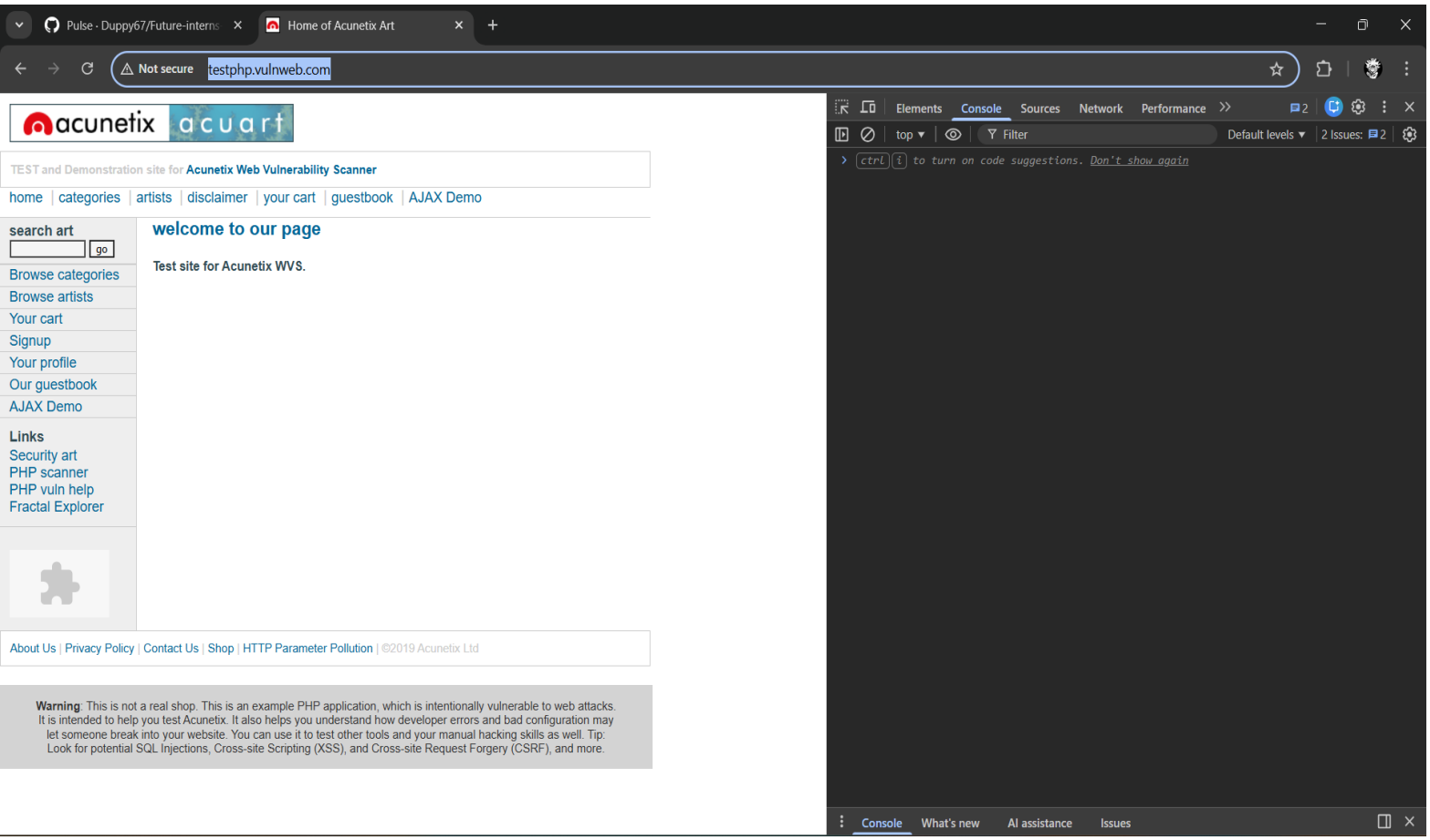
Introduction

This report presents a vulnerability assessment which had been done on a publicly accessible website http://testphp.vulnweb.com/. The purpose of this assessment was to identify common web application vulnerabilities using ethical testing techniques. It was performed using manual browser based analysis and passive scanning tools such as OWASP ZAP.

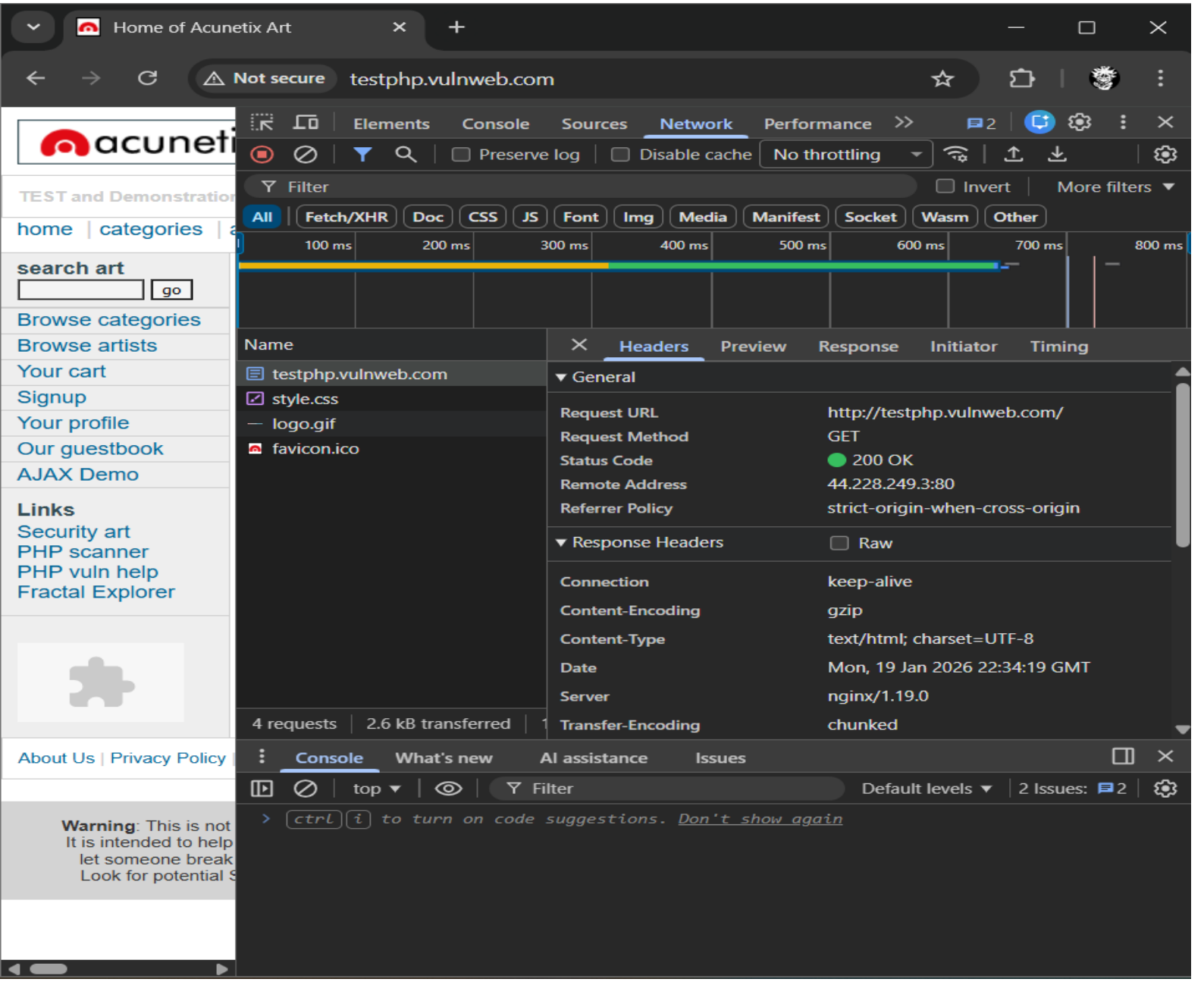
Task 1

Checking JavaScript errors or warnings



The browser developer console had been reviewed during page load to identify any errors or warnings, no errors or warnings were observed which shows stable client side execution.

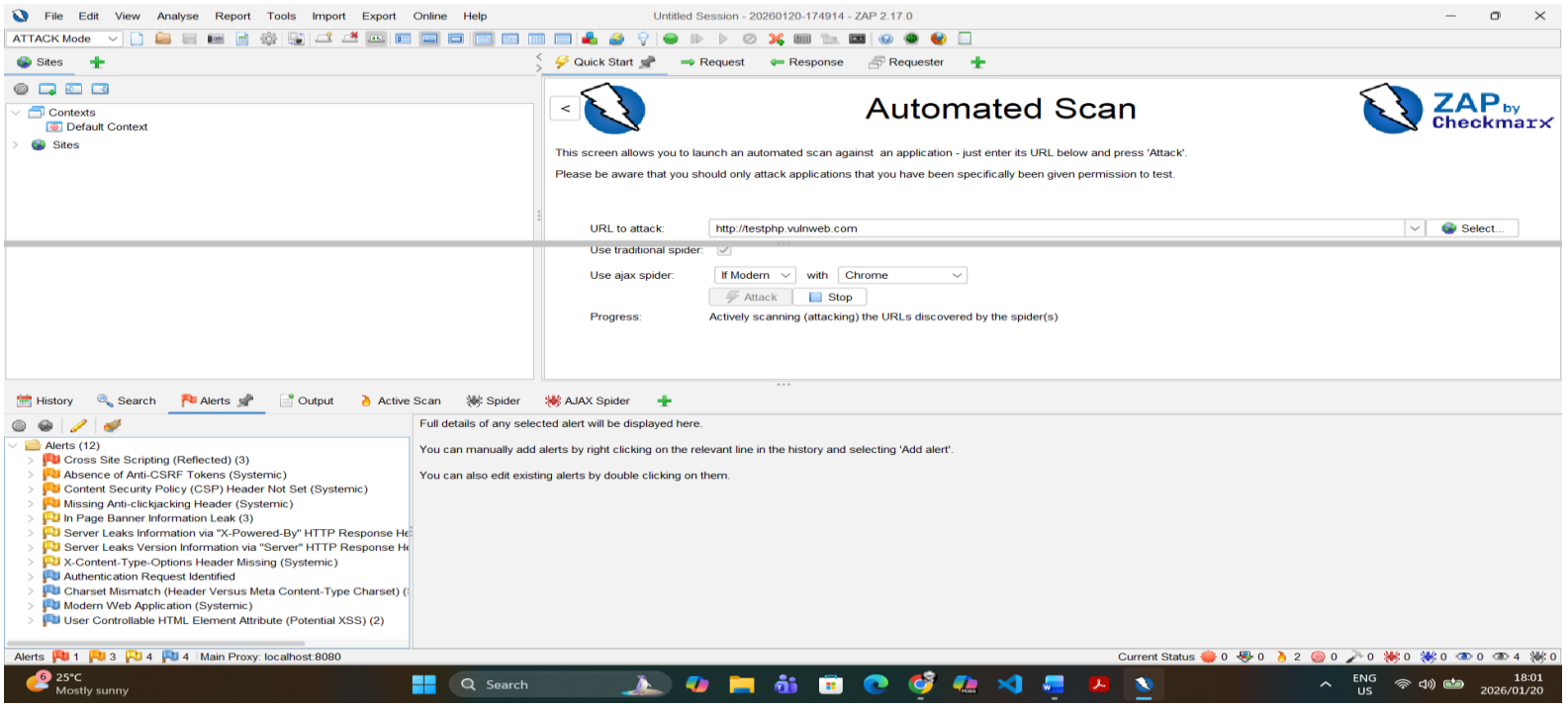
Checking network requests and security headers

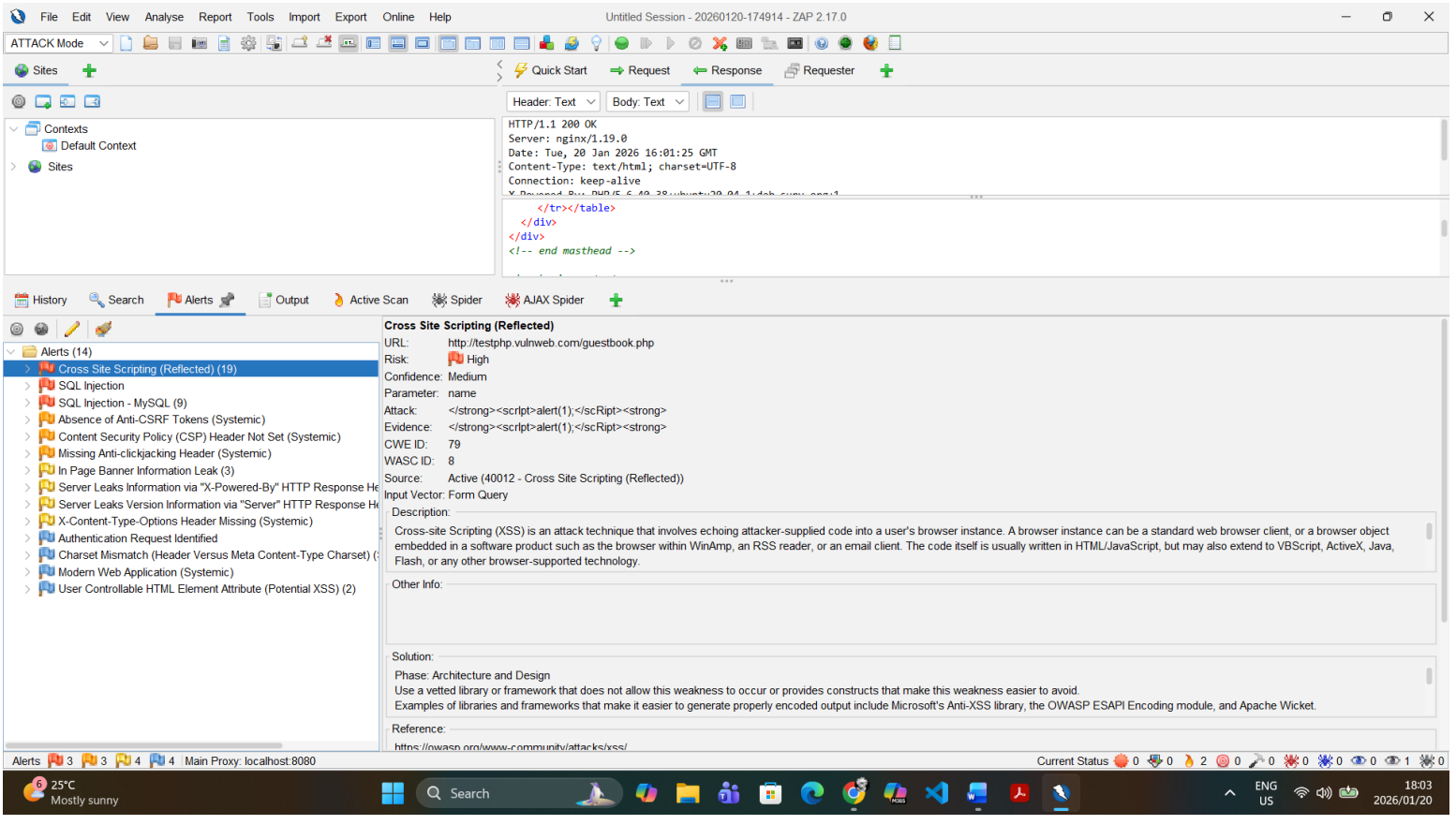


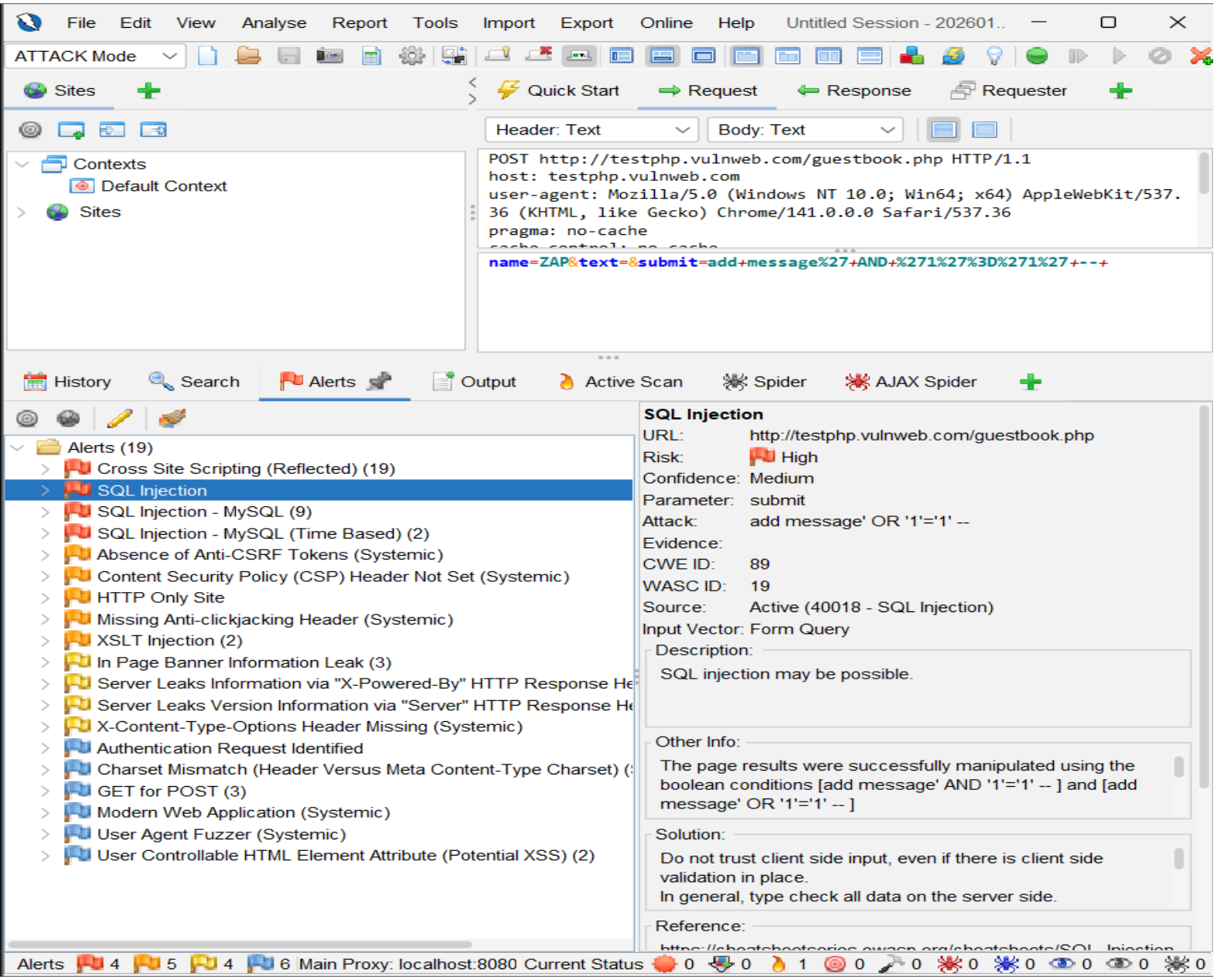
This site lacks multiple security headers such as content security policy, x frame options, x content type options and strict transport security. This exposes the website to multiple attacks such as cross site scripting and man in the middle attacks.

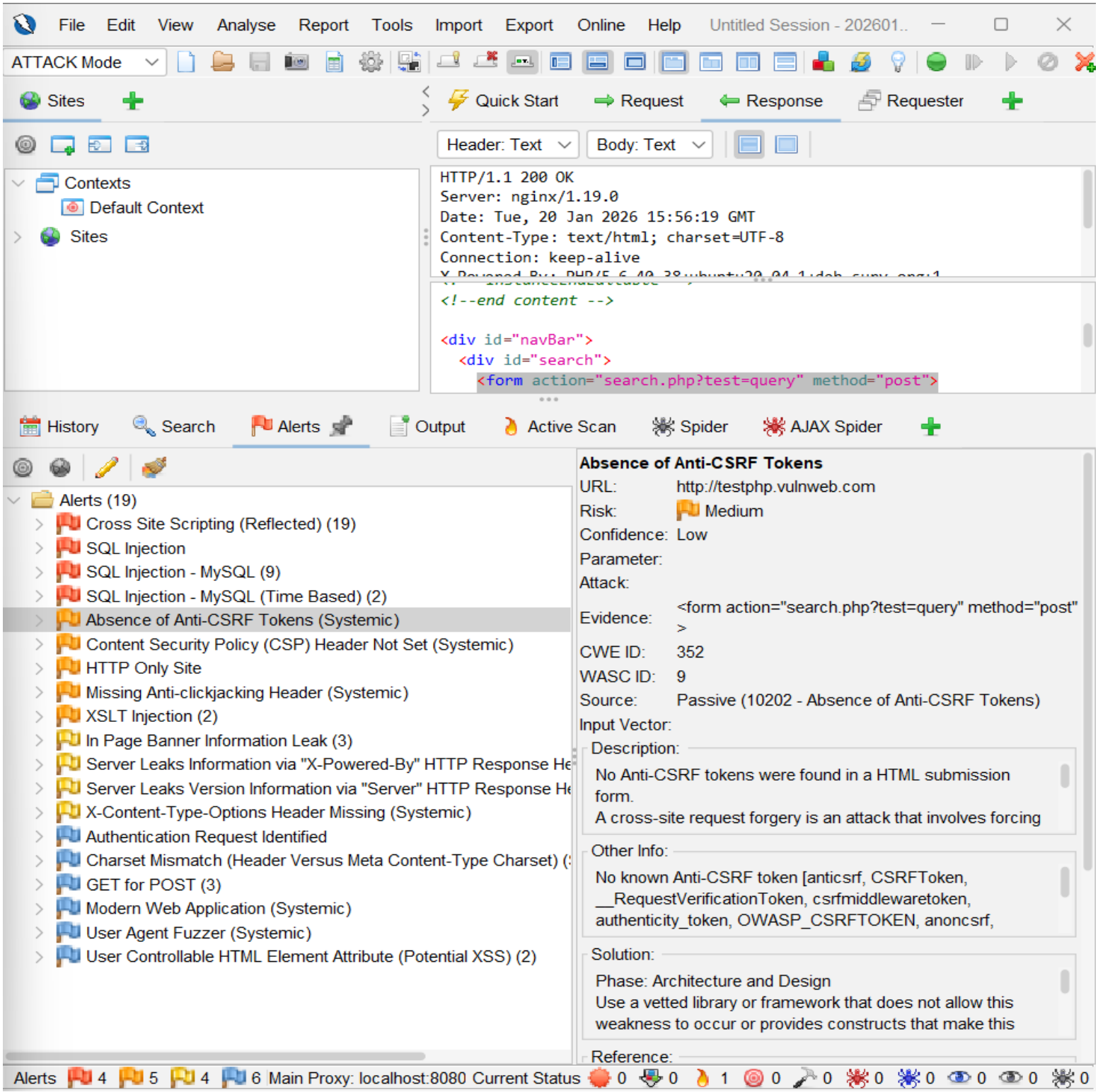
The request method used on this site is: Request Method: GET this means that data is passed via URL and this makes it easier to test for SQLi.

Automated Passive Vulnerability Assessment (OWASP ZAP)






File Edit View Analyse Report Tools Import Export Online Help Untitled Session - 202601..
ATTACK Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
Contexts
Default Context
Sites
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
name=ZAP&text=&submit=add+message%27+AND+%271%27%3D%271%27+--+
History
Search
Alerts
Output
Active Scan
Spider
AJAX Spider
Alerts (19)
Cross Site Scripting (Reflected) (19)
SQL Injection
SQL Injection - MySQL (9)
SQL Injection - MySQL (Time Based) (2)
Absence of Anti-CSRF Tokens (Systemic)
Content Security Policy (CSP) Header Not Set (Systemic)
HTTP Only Site
Missing Anti-clickjacking Header (Systemic)
XSLT Injection (2)
In Page Banner Information Leak (3)
Server Leaks Information via "X-Powered-By" HTTP Response He
Server Leaks Version Information via "Server" HTTP Response He
X-Content-Type-Options Header Missing (Systemic)
Authentication Request Identified
Charset Mismatch (Header Versus Meta Content-Type Charset) (
GET for POST (3)
Modern Web Application (Systemic)
User Agent Fuzzer (Systemic)
User Controllable HTML Element Attribute (Potential XSS) (2)
SQL Injection
URL: http://testphp.vulnweb.com/guestbook.php
Risk: High
Confidence: Medium
Parameter: submit
Attack: add message' OR '1'='1' --
Evidence:
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.
Reference:
Alerts 4 5 4 6 Main Proxy: localhost:8080 Current Status 0 0 1 0 0 0 0 0 0 0


File Edit View Analyse Report Tools Import Export Online Help
Untitled Session - 202601..
ATTACK Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
Contexts
Default Context
Sites
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Tue, 20 Jan 2026 15:56:19 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
<!--end content -->

<form action="search.php?test=query" method="post">
History
Search
Alerts
Output
Active Scan
Spider
AJAX Spider
Alerts (19)
Cross Site Scripting (Reflected) (19)
SQL Injection
SQL Injection - MySQL (9)
SQL Injection - MySQL (Time Based) (2)
Absence of Anti-CSRF Tokens (Systemic)
Content Security Policy (CSP) Header Not Set (Systemic)
HTTP Only Site
Missing Anti-clickjacking Header (Systemic)
XSLT Injection (2)
In Page Banner Information Leak (3)
Server Leaks Information via "X-Powered-By" HTTP Response He
Server Leaks Version Information via "Server" HTTP Response He
X-Content-Type-Options Header Missing (Systemic)
Authentication Request Identified
Charset Mismatch (Header Versus Meta Content-Type Charset) (
GET for POST (3)
Modern Web Application (Systemic)
User Agent Fuzzer (Systemic)
User Controllable HTML Element Attribute (Potential XSS) (2)
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:
Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this
Reference:
Alerts 4 5 4 6 Main Proxy: localhost:8080 Current Status 0 0 1 0 0 0 0 0 0 0

Recording Vulnerability Assessment

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

Absence of anti CSRF token (Risk Level: Medium) – No anti CSRF tokens were found in HTML submission form.

Content security policy (CSP) Header not set (Risk Level: Medium) – This is a added layer of security that helps to detect and mitigate certain types of attacks such as cross site scripting and data injection.

HTTP only site (Risk Level: Medium) – The site is only served under HTTP and not HTTPS.

XSLT Injection (Risk Level: Medium) - Injection using XSL transformations may be possible and may allow an attacker to read system information.

In page banner information leak (Risk Level: Medium) - The server returned a version banner string in the response content. Such information leaks may allow attackers to further target specific issues impacting the product and version in use.

Remediation Recommendations

Cross site scripting – encode output before rendering in HTML, validate all user input on server side, implement a strong content security policy.

Eg : use content security policy : Content-Security-Policy: default-src 'self';

Security Benefits: Prevents malicious scripts from executing in users browsers.

SQL Injection – use parameterized queries, apply strict server side input validations, use least privilege database accounts.

Eg: SELECT * FROM users WHERE id = ?

Security Benefits: Prevents attackers from manipulating database queries.

Absence of anti CSRF token – Implement unique CSRF tokens in all state changing forms, validate tokens on the server.

Eg: Add hidden CSRF token field to forms.

Security Benefits: Prevents cross site request forgery attacks.

Content security policy(CSP) header not set – Define a strict CSP header, limit scripts, styles and media to trusted domains.

Eg: Content-Security-Policy: default-src 'self';

Security Benefits: reduces impact of data injection attacks.

HTTP only site – implement HTTPS using valid TLS certificate, redirect all HTTP traffic to HTTPS.

Eg: Strict-Transport-Security: max-age=31536000; includeSubDomains

Security Benefits: Protects data confidentiality and integrity.

XSLT Injection – Disable external entity resolution in XSLT processors, apply strict input validation.

Security Benefits: Prevents abuse of XML processing engines

In page banner information leak – Disable detailed error messages, configure server to hide version information.

Eg:

Nginx: server_tokens off;

PHP: expose_php = Off

Security Benefits: Reduces information available to attackers.

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.