

# WebSec\_AutoGuard: A Basic Vulnerability Scanner Using Python

## Project Overview

**WebSec\_AutoGuard** is a Python-based web vulnerability scanning tool developed to identify client-side security flaws, primarily focusing on detecting **Cross-Site Scripting (XSS)** vulnerabilities in HTML forms across webpages. The tool is structured with modular components—scanner, parser, detector, and reporter—ensuring clarity, maintainability, and extensibility.

In addition to command-line usability, the tool features a user-friendly **Flask-based WebUI** that allows non-technical users to interact with the scanner seamlessly. It produces professional **PDF-based vulnerability reports** and maintains execution logs, making it suitable for both academic and lightweight real-world security assessments. The complete project, including documentation and a demonstration video, has been published on GitHub for open-source access and version control.

## Objectives

The primary objective of **WebSec\_AutoGuard** is to develop a modular, efficient, and user-friendly tool that identifies security vulnerabilities—especially **Cross-Site Scripting (XSS)**—in web forms. The tool is intended for academic, educational, and lightweight security auditing purposes.

- To build a Python-based web vulnerability scanner with a primary focus on XSS detection.
- To extract and analyze HTML forms from target websites using automated parsing.
- To simulate malicious payload submissions and detect insecure input handling.
- To generate comprehensive and downloadable **PDF-based security reports**.
- To create a Flask-powered **Web User Interface** that allows non-technical users to initiate and review scans easily.
- To log scanning activity for transparency and future debugging.
- To maintain clean documentation and publish the project on **GitHub** with a proper README, demo, and licensing.

## TOOL ARCHITECTURE

```
WebSec_AutoGuard/
├── scanner.py          # Performs HTTP scanning and fetches HTML
├── form_parser.py      # Parses HTML and extracts forms
├── vulnerability_detector.py # Checks forms for XSS vulnerabilities
├── report_generator.py  # Creates vulnerability reports in PDF format
├── utils.py            # Reusable utility functions
├── config.json         # Configuration file
├── templates/         # HTML templates for WebUI
│   ├── index.html     # Dashboard UI
│   └── result.html     # Results display UI
├── static/            # Optional CSS or JS files for WebUI
├── app.py / Dashboard.py # Flask-based Web interface
├── README.md, LICENSE  # Documentation and licensing
├── requirements.txt    # Required Python libraries
├── demo.mp4           # Demo video showing functionality
├── WebSec_Report.pdf   # Sample generated report
└── websec_autoguard.log # Logs
```

## Key Features

- **Modular Codebase:** The project is structured into independent, logically separated modules for scanning, parsing, detection, reporting, and UI. This ensures maintainability and scalability.
- **XSS Detection Engine:** Uses simulated input attacks to test for potential **Cross-Site Scripting (XSS)** vulnerabilities in form elements, a common and dangerous web security flaw.
- **PDF Report Generation:** After scanning, the tool compiles findings into a clean, downloadable **PDF report** that summarizes each vulnerability with its risk level.
- **User-Friendly Web Dashboard:** Built with Flask and Bootstrap, the UI allows users to easily input a URL, select scanning options, and view results without using the terminal.
- **Logging Mechanism:** Automatically logs each scanning session, errors, and findings into websec\_autoguard.log, useful for debugging and record-keeping.
- **CLI and Web Support:** Offers flexibility by supporting both **Command-Line Interface (CLI)** for advanced users and **Web Interface** for non-technical users.
- **Lightweight and Customizable:** Uses minimal third-party packages and allows developers to expand detection logic for other vulnerabilities.

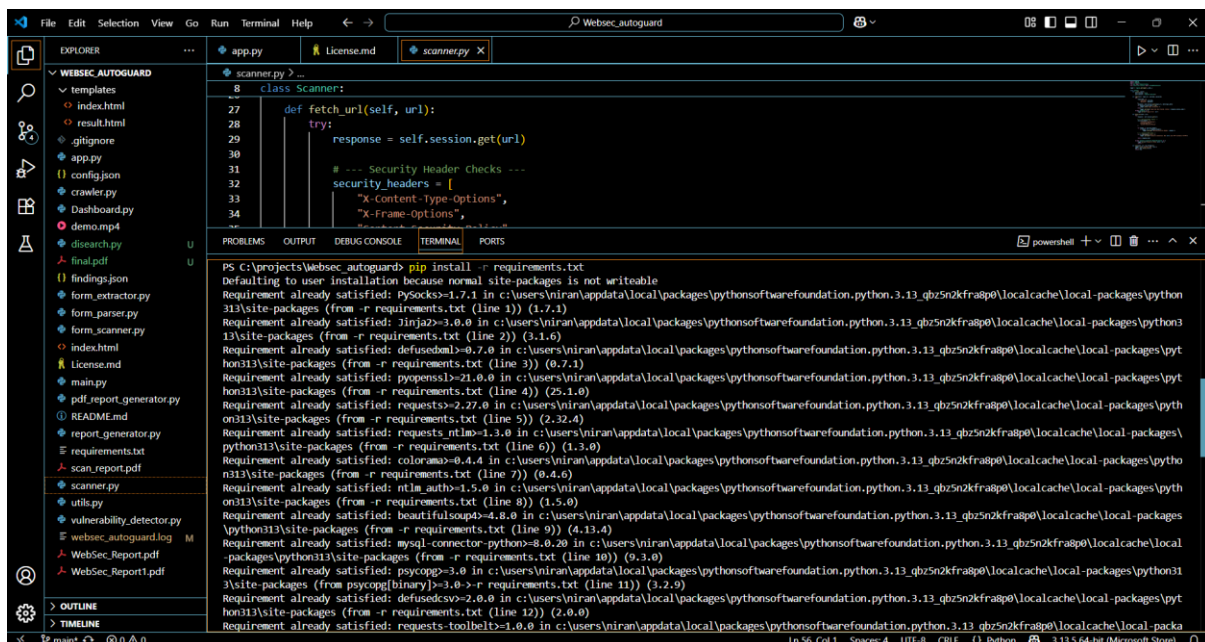
## Technologies Used

Technology	Purpose
Python	Core language for building scanner logic, detection engine, and utilities.
Flask	Lightweight web framework for building the interactive WebUI.
BeautifulSoup	HTML parsing and form element extraction from webpages.
requests	Sending HTTP requests to fetch HTML content from target URLs.
fpdf	Generating structured and printable PDF reports.
Git & GitHub	Version control and publishing the complete project online.
Bootstrap	Styling and responsive design for the web interface.

## Step-by-Step Implementation

### Step 1: Setup & Environment Configuration

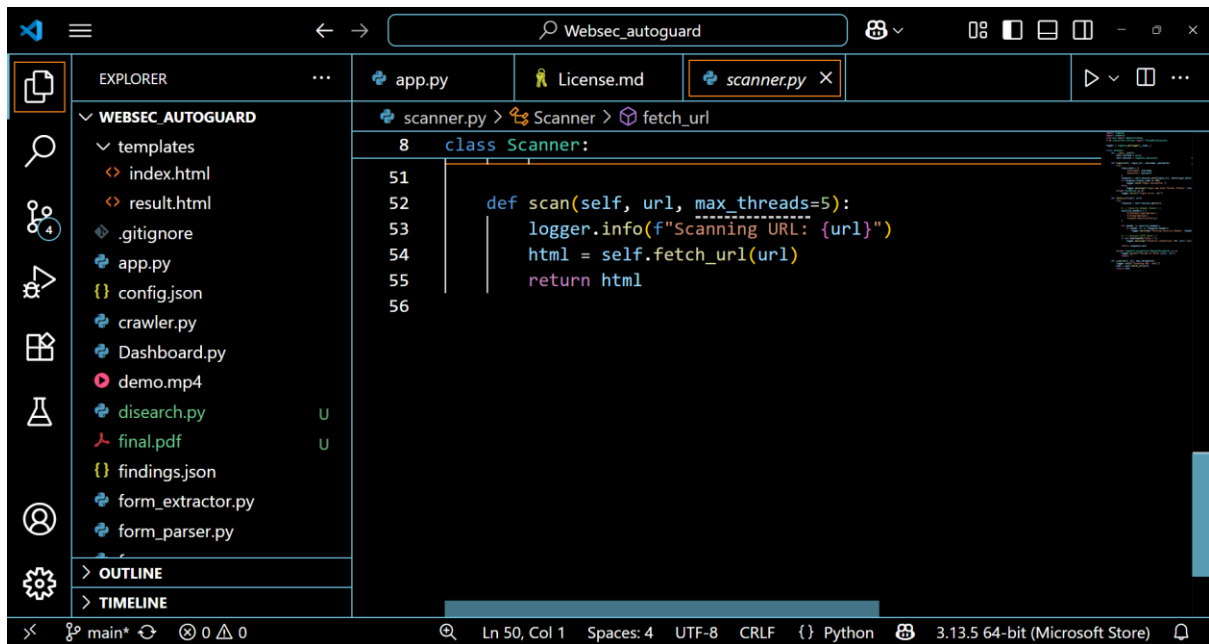
- Created a virtual environment.
- Installed required packages using `pip install -r requirements.txt`.



```
PS C:\projects\Websec_Autoguard> pip install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: PySocks==1.7.1 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 1)) (1.7.1)
Requirement already satisfied: Jinja2==3.0.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 2)) (3.1.6)
Requirement already satisfied: defusedxml==0.7.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 3)) (0.7.1)
Requirement already satisfied: pyopenssl==21.0.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 4)) (25.1.0)
Requirement already satisfied: requests==2.27.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 5)) (2.32.4)
Requirement already satisfied: requests_ntlm==1.3.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 6)) (1.3.0)
Requirement already satisfied: colorama==0.4.4 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 7)) (0.4.6)
Requirement already satisfied: ntlm_auth==1.5.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 8)) (1.5.0)
Requirement already satisfied: beautifulsoup4==4.8.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 9)) (4.13.4)
Requirement already satisfied: mysql-connector-python==8.0.20 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 10)) (9.3.0)
Requirement already satisfied: pycopp==3.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from pycopp[binary]>=3.0->-r requirements.txt (line 11)) (3.2.9)
Requirement already satisfied: defusedcsv==2.0.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 12)) (2.0.0)
Requirement already satisfied: requests-toolbelt==1.0.0 in c:\users\niran\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from -r requirements.txt (line 13)) (1.0.0)
```

### Step 2: Built the Scanner Module

- Wrote scanner.py to fetch page source using requests.



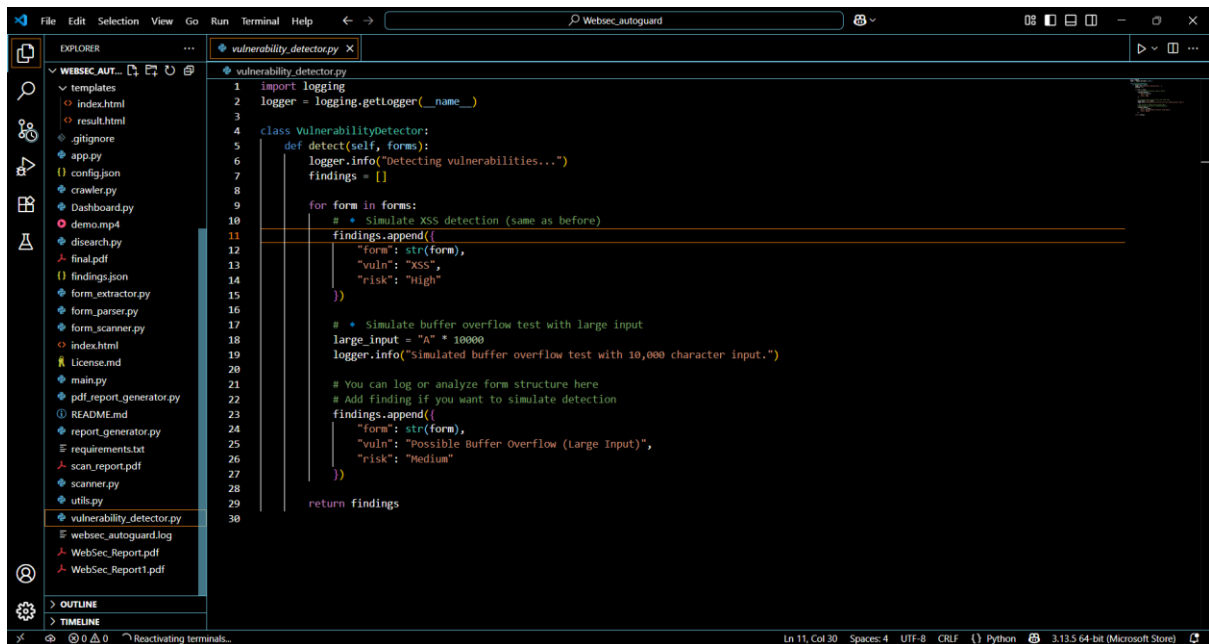
### Step 3: Form Parsing

- Parsed forms using form\_parser.py and BeautifulSoup.



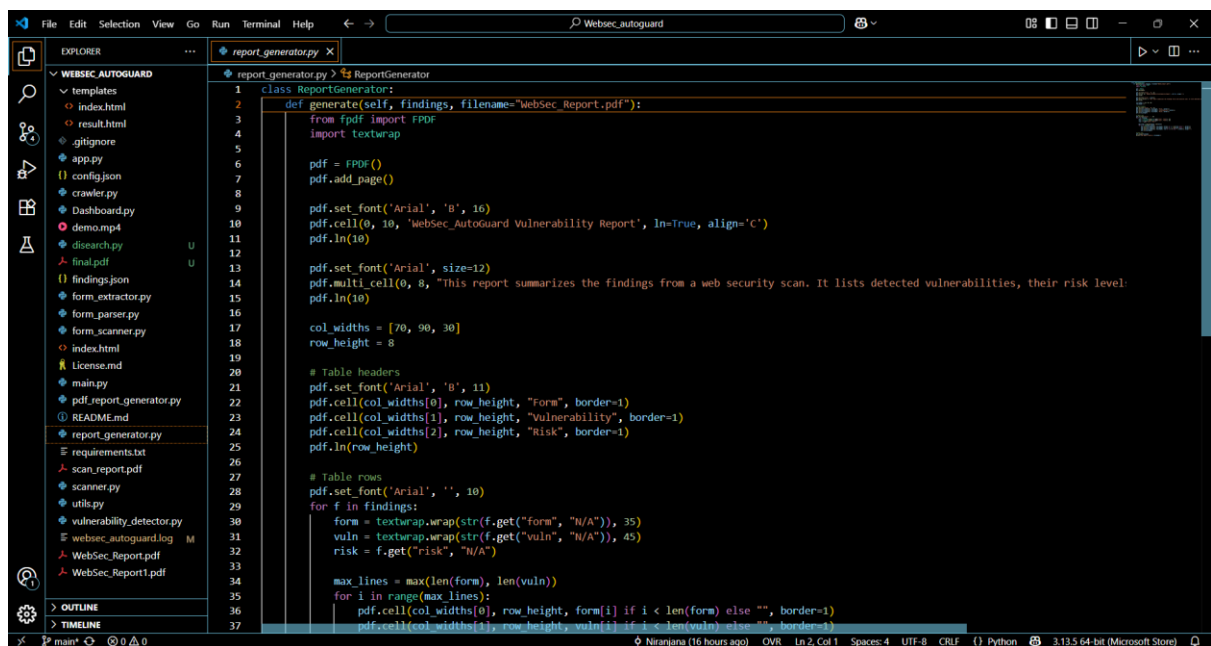
### Step 4: Vulnerability Detection

- Injected XSS payloads and checked reflected inputs.
- Implemented detection logic in vulnerability\_detector.py.



## Step 5: Report Generation

- Created formatted PDF reports using report\_generator.py.



## Step 6: Web UI Development

- Developed Flask app in app.py.
- Built HTML templates: index.html and result.html.

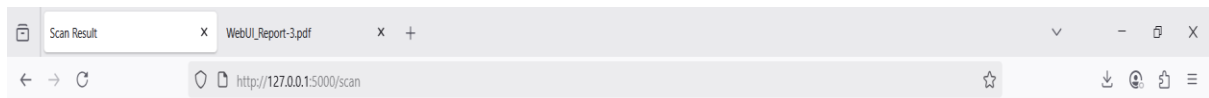


## Welcome to WebSec AutoGuard

Enter URL:

☒ XSS Scan

Start Scan



## Scan Result for http://testphp.vulnweb.com

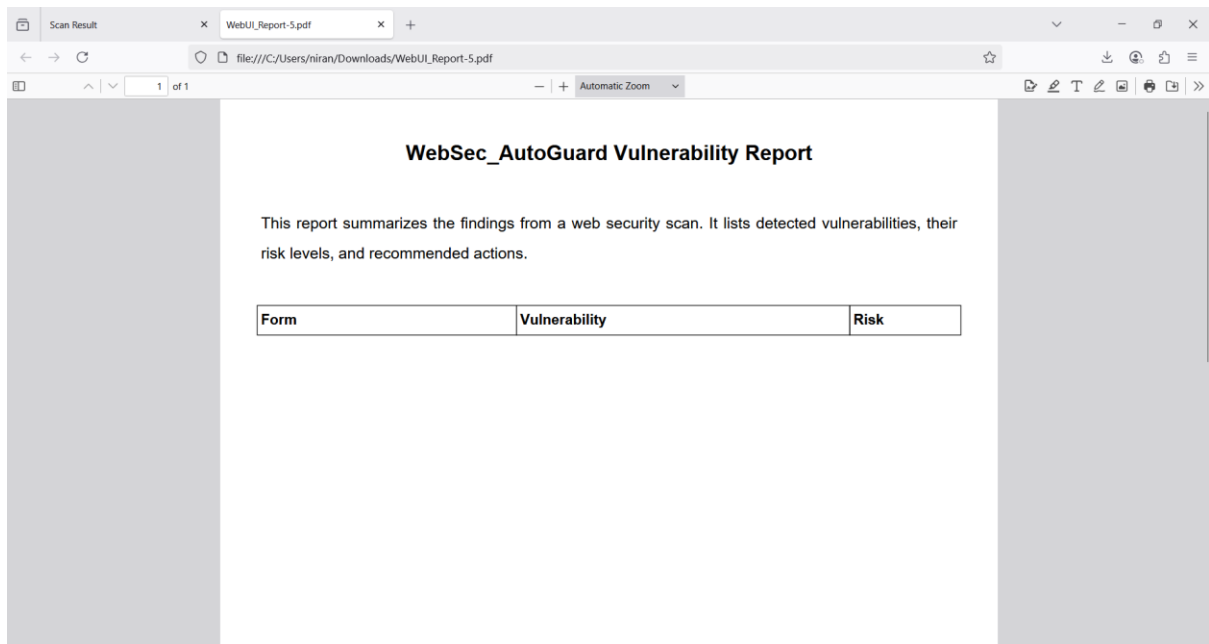
Vulnerabilities:

No vulnerabilities found.

Report File:

[WebUI\\_Report.pdf](#)

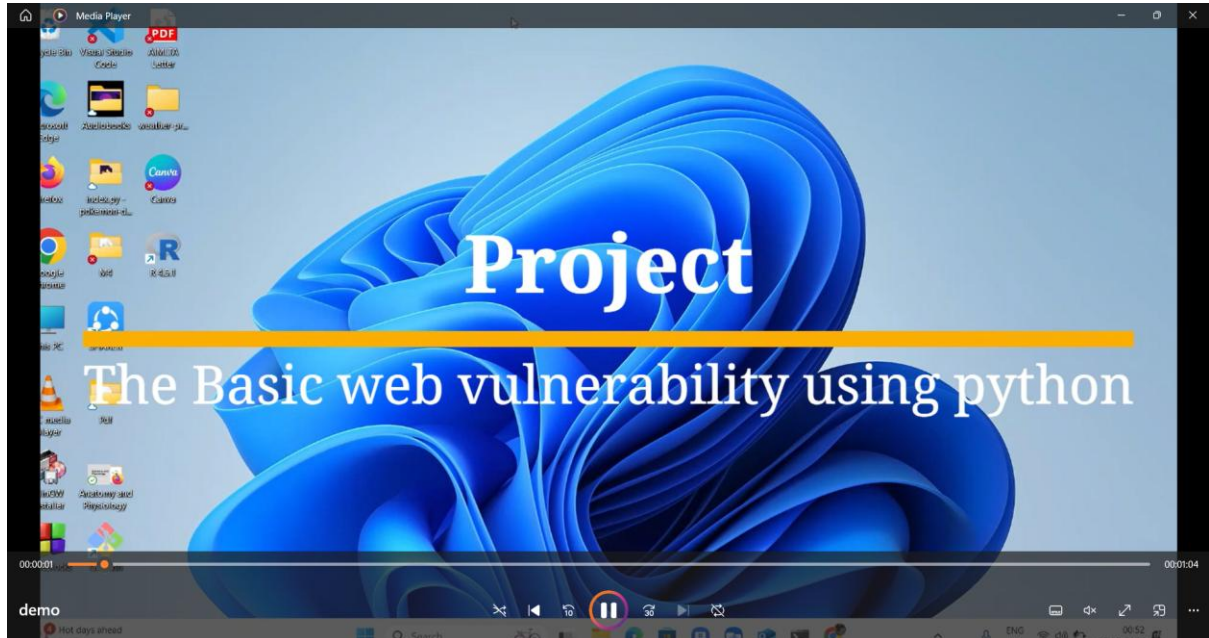
Go Back



Screenshot of dashboard form, scan results and pdf report.

### Step 7: Demo Video Recording

- Used Screen Recording for screen recording.
- Recorded usage of dashboard with audio/subtitles.



### Step 8: GitHub Publishing

- Initialized Git, added .gitignore, LICENSE, README.md, and requirements.txt.
- Committed files and pushed to GitHub repo.

```
Administrator: Windows PowerShell
1 file changed, 46 insertions(+)
create mode 100644 index.html
PS C:\projects\Websec_autoguard> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 917 bytes | 917.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Anaj-narinV/Websec_AutoGuard.git
e7fff18..ea7d7ba main -> main
PS C:\projects\Websec_autoguard> git add pdf_report_generator.py
PS C:\projects\Websec_autoguard> git commit -m "Added pdf_report_generator.py"
[main 5695294] Added pdf_report_generator.py
1 file changed, 68 insertions(+)
create mode 100644 pdf_report_generator.py
PS C:\projects\Websec_autoguard> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.06 KiB | 1.06 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Anaj-narinV/Websec_AutoGuard.git
ea7d7ba..5695294 main -> main
PS C:\projects\Websec_autoguard> git add scan_report.pdf
PS C:\projects\Websec_autoguard> git commit -m "Added scan_report.pdf"
[main 5a676f6] Added scan_report.pdf
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 scan_report.pdf
PS C:\projects\Websec_autoguard> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 13.96 KiB | 13.96 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Anaj-narinV/Websec_AutoGuard.git
```

## Demo Video With Subtitles

The video file is attached in Github repo "Demo.mp4"

## Final Deliverables

- scanner.py, form\_parser.py, vulnerability\_detector.py, report\_generator.py, utils.py
- app.py, templates/index.html, templates/result.html, static/
- README.md, LICENSE, .gitignore, requirements.txt,
- WebSec\_Report.pdf, config.json, websec\_autoguard.log

## GitHub Repository

[https://github.com/Anaj-narinV/Websec\\_AutoGuard](https://github.com/Anaj-narinV/Websec_AutoGuard)

## Conclusion

WebSec\_AutoGuard is a comprehensive and beginner-friendly web vulnerability scanner focused on detecting XSS threats. Through modular code structure, a web interface, reporting, and GitHub documentation, it showcases essential skills in cybersecurity, web development, and open-source contribution. This project serves as a practical foundation for understanding form security, input validation, and safe coding practices in modern web applications.