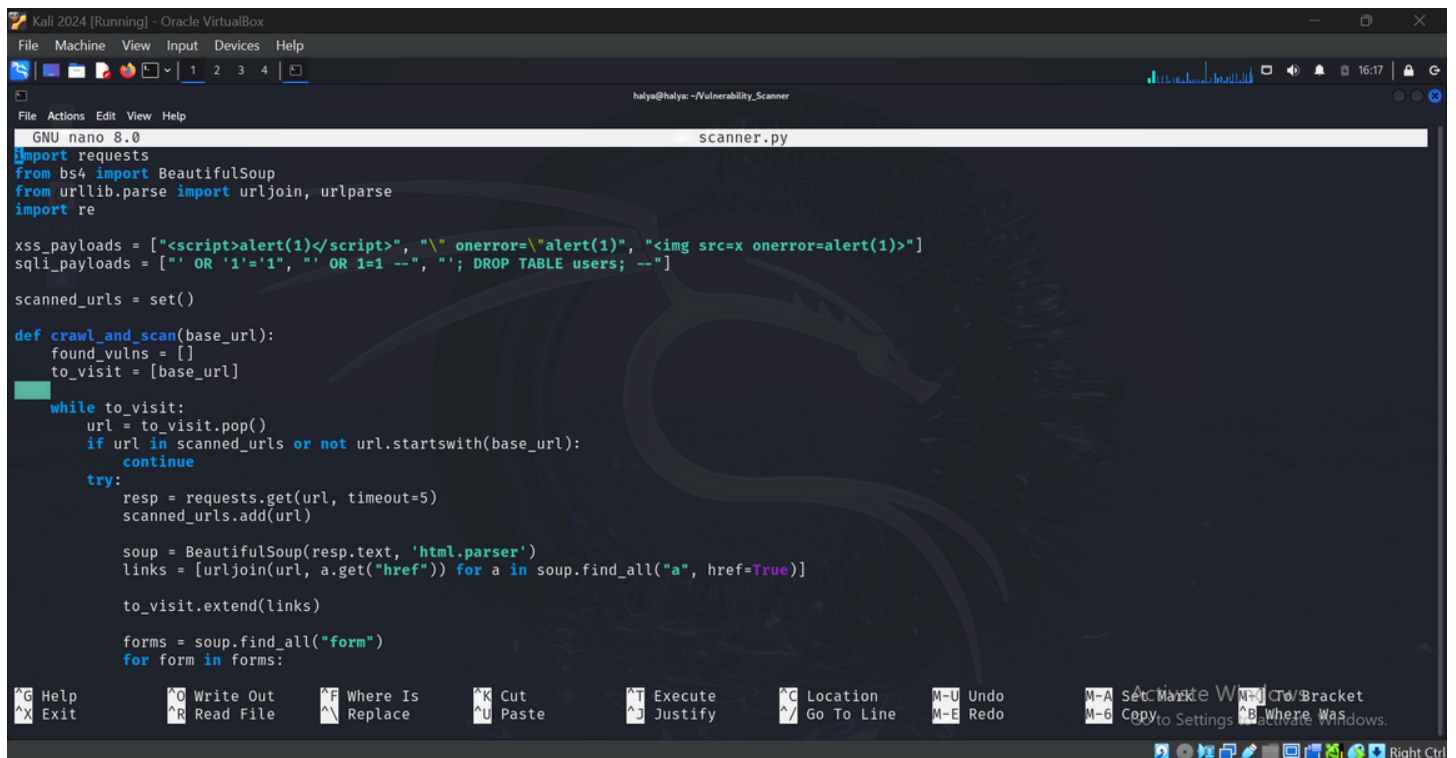
 "severity": "High" if vuln_type == "XSS" else "Medium"

 })

except Exception as e:

 continue

return found_vulns



```
Kali 2024 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
halys@halys: ~/Vulnerability_Scanner
File Actions Edit View Help
GNU nano 8.0 scanner.py
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urlparse
import re

xss_payloads = ["<script>alert(1)</script>", "\\" onerror=\"alert(1)\", "<img src=x onerror=alert(1)>"]
sql_payloads = ["' OR '1'='1", "' OR 1=1 --", "'; DROP TABLE users; --"]

scanned_urls = set()

def crawl_and_scan(base_url):
    found_vulns = []
    to_visit = [base_url]

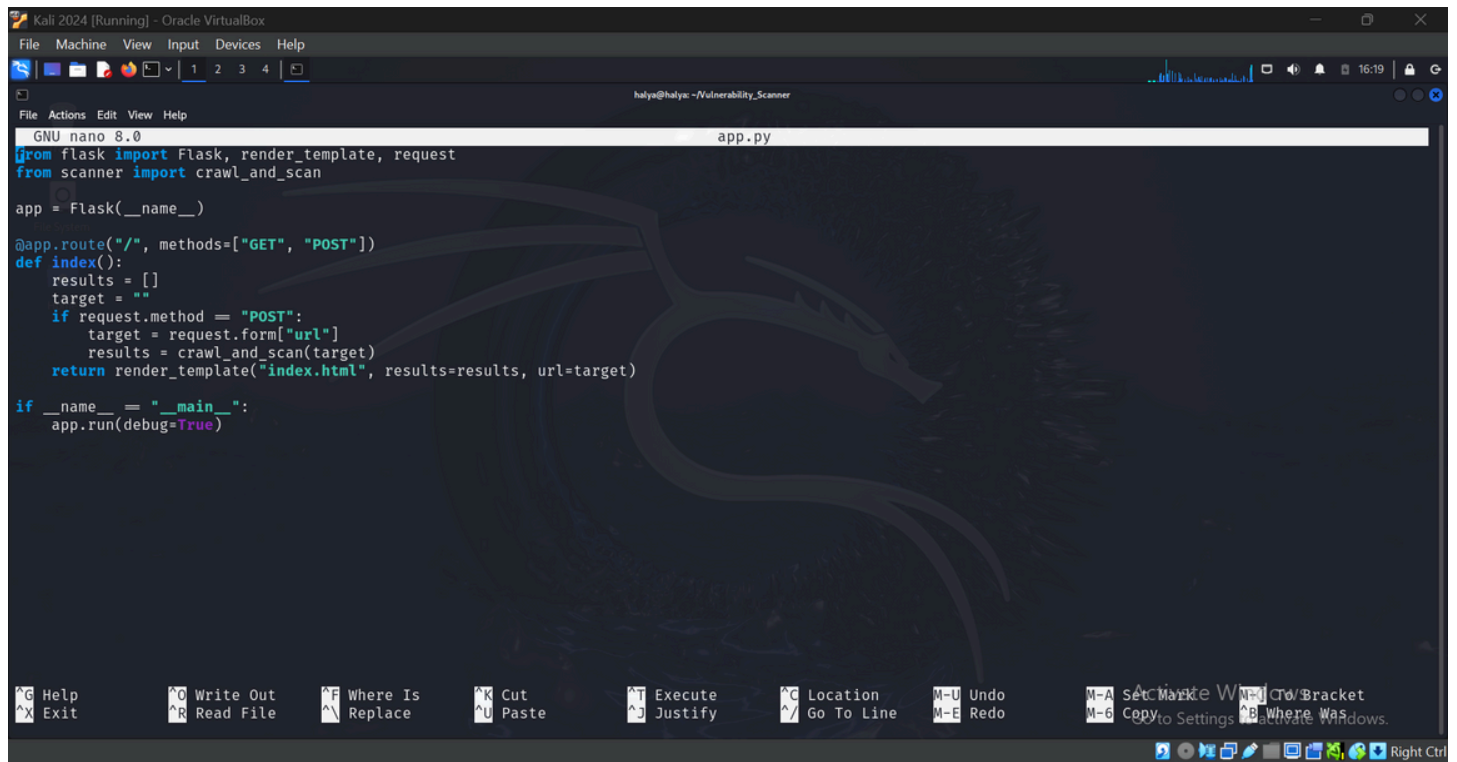
    while to_visit:
        url = to_visit.pop()
        if url in scanned_urls or not url.startswith(base_url):
            continue
        try:
            resp = requests.get(url, timeout=5)
            scanned_urls.add(url)

            soup = BeautifulSoup(resp.text, 'html.parser')
            links = [urljoin(url, a.get("href")) for a in soup.find_all("a", href=True)]
            to_visit.extend(links)

            forms = soup.find_all("form")
            for form in forms:
```

Step 3: Create app.py (Flask Web Interface)

Command: nano app.py



```
GNU nano 8.0 app.py
from flask import Flask, render_template, request
from scanner import crawl_and_scan

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    results = []
    target = ""
    if request.method == "POST":
        target = request.form["url"]
        results = crawl_and_scan(target)
    return render_template("index.html", results=results, url=target)

if __name__ == "__main__":
    app.run(debug=True)
```

Step 4: Create HTML Template

Command: mkdir templates
nano templates/index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Vulnerability Scanner</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      padding: 40px;
      background-color: #f7f9fc;
      color: #333;
    }

    h1 {
      color: #1e90ff;
    }

    form {
      margin-bottom: 30px;
    }

    input[type="text"] {
```

```
padding: 10px;
width: 400px;
border: 1px solid #ccc;
border-radius: 5px;
}
```

```
button {
padding: 10px 20px;
border: none;
background-color: #1e90ff;
color: white;
border-radius: 5px;
cursor: pointer;
}
```

```
button:hover {
background-color: #0d6efd;
}
```

```
table {
width: 100%;
border-collapse: collapse;
margin-top: 20px;
background-color: white;
border-radius: 5px;
overflow: hidden;
box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
```

```
th, td {
padding: 12px;
text-align: left;
border-bottom: 1px solid #eee;
}
```

```
th {
background-color: #f2f2f2;
color: #333;
}
```

```

.severity-High {
  color: white;
  background-color: #e74c3c; /* Red */
  font-weight: bold;
}

.severity-Medium {
  color: black;
  background-color: #f39c12; /* Orange */
  font-weight: bold;
}

.severity-Low {
  color: black;
  background-color: #f0e68c; /* Light Yellow */
}
</style>
</head>
<body>
  <h1> Automated Web App Vulnerability Scanner</h1>
  <form method="POST">
    <input type="text" name="url" placeholder="https://example.com" required>
    <button type="submit">Start Scan</button>
  </form>

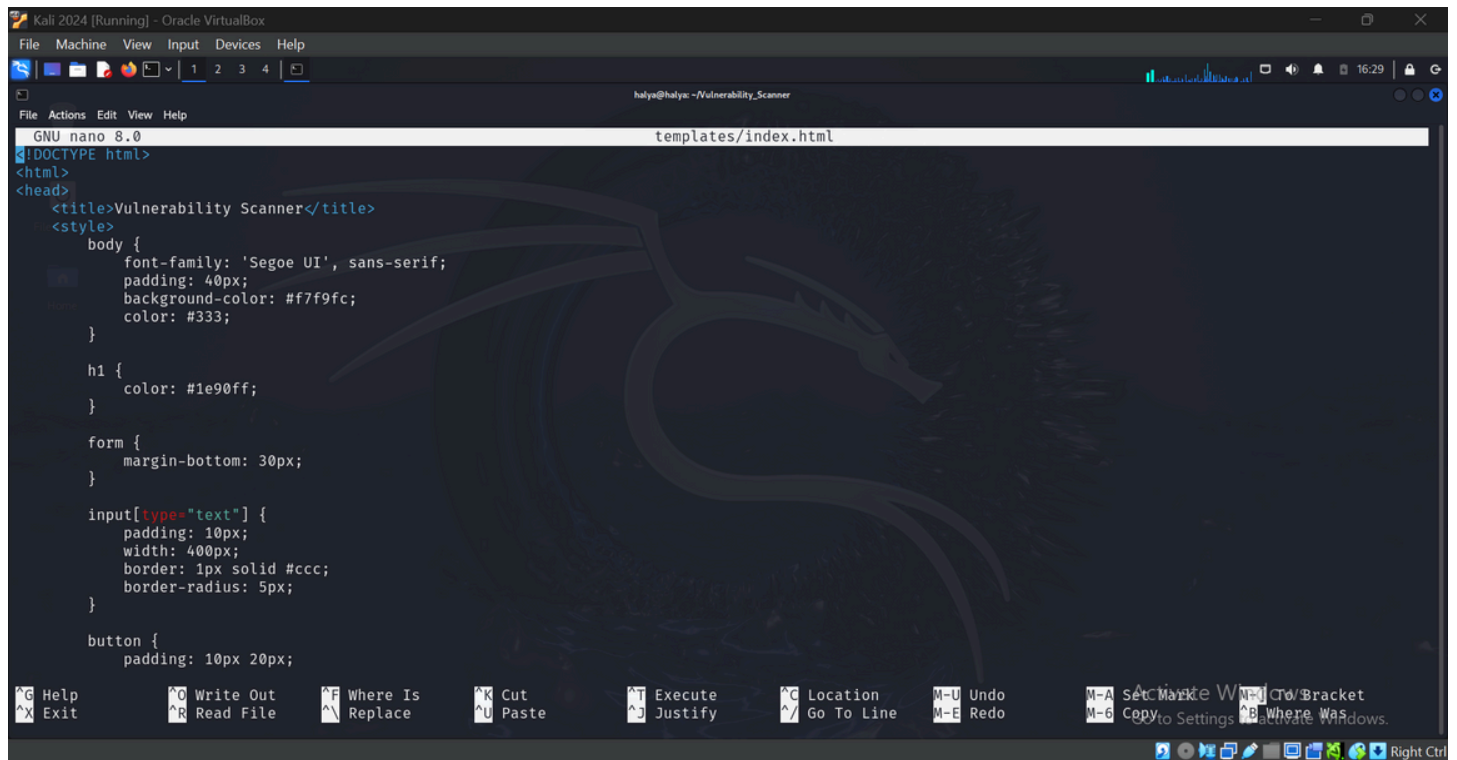
  {% if results %}
  <h2>Scan Results for <span style="color: #1e90ff;">{{ url }}</span></h2>
  <table>
    <tr>
      <th>Type</th>
      <th>Payload</th>
      <th>Target URL</th>
      <th>Severity</th>
    </tr>
    {% for r in results %}
    <tr>
      <td>{{ r.type }}</td>
      <td><code>{{ r.payload }}</code></td>
      <td><a href="{{ r.url }}" target="_blank">{{ r.url }}</a></td>
      <td class="severity-{{ r.severity }}">{{ r.severity }}</td>
    </tr>
  </table>
  </body>
</html>

```

```

</tr>
{% endfor %}
</table>
{% endif %}
</body>
</html>

```



```

GNU nano 8.0 templates/index.html
<!DOCTYPE html>
<html>
<head>
<title>Vulnerability Scanner</title>
<style>
body {
font-family: 'Segoe UI', sans-serif;
padding: 40px;
background-color: #f7f9fc;
color: #333;
}

h1 {
color: #1e90ff;
}

form {
margin-bottom: 30px;
}

input[type="text"] {
padding: 10px;
width: 400px;
border: 1px solid #ccc;
border-radius: 5px;
}

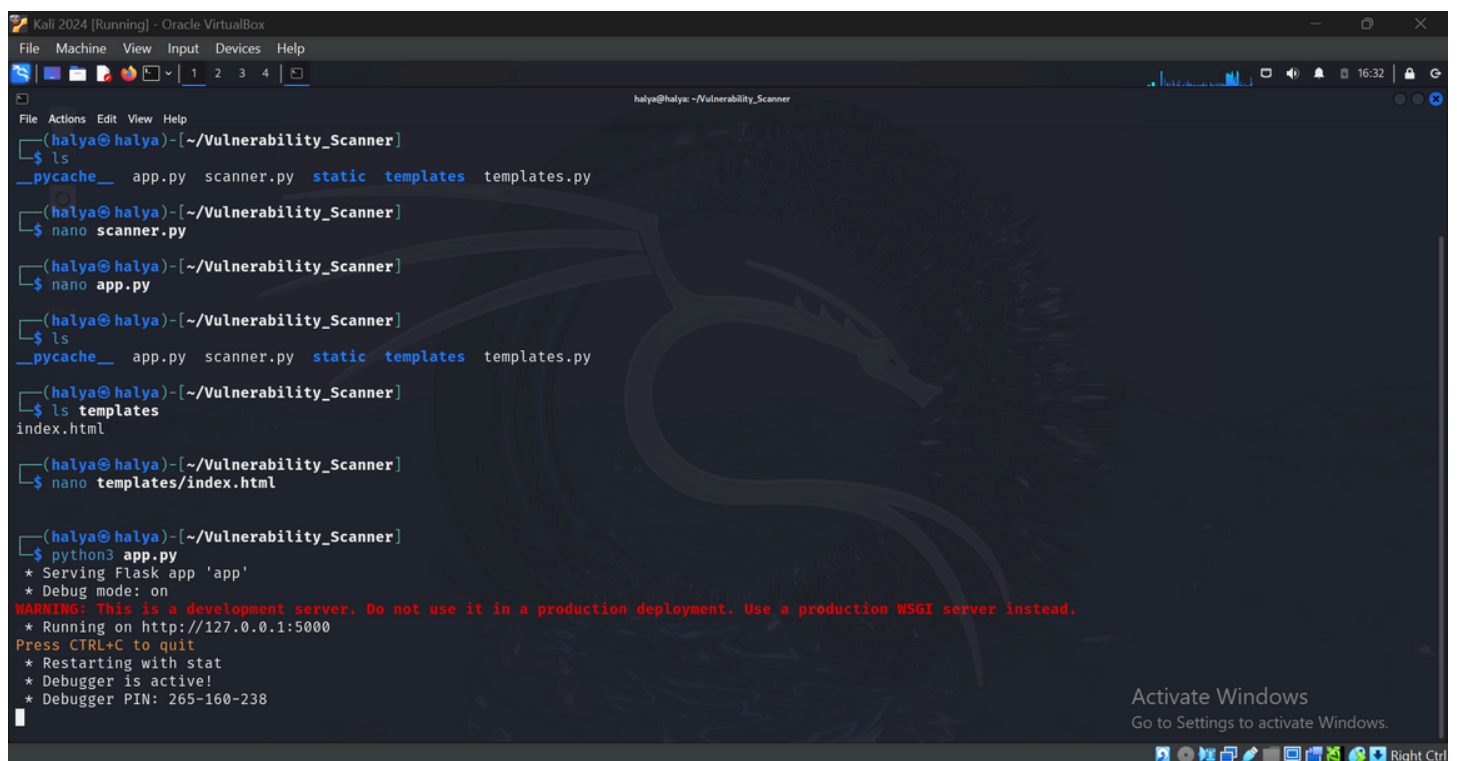
button {
padding: 10px 20px;

```

Part 4: Run the Application

Run the Flask App

Command: python3 app.py



```

(halya@halya)-[~/Vulnerability_Scanner]
$ ls
__pycache__  app.py  scanner.py  static  templates  templates.py

(halya@halya)-[~/Vulnerability_Scanner]
$ nano scanner.py

(halya@halya)-[~/Vulnerability_Scanner]
$ nano app.py

(halya@halya)-[~/Vulnerability_Scanner]
$ ls
__pycache__  app.py  scanner.py  static  templates  templates.py

(halya@halya)-[~/Vulnerability_Scanner]
$ ls templates
index.html

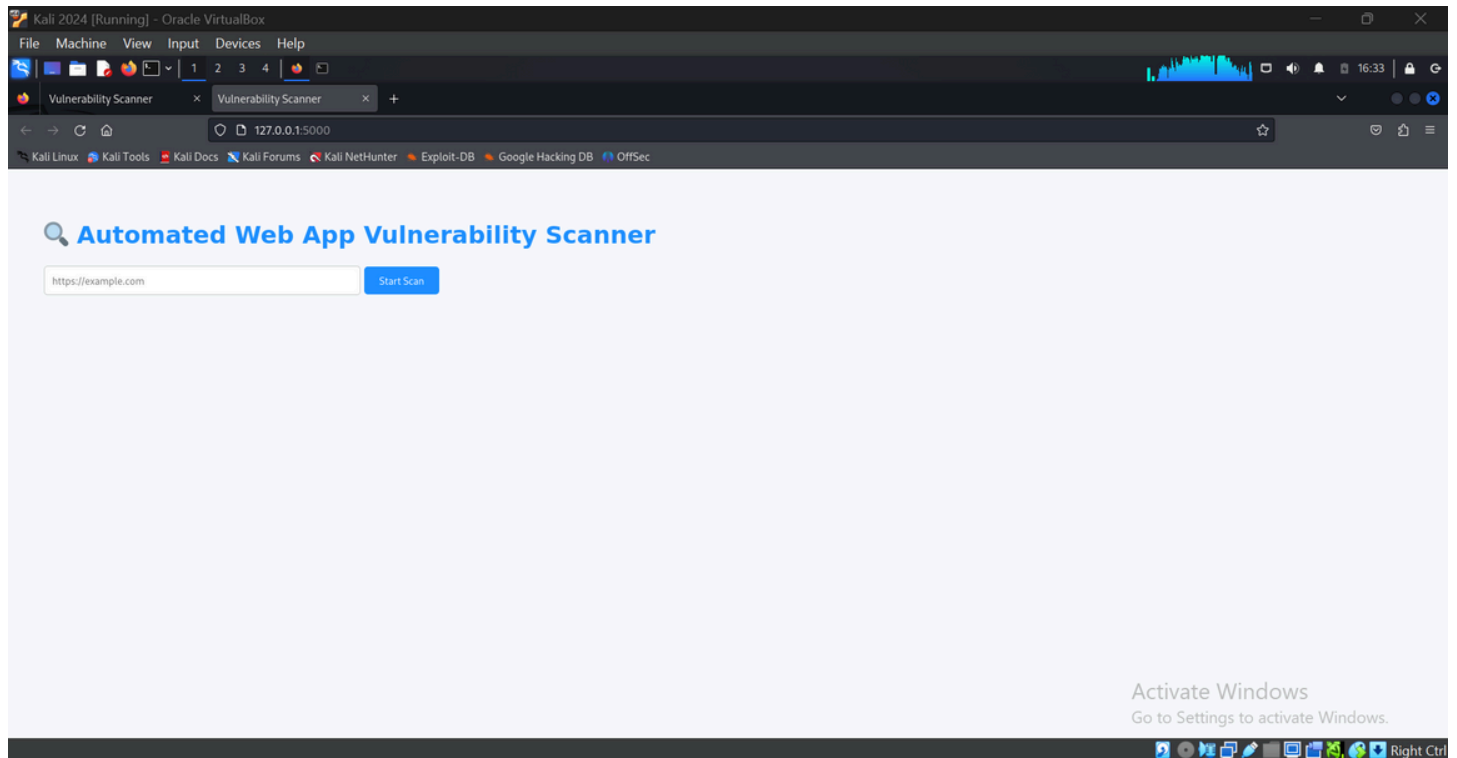
(halya@halya)-[~/Vulnerability_Scanner]
$ nano templates/index.html

(halya@halya)-[~/Vulnerability_Scanner]
$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 265-160-238

```

Open in Browser

Go to: <http://127.0.0.1:5000>



Part 5: Save and Organize Scripts on Kali Linux

1. Save the folder Vulnerability_scanner in your home directory: **~/Vulnerability_Scanner**
2. To run anytime:

Command: `cd ~/Vulnerability_scanner`

`source venv/bin/activate`

`python3 app.py`

```
(halya@halya)-[~]
$ cd Vulnerability_Scanner
source venv/bin/activate
python3 app.py

source: no such file or directory: venv/bin/activate
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 265-160-238
127.0.0.1 - - [20/Jun/2025 16:54:47] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Jun/2025 16:54:48] "GET /favicon.ico HTTP/1.1" 404 -
^C
```

To edit files:

- Use nano or open VSCode:

Command: `code ~/Vulnerability_scanner`


```
Kali 2024 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4
halya@halya: ~/Vulnerability_Scanner
(halya@halya)-[~/Vulnerability_Scanner]
$ code ~/Vulnerability_Scanner
Command 'code' not found, but can be installed with:
sudo apt install code-oss
Do you want to install it? (N/y)y
sudo apt install code-oss
The following packages were automatically installed and are no longer required:
fonts-liberation2 libboost-thread1.83.0 libgfrpc0 libhdfs-hl-100t64 libpoppler134 libsuperlu6 samba-vfs-modules
ibverbs-providers libcephfs2 libgfrpc0 libhdfs-hl-100t64 libpython3.11-dev python3.11
libarmadillo12 libgdal34t64 libglusterfs0 liblbfgsb0 librados2 python3.11-dev
libboost-iostreams1.83.0 libgapi0 libhdfs-103-1t64 libnetcdf19t64 librdmacm1t64 python3.11-minimal
Use 'sudo apt autoremove' to remove them.

Installing:
code-oss

Installing dependencies:
libnode115 node-balanced-match node-cjs-module-lexer node-minimatch node-xtend nodejs-doc
node-acorn node-brace-expansion node-corepack node-undici nodejs

Suggested packages:
npm

Summary:
Upgrading: 0, Installing: 12, Removing: 0, Not Upgrading: 2009
Download size: 327 MB
Space needed: 2694 MB / 6584 MB available

Continue? [Y/n] y
Ign:1 http://http.kali.org/kali kali-rolling/main amd64 node-xtend all 4.0.2-3
Ign:2 http://http.kali.org/kali kali-rolling/main amd64 node-acorn all 8.8.1+ds+~cs25.17.7-2
Ign:3 http://http.kali.org/kali kali-rolling/main amd64 node-cjs-module-lexer all 1.2.3+dfsg-1
Ign:4 http://http.kali.org/kali kali-rolling/main amd64 node-balanced-match all 2.0.0-1

Activate Windows
Go to Settings to activate Windows.
```

```
Kali 2024 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4
(halya@halya)-[~/Vulnerability_Scanner]
$ code ~/Vulnerability_Scanner
(Message from Kali developers)
code is not the binary you may be expecting.
You are looking for \"code-oss\"
Starting code-oss for you ...

Warning: 'unity-launch' is not in the list of known options, bu
[main 2025-06-20T16:17:54.379Z] update#setState disabled
[main 2025-06-20T16:17:54.419Z] update#ctor - updates are disab
[42428:0620/214755.660780:ERROR:command_buffer_proxy_impl.cc(13
[42346:0620/214815.397229:ERROR:atom_cache.cc(229)] Add chromiu
[42346:0620/214817.185167:ERROR:atom_cache.cc(229)] Add _NET_RE
[main 2025-06-20T16:18:32.417Z] Extension host with pid 42543 e

(halya@halya)-[~/Vulnerability_Scanner]
$ code ~/Vulnerability_Scanner
(Message from Kali developers)
code is not the binary you may be expecting.
You are looking for \"code-oss\"
Starting code-oss for you ...

Warning: 'unity-launch' is not in the list of known options, bu
[main 2025-06-20T16:18:39.365Z] update#setState disabled
[main 2025-06-20T16:18:39.369Z] update#ctor - updates are disab
[main 2025-06-20T16:19:02.579Z] Extension host with pid 43068 e

(halya@halya)-[~/Vulnerability_Scanner]
$ code ~/Vulnerability_Scanner
(Message from Kali developers)
code is not the binary you may be expecting.
You are looking for \"code-oss\"
Starting code-oss for you ...

Warning: 'unity-launch' is not in the list of known options, bu
[main 2025-06-20T16:20:36.434Z] update#setState disabled
[main 2025-06-20T16:20:36.451Z] update#ctor - updates are disab
[43964:0620/215045.113971:ERROR:atom_cache.cc(229)] Add chromium/from-j

EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.

Code - OSS
Editing evolved

Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...

Recent
You have no recent folders, open a folder to start.

Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals

Activate Windows
Go to Settings to activate Windows.
```

Conclusion – Web Application Vulnerability Scanner

The **Web Application Vulnerability Scanner** project effectively demonstrates how to detect common web vulnerabilities such as **Cross-Site Scripting (XSS)** and **SQL Injection (SQLi)** using Python, web crawling, payload injection, and response analysis. By automating the identification of security flaws, this tool aids in reducing the manual effort required in penetration testing.

The addition of a user-friendly **Flask web interface** enhances usability, allowing users to input target URLs, launch scans, and view detailed results, including evidence and severity. The

scanner adheres to security best practices outlined in the **OWASP Top 10**, providing a solid foundation for more advanced security tools.

This project is ideal for learners and security professionals aiming to:

- Understand basic web vulnerabilities.
- Gain hands-on experience with ethical hacking.
- Practice secure coding and testing in a controlled environment.

Future improvements could include:

- CSRF token analysis,
- Authentication handling,
- Multithreading for better performance,
- Enhanced report generation and UI.

By developing and running this scanner on **Kali Linux**, users gain real-world exposure to offensive security techniques while reinforcing responsible ethical practices in cybersecurity.