A Simple Web Application Vulnerability Scanner

Vulnerability scanner: A vulnerability scanner is an automated tool that detects security weaknesses, such as misconfigurations and unpatched software, in a network, system, or application before attackers can exploit them. These scanners perform a scan to identify flaws and often provide a detailed report that prioritizes the risks and suggests remediation steps.

Abstract

With the rapid expansion of web applications and digital infrastructure, security vulnerabilities have become a critical concern for organizations and individuals alike. This project focuses on the design and development of an automated Vulnerability Scanner capable of identifying common security flaws in web applications. The scanner systematically analyzes target systems to detect vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), insecure configurations, and other threats aligned with the OWASP Top 10 standards.

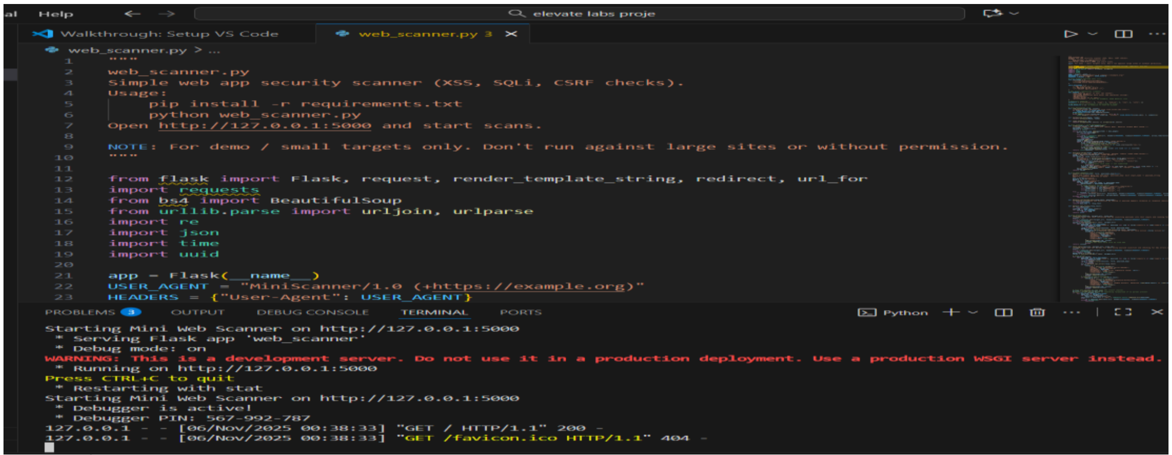
The proposed system utilizes techniques such as URL crawling, input field detection, payload injection, and response analysis to evaluate potential security risks. It provides detailed reports highlighting detected vulnerabilities along with their severity levels and remediation suggestions. By automating the vulnerability assessment process, the scanner reduces manual effort, increases efficiency, and enhances the overall security posture of web applications. This tool serves as a practical solution for developers and security professionals to proactively identify and mitigate security risks before deployment.

Tools used: visual studio code, Microsoft edge browser, python 3.12.10

Packages used: requests, BeautifulSoup, OWASP top 10 checklist, Flask

Process steps:

- Install all required packages.
- Run the code.
- Open the browser and run 127.0.0.1:5000
- Select the target url and run the scanner
- Monitor the results.



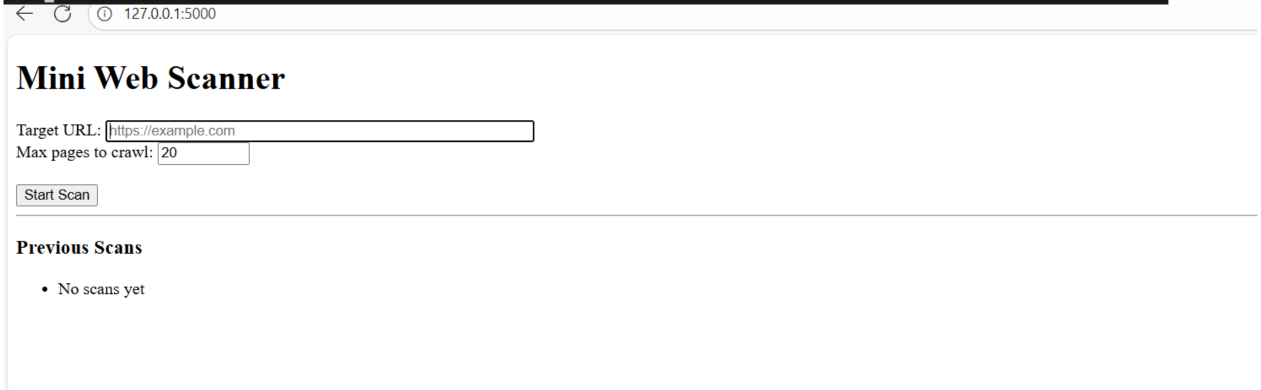
```
"""
web_scanner.py
Simple web app security scanner (XSS, SQLi, CSRF checks).
Usage:
    pip install -r requirements.txt
    python web_scanner.py
Open http://127.0.0.1:5000 and start scans.

NOTE: For demo / small targets only. Don't run against large sites or without permission.
"""

from flask import Flask, request, render_template_string, redirect, url_for
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urlparse
import re
import json
import time
import uuid

app = Flask(__name__)
USER_AGENT = "MiniScanner/1.0 (+https://example.org)"
HEADERS = {"User-Agent": USER_AGENT}
```

```
Starting Mini Web Scanner on http://127.0.0.1:5000
 * Serving Flask app 'web_scanner'
 * 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
Starting Mini Web Scanner on http://127.0.0.1:5000
 * Debugger is active!
 * Debugger PIN: 567-992-787
127.0.0.1 - - [06/Nov/2025 00:38:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2025 00:38:33] "GET /favicon.ico HTTP/1.1" 404 -
```



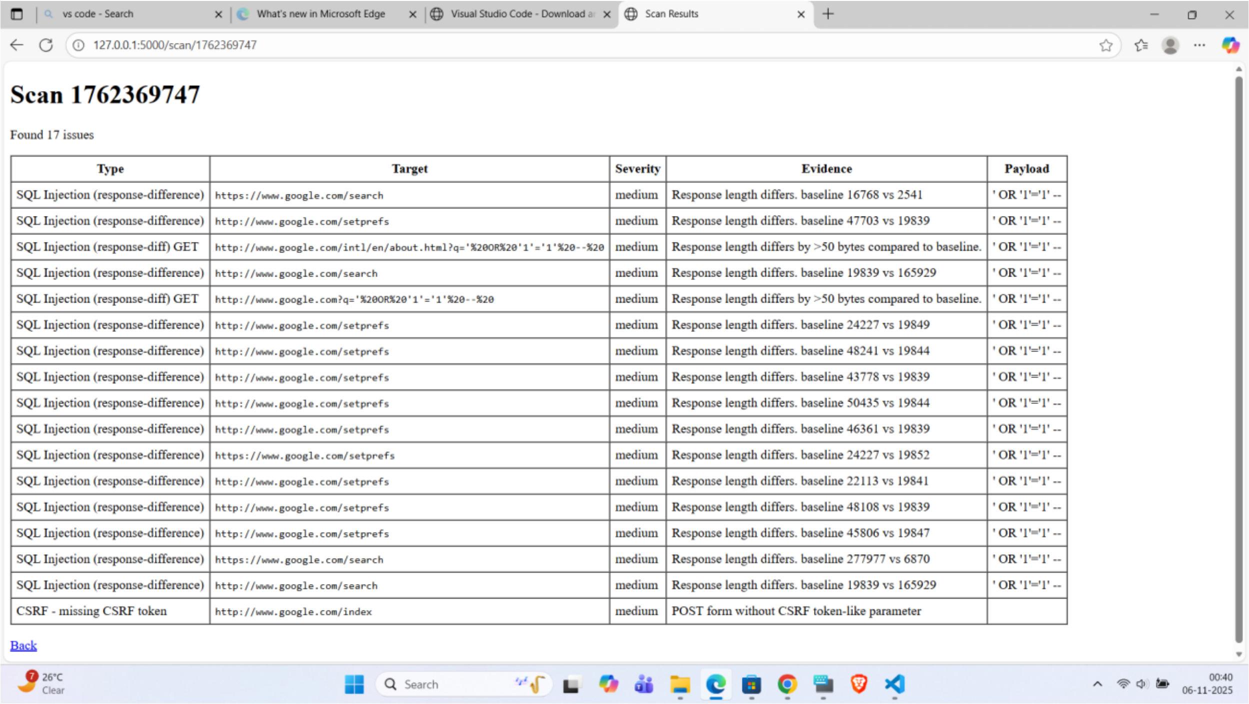
Mini Web Scanner

Target URL: https://example.com

Max pages to crawl: 20

Previous Scans

- No scans yet


vs code - Search
What's new in Microsoft Edge
Visual Studio Code - Download a
Scan Results
127.0.0.1:5000/scan/1762369747
26°C
Clear
Search
00:40
06-11-2025

Scan 1762369747

Found 17 issues

Type	Target	Severity	Evidence	Payload
SQL Injection (response-difference)	`https://www.google.com/search`	medium	Response length differs. baseline 16768 vs 2541	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 47703 vs 19839	' OR '1'='1' --
SQL Injection (response-diff) GET	`http://www.google.com/intl/en/about.html?q='%20OR%20'1'='1'%20--%20`	medium	Response length differs by >50 bytes compared to baseline.	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/search`	medium	Response length differs. baseline 19839 vs 165929	' OR '1'='1' --
SQL Injection (response-diff) GET	`http://www.google.com?q='%20OR%20'1'='1'%20--%20`	medium	Response length differs by >50 bytes compared to baseline.	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 24227 vs 19849	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 48241 vs 19844	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 43778 vs 19839	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 50435 vs 19844	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 46361 vs 19839	' OR '1'='1' --
SQL Injection (response-difference)	`https://www.google.com/setprefs`	medium	Response length differs. baseline 24227 vs 19852	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 22113 vs 19841	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 48108 vs 19839	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/setprefs`	medium	Response length differs. baseline 45806 vs 19847	' OR '1'='1' --
SQL Injection (response-difference)	`https://www.google.com/search`	medium	Response length differs. baseline 277977 vs 6870	' OR '1'='1' --
SQL Injection (response-difference)	`http://www.google.com/search`	medium	Response length differs. baseline 19839 vs 165929	' OR '1'='1' --
CSRF - missing CSRF token	`http://www.google.com/index`	medium	POST form without CSRF token-like parameter	

Back