National College of Ireland

Configuration Manual

MSc Research Project

Cyber Security

Suraj Suprabha Raju

Student ID: 23183462

School of Computing

National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	……Suraj Suprabha Raju……
Student ID:	……23183462……
Programme:	……MSc Cyber Security…… **Year:** ……2024……
Module:	……MSc Research Practicum……
Lecturer:	……Khadija Hafeez……
Submission Due Date:	……12/08/2024……
Project Title:	Enhancing Web Security: Detecting and Preventing Reflected Cross-Site Scripting (XSS) Attacks
Word Count:	……1037…… **Page Count:** ……9……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Suraj Suprabha Raju……

Date: ……12/08/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Suraj Suprabha Raju
Student ID: 23183462

1 Introduction

This configuration manual will guide you to the entire information about the project. Project name is “Enhancing Web Security: Detecting and Preventing Reflected Cross-Site Scripting (XSS) Attacks”, which is a framework/model to effectively detect and prevent reflected XSS attacks. Reflected XSS allows attackers to inject malicious scripts into the URL of legitimate web servers. These scripts can play a major role to steal individual records, impersonate customers, and perform distinctive malicious moves.

2 Hardware requirements

• System type: 64-bit Windows system, x64 based processor
• Processor: 11th Gen Intel(R) Core(TM) i5 @ 2.40GHz
• RAM: 8.0 GB
• Storage: 120GB SSD

3 Software requirements

Operating System: Windows 10
Languages: Python 3.12.5, Javascript and HTML, SQLLite
Framework: Python flask
IDE: Visual Studio Flask 1.91.1

4 Python Libraries

The following libraries are required to run this application in flask.
Flask

- re: regular expression library
- markupsafe
- sqlite3
- datetime
- Flask WTF
- Wtforms

```
from flask import Flask, request, render_template_string, jsonify, make_response
import re
from markupsafe import escape
import sqlite3
from datetime import datetime
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired
from flask_wtf.csrf import CSRFProtect
```

Figure 1: Libraries

5 Implementation & Testing

After the installation of both visual studio code and python on your machine, we should make sure that above said (see figure 1) python libraries are up to date on your system.

```
def check_xss(input_text):
    xss_patterns = [
        r'<.*?script.*?>.*?</.*?script.*?>',
        r'<.*?style.*?>.*?</.*?style.*?>',
        r'<.*?img.*?src.*?=.*?javascript:.*?>',
        r'<.*?iframe.*?>.*?</.*?iframe.*?>',
        r'<.*?a.*?href.*?=.*?javascript:.*?>',
        r'javascript:.*',
        r'(<|%3C).*?script.*?(>|%3E)',
        r'(<|%3C).*?img.*?(>|%3E).*?src=.*?javascript:.*?(>|%3E)'
    ]

    for pattern in xss_patterns:
        if re.search(pattern, input_text, re.IGNORECASE):
            return "XSS Detected!"
    sanitized_input = escape(input_text)
    return "No XSS Detected."
```

Figure 2: Function for checking XSS and sanitization of user input

Whenever a user gives input to this system this function (see figure 2) will compare with regular expression patterns. These patterns are mainly designed to match and detect XSS attack vectors. Because they will try to inject malicious scripts to exploit HTML tags. This function detects the following.

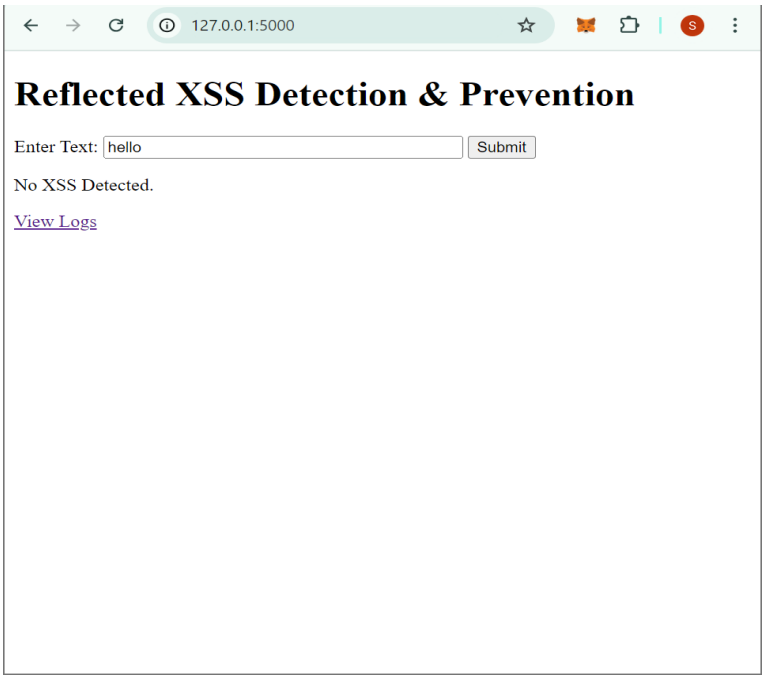

Figure 3: User input - GUI

The above figure shows the home page of the model, where the user enters input through the text box. This model is fully secured against reflected XSS attacks, SQL injection attacks, etc.

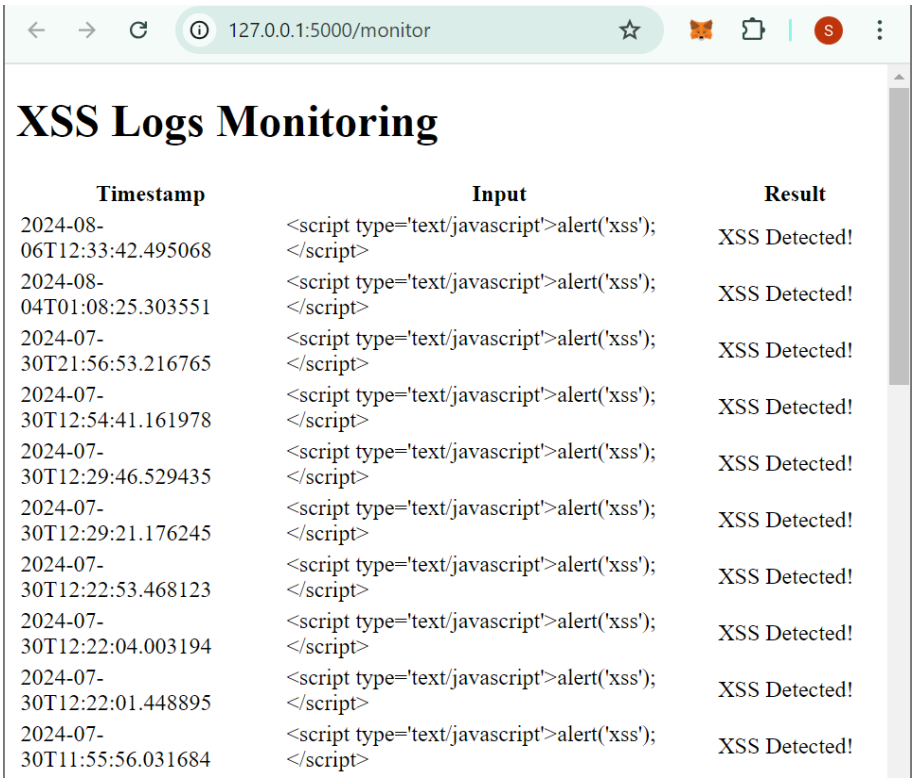

Figure 3: Real time monitoring - GUI

If a user provides malicious input to the web application, then this model immediately prevents the same and stores its details into the log database as well as displays onto the real time monitoring page.

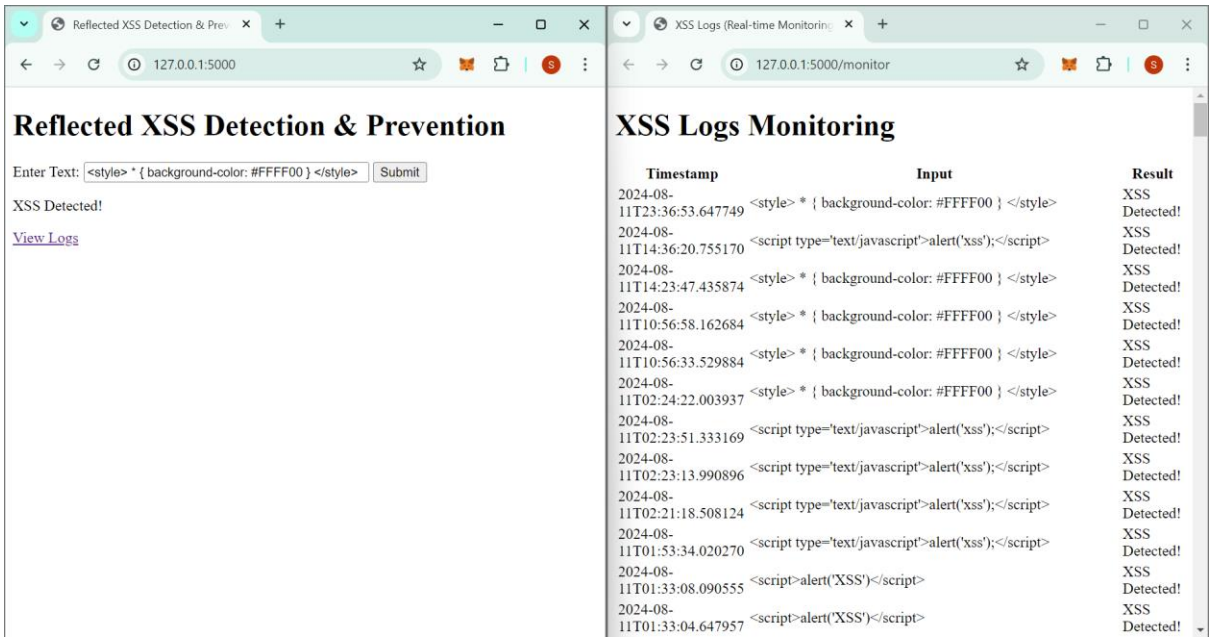

Figure 3: Checking for reflected XSS attack in proposed system

As per the above figure 3, I have submitted an input to change the background colour of the web page as yellow. The system has successfully detected the reflected XSS attack and the same is displayed on the real time monitoring page.

For the comparison purpose I have developed a Vulnerable web application, which is vulnerable to XSS and all other cyber attacks.

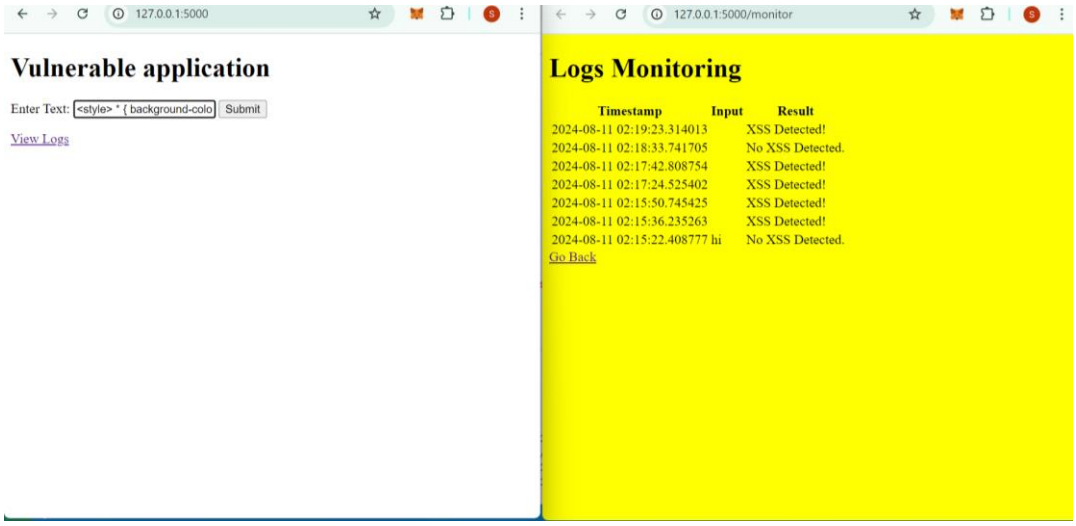

Figure 4: Checking for XSS attack in Vulnerable web application

Here I have submitted "<style> * { background-color: #FFFF00 } </style> " input (Snyk, 2024) to the vulnerable application. Then the color of the log monitoring page has been changed into yellow. So that we can say that this application is vulnerable to XSS attacks. The same input was given to the proposed system (see figure 3), which successfully detected & prevented reflected XSS attacks.

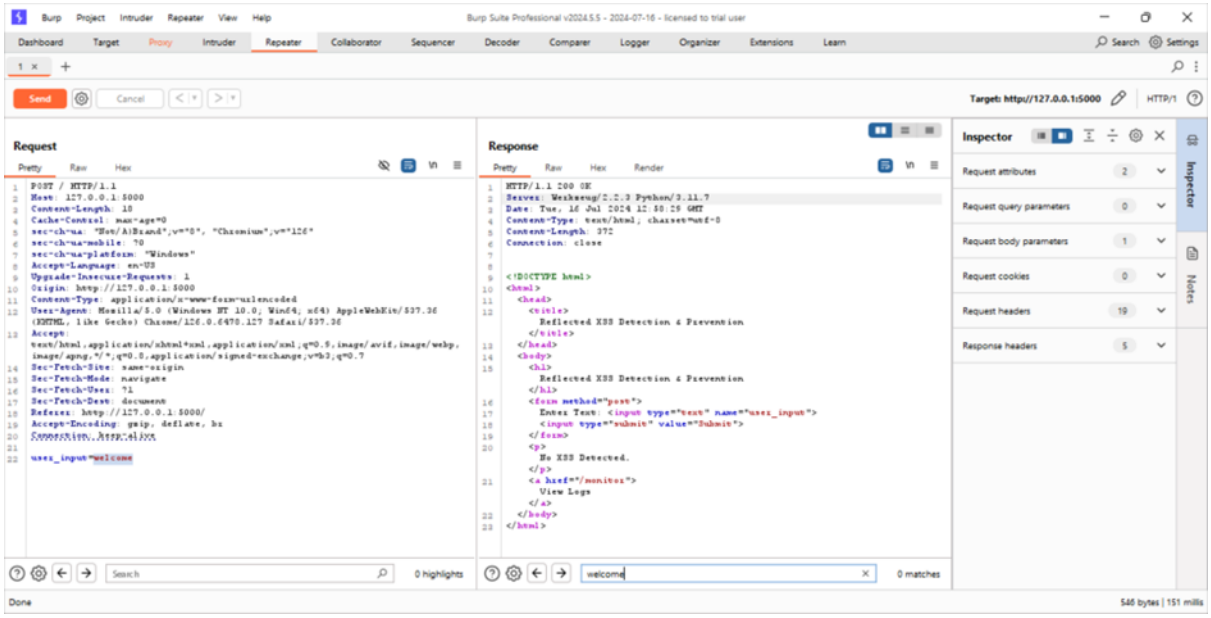

Figure 5: Testing of the proposed system by Burp suite Professional tool

First of all, I have tested whether this application has reflected XSS vulnerabilities or not. I Turned on the intercept feature in the Burp Suite tool and submitted the text “welcome” into the web application. This user input is forwarded into the repeater and intruder of this tool. Figure 4 shows the response part of the repeater does not have the user input, which is “welcome”. So, the reflected XSS vulnerability is not present in this application.

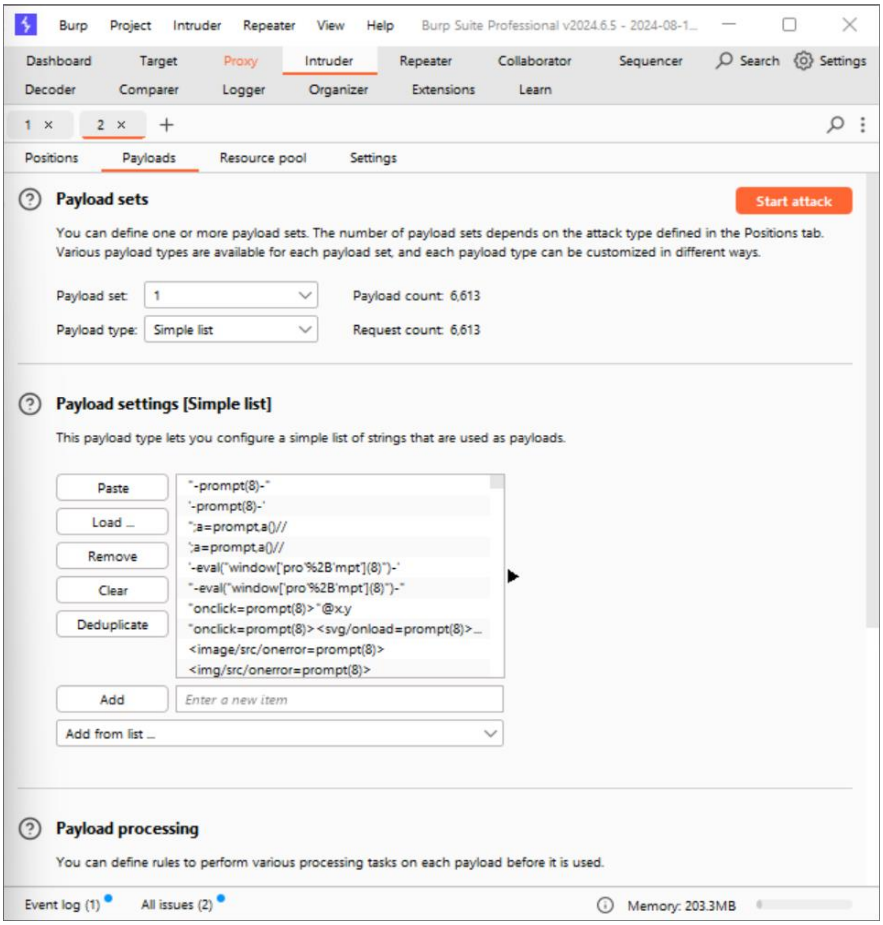

Figure 6: Testing of the proposed system by Burp suite Professional tool

Used a payload set (Riodrwn Rio D. 2021) of 6613 scripts to attack the proposed system through the intruder feature of the burp suite tool.

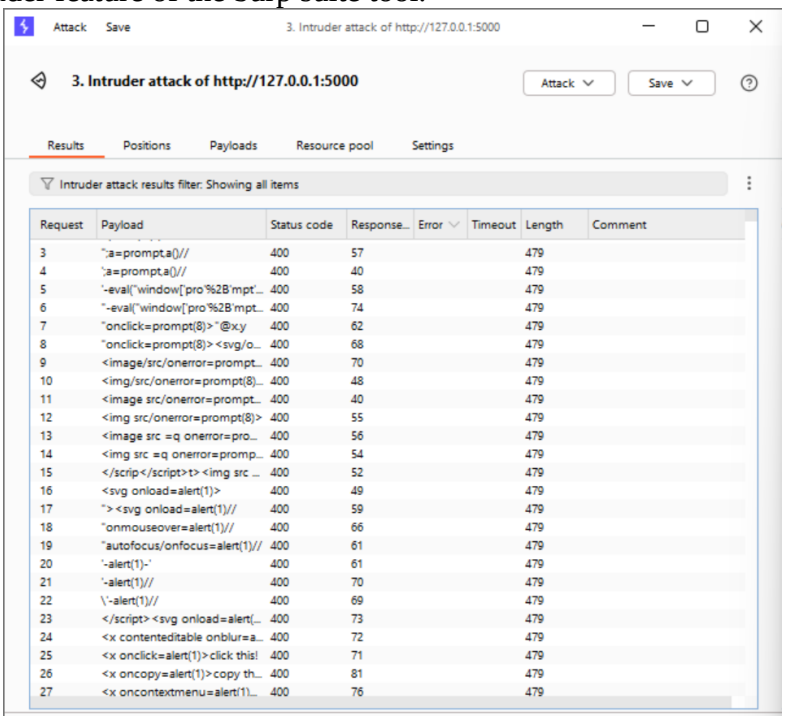

Figure 7: Testing Results - Burp suite Professional tool

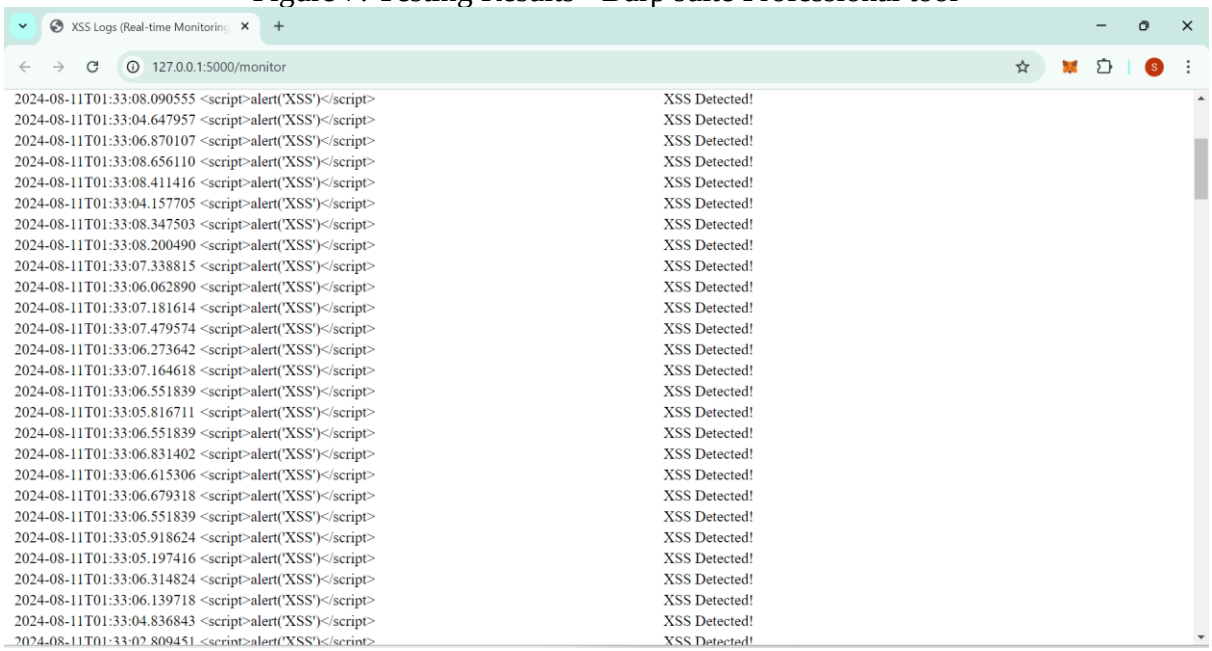

Figure 8: Live monitoring page showing the detected XSS attacks.

After the successful attack implemented by the Burp Suite tool, only the XSS attacks detected are shown in the live monitoring page. This helps the admin find out and filter the XSS attack payload. As per the above results, the proposed system was successfully detected and prevented the whole attacks with less response time. Size of the dataset (Riodrwn Rio D. 2021) was 6613. Logs regarding the detected attacks have been stored into the logs database and displayed onto the real time monitoring page.

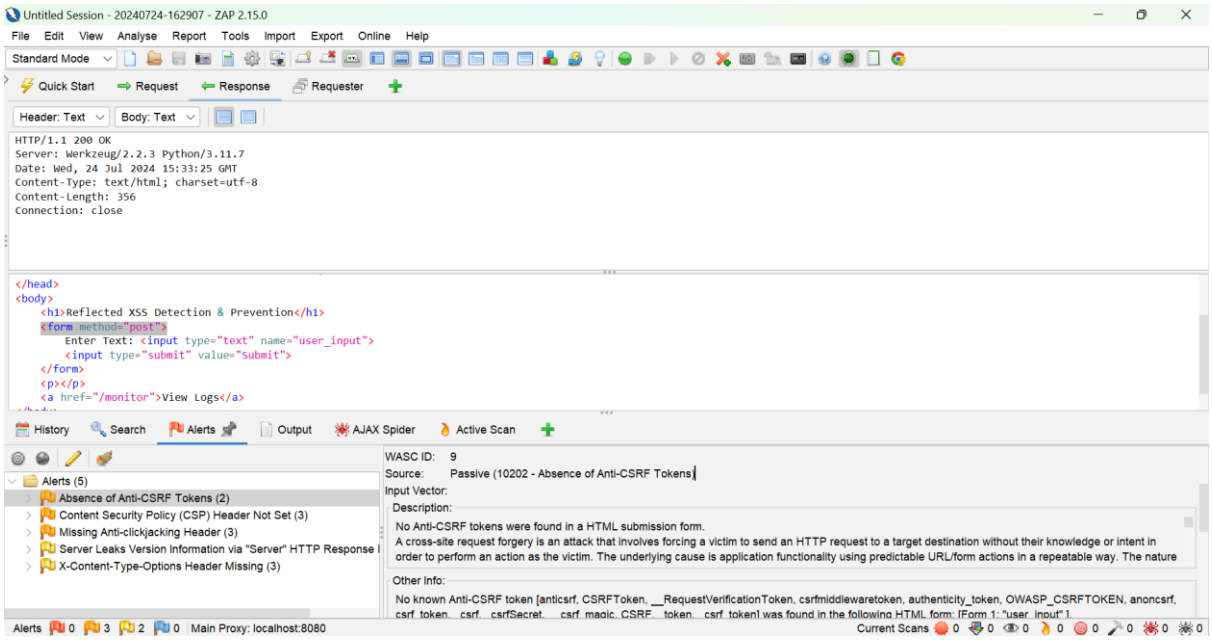

Figure 9: Testing - OWASP ZAP tool

During this testing using the OWASP ZAP tool, the following vulnerabilities were present in the system.

- CSRF token was missing in the HTML submission form. So there was a chance for the Cross site request forgery attack.
- The anti-clickjacking header was missing, so there was a possibility of anti-clickjacking attacks.

I have fixed the above vulnerabilities by implementing both CSRF token and Content Security policies (CSP). The CSRF token prevents any unauthorised commands being executed by any other actor on behalf of the user. I have implemented Content Security policies for preventing the anti-clickjacking attacks. The Content Security policies (CSP) will prevent the anti-clickjacking attacks and other kinds of attacks including XSS attacks.

Again I had conducted another testing for finding the vulnerabilities.

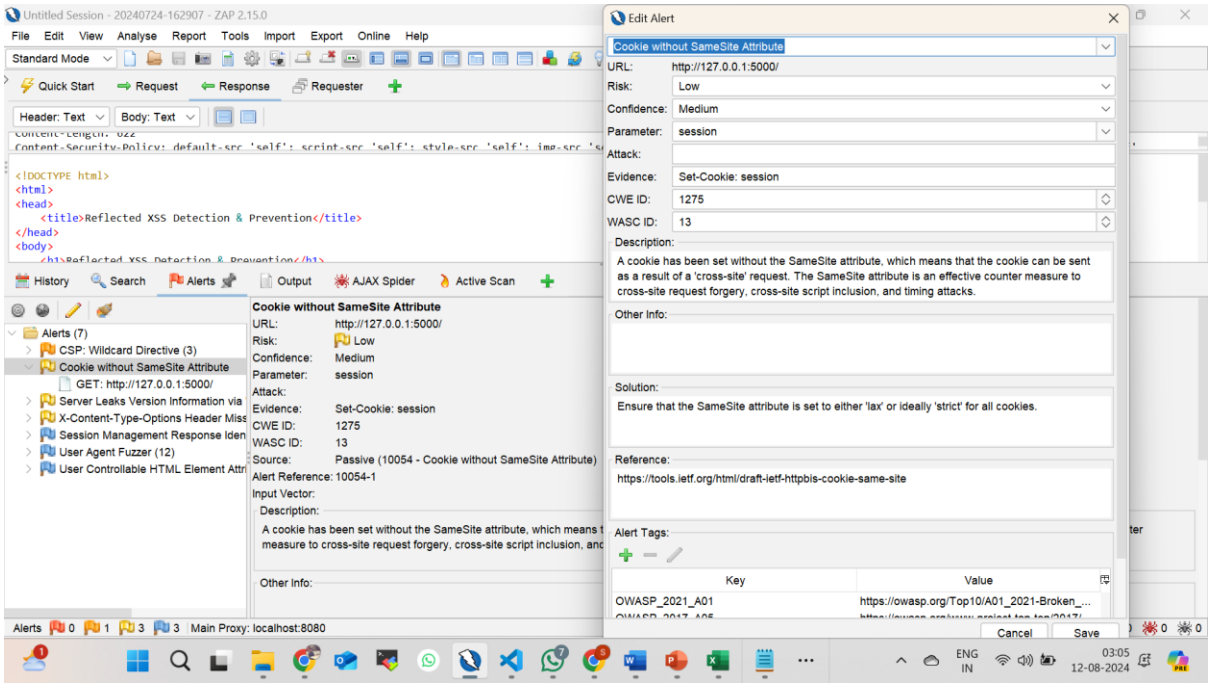

Figure 10: Another Testing - OWASP ZAP tool

In this testing, there was another vulnerability, which is “cookie without Same attribute”. So there was a chance of a cross site request forgery attack. Here I have implemented CSRF protection by setting the same site attributes for the session cookies. It ensures that cookies are only transmitted via HTTPS which ensures confidentiality and authenticity of the input (Amal Shaji. 2022).

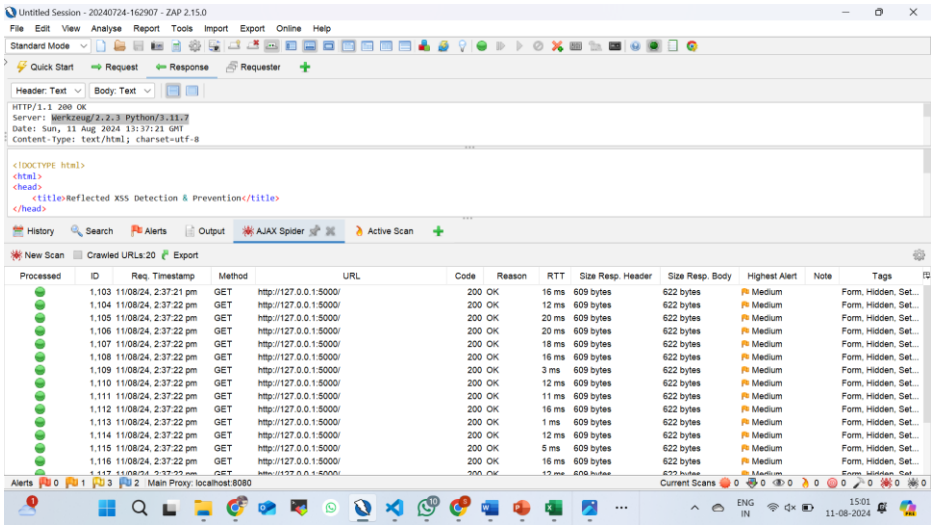

Figure 11: Scanning - OWASP ZAP tool

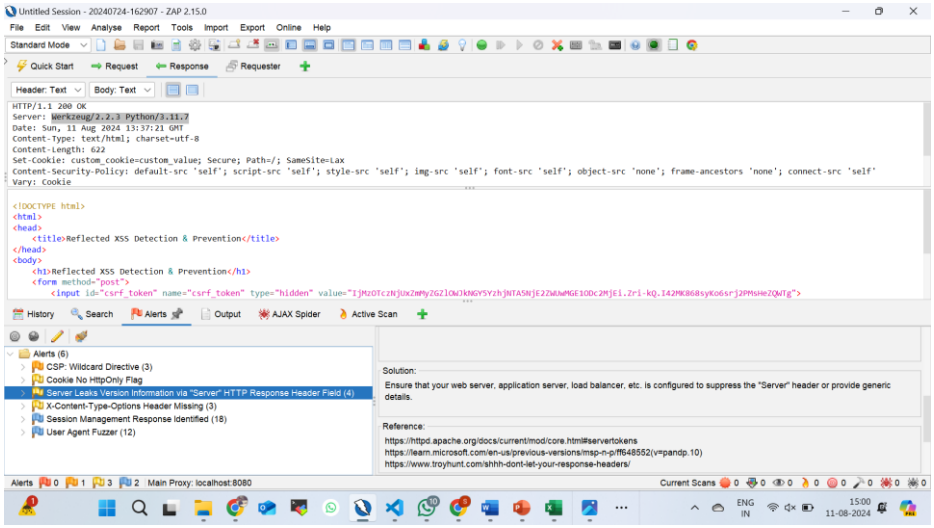

Figure 12: Testing Results - OWASP ZAP tool

To fine tune the results I did multiple sample testing in order to make my proposed system a secured one against not only reflected XSS attacks but other attacks too. Currently we can see an alert raised for both http header and CSP header (as seen in figure 12), as it is running on the Python flask and hosted on the local host system.

References

Amal Shaji, 2022. CSRF Protection in Flask. [Online] Available at: https://testdriven.io/blog/csrf-flask/ [Accessed 30 July 2024].

Snyk, 2024. Cross-site scripting (XSS). [Online] Available at: https://learn.snyk.io/lesson/xss/ [Accessed 3 August 2024].

Riodrwn Rio D. 2021. Cross Site Scripting (XSS) Vulnerability Payload List. [Online] Available at: https://github.com/payloadbox/xss-payload-list/tree/master/Intruder [Accessed 1 Aug 2024].