

# Vulnerability Assessment Report

Future Interns Task 1

By Aayush Kussial

# INTRODUCTION

This is a task given to me by Future interns to complete which involved doing a vulnerability assessment report on a live public website. The website i had selected to do this report on is <http://testphp.vulnweb.com>. The scope of this assessment had included passive and non intrusive testing only, manual inspection using browser developer tools, automated scanning using the tool OWASP ZAP and no exploitation or active attacks were performed.

Tools used were: OWASP ZAP, Google chrome developer tools and Canva

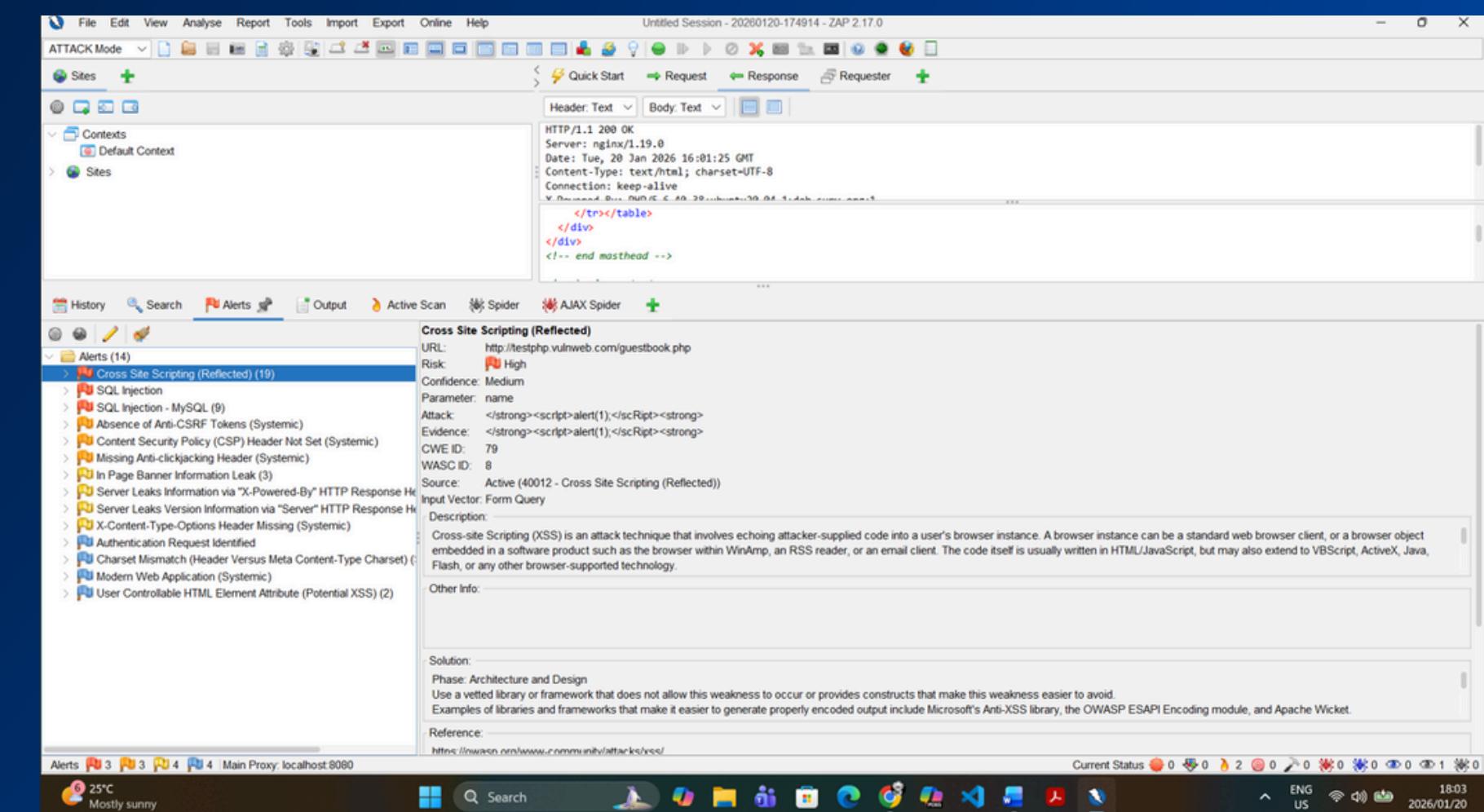


# Scope and Methodology

- Website tested : <http://testphp.vulnweb.com>.
- Passive testing only
- Tools used : OWASP ZAP, Google chrome developer tools, Canva
- No exploitation

# Vulnerabilities

- Cross site scripting( Risk Level: High):attack technique that involves echoing attacker-supplied code into a user's browser instance



# SQL Injection (Risk Level:High):

A cyber attack where malicious SQL code is inserted into database queries.

The screenshot shows the ZAP interface with the following details:

- Header:** POST http://testphp.vulnweb.com/guestbook.php HTTP/1.1  
host: testphp.vulnweb.com  
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36  
pragma: no-cache  
cache-control: no-cache
- Body:** name=ZAP&text=&submit=add+message%27+AND+%271%27%3D%271%27+--+
- Alerts:** SQL Injection (selected)
- Details of Selected Alert:**
  - URL:** http://testphp.vulnweb.com/guestbook.php
  - Risk:** High
  - Confidence:** Medium
  - Parameter:** submit
  - Attack:** add message' OR '1'='1' --
  - Evidence:** None
  - CWE ID:** 89
  - WASC ID:** 19
  - Source:** Active (40018 - SQL Injection)
  - Input Vector:** Form Query
  - Description:** SQL injection may be possible.
  - Other Info:** The page results were successfully manipulated using the boolean conditions [add message' AND '1'='1' -- ] and [add message' OR '1'='1' -- ]
  - Solution:** Do not trust client side input, even if there is client side validation in place.  
In general, type check all data on the server side.

# Absence of anti CSRF token (Risk Level: Medium):

No anti CSRF tokens were found in HTML submission form.

The screenshot shows a web-based security analysis tool. At the top, there's a header bar with tabs for 'Header: Text' and 'Body: Text'. Below the header, the 'Body: Text' tab is active, displaying the raw HTTP response headers and the beginning of the HTML page source. The page source shows a form element:

```
<!--end content -->  
  
<div id="navBar">  
  <div id="search">  
    <form action="search.php?test=query" method="post">
```

Below the page source, there's a navigation bar with icons for History, Search, Alerts, Output, Active Scan, Spider, and AJAX Spider. The 'Alerts' tab is selected. On the left, a sidebar lists various alerts categorized under 'Alerts (19)'. One alert is highlighted: 'Absence of Anti-CSRF Tokens (Systemic)'. To the right of this alert, detailed information is displayed in a panel:

**Absence of Anti-CSRF Tokens**  
URL: http://testphp.vulnweb.com  
Risk: Medium  
Confidence: Low  
Parameter:  
Attack:  
Evidence: <form action="search.php?test=query" method="post">  
CWE ID: 352  
WASC ID: 9  
Source: Passive (10202 - Absence of Anti-CSRF Tokens)  
Input Vector:  
Description:  
No Anti-CSRF tokens were found in a HTML submission form.  
A cross-site request forgery is an attack that involves forcing  
Other Info:  
No known Anti-CSRF token [anticsrf, CSRFToken, \_\_RequestVerificationToken, csrfmiddlewaretoken, authenticity\_token, OWASP\_CSRFTOKEN, anoncsrf, Solution:

# Remediation Recommendations

- Cross site scripting – encode output before rendering in HTML, validate all user input on server side, implement a strong content security policy.
- SQL Injection – use parameterized queries, apply strict server side input validations, use least privilege database accounts.
- Absence of anti CSRF token – Implement unique CSRF tokens in all state changing forms, validate tokens on the server.

# Conclusion

The vulnerability assessment identified multiple security weaknesses including high risk issues such as cross site scripting and SQL injection. These vulnerabilities could potentially be exploited if not resolved. Implementing the recommended remediation measures would improve the security of the application.

**THANK  
YOU**