VULNERABILITY ASSESSMENT REPORT – FINAL CONTENT

Target Website: OWASP Juice Shop

Tool Used: OWASP ZAP (Passive Scan)

Internship Program: Future Interns – Cyber Security

Task: Task 1

Prepared By: Sohel Mujawar

Date: 31/01/2026

About the Task

Every business today relies on a website to represent its brand and services. However, many websites operate with security misconfigurations such as missing security headers, outdated components, or unintentional information exposure.

Most clients do not ask for hacking or exploitation. Instead, they seek clear answers to important business questions such as whether their website is secure, what risks exist, and which issues should be fixed first.

This task focuses on identifying common security weaknesses in a professional and ethical manner. The objective is to provide clarity, assess risk, and deliver actionable recommendations, similar to how a real-world security consultant would communicate findings to a business client.

Introduction

Web applications are frequently exposed to security risks due to misconfigurations and insecure design. This report presents a vulnerability assessment performed on a deliberately vulnerable web application using OWASP ZAP.

Objective

The objective of this assessment is to identify common web application vulnerabilities, classify their risk levels, and suggest appropriate remediation measures. The objective also includes presenting security findings in a business-friendly manner, avoiding excessive technical jargon, and prioritizing risks based on their potential impact on the organization. The goal is to support informed decision-making rather than technical exploitation.

Scope of Testing

The assessment was limited to non-intrusive testing of the OWASP Juice Shop application using passive scanning and manual exploration. No intrusive or exploitative testing was performed.

Tools Used

- OWASP ZAP
- Web Browser
- Browser DevTools
- Canva used for designing a clean and professional vulnerability assessment report

Scope & Ethics

This assessment was conducted by strictly following ethical security testing practices.

Allowed Activities:

- Analysis of publicly accessible web pages only
- Passive vulnerability scanning
- HTTP security header inspection
- Configuration and response analysis

Not Allowed Activities:

- Authentication bypass attempts
- Exploitation of vulnerabilities
- Brute force attacks
- Denial-of-Service (DoS) attacks
- Any activity that could disrupt or harm the target website

The assessment was performed from the perspective of a security auditor, focusing on risk identification and mitigation rather than offensive exploitation.

Methodology

The assessment process began with selecting a publicly accessible test website. Passive scanning was performed using OWASP ZAP to identify potential security misconfigurations without actively attacking the application. Manual exploration was conducted using a web browser and developer tools to inspect HTTP headers and client-side behavior.

All identified alerts were reviewed, categorized based on risk severity, and analyzed to understand their potential impact. No exploitation or intrusive testing was carried out during the assessment.

Target Website Details

- Website Name: OWASP Juice Shop
- Purpose : Intentionally vulnerable web application designed for security training and testing.

Vulnerability Summary

The following table provides a summary of the identified vulnerabilities and their associated risk levels to give a quick overview of the website's security posture.

Vulnerability Name	**Risk Level**
Cross-Domain Misconfiguration	Medium
CSP Header Not Set	Medium
Missing Anti-clickjacking Header	Medium
X-Content-Type-Options Missing	Low
Information Disclosure	Informational

Detailed Vulnerability Findings

1. Cross-Domain Misconfiguration

Risk Level:
Medium

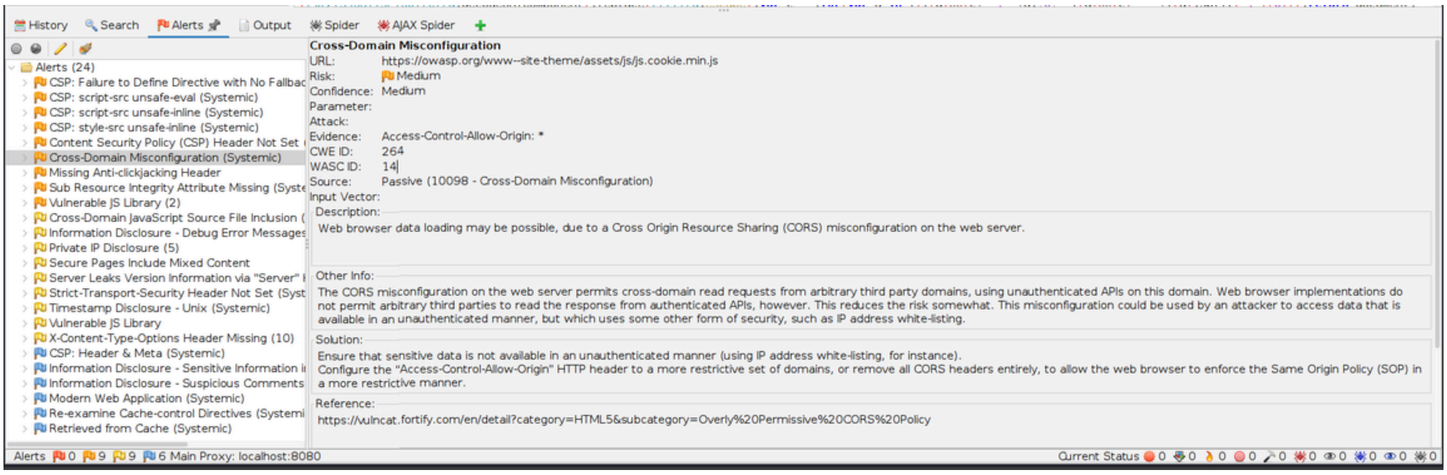
Affected URL:
https://owasp.org/www--site-theme/assets/js/js.cookie.min.js

Description:
Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server

Impact:
This issue may affect user trust and data integrity if exploited.

Evidence:
Access-Control-Allow-Origin: *

Remediation Steps:
Ensure that sensitive data is not available in an unauthenticated manner (using IP address whitelisting, for instance). Configure the “Access-Control-Allow-Origin” HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.



```
Alerts (24)
  CSP: Failure to Define Directive with No Fallbac
  CSP: script-src unsafe-eval (Systemic)
  CSP: script-src unsafe-inline (Systemic)
  CSP: style-src unsafe-inline (Systemic)
  Content Security Policy (CSP) Header Not Set (
  Cross-Domain Misconfiguration (Systemic)
  Missing Anti-clickjacking Header
  Sub Resource Integrity Attribute Missing (Syste
  Vulnerable JS Library (2)
  Cross-Domain JavaScript Source File Inclusion (
  Information Disclosure - Debug Error Messages
  Private IP Disclosure (5)
  Secure Pages Include Mixed Content
  Server Leaks Version Information via "Server" I
  Strict-Transport-Security Header Not Set (Syst
  Timestamp Disclosure - Unix (Systemic)
  Vulnerable JS Library
  X-Content-Type-Options Header Missing (10)
  CSP: Header & Meta (Systemic)
  Information Disclosure - Sensitive Information i
  Information Disclosure - Suspicious Comments
  Modern Web Application (Systemic)
  Re-examine Cache-control Directives (Systemi
  Retrieved from Cache (Systemic)
```

Cross-Domain Misconfiguration

```
URL:          https://owasp.org/www--site-theme/assets/js/js.cookie.min.js
Risk:         Medium
Confidence:   Medium
Parameter:
Attack:
Evidence:     Access-Control-Allow-Origin: *
CWE ID:       264
WASC ID:      14
Source:       Passive (10098 - Cross-Domain Misconfiguration)
Input Vector:
```

Description:
Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.

Other Info:
The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.

Solution:
Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).
Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

Reference:
https://vulncat.fortify.com/en/detail?category=HTML5&subcategory=Overly%20Permissive%20CORS%20Policy

```
Alerts 0 9 9 6 Main Proxy: localhost:8080          Current Status 0 0 0 0 0 0 0 0 0 0
```

2. Content Security Policy (CSP) Header Not Set

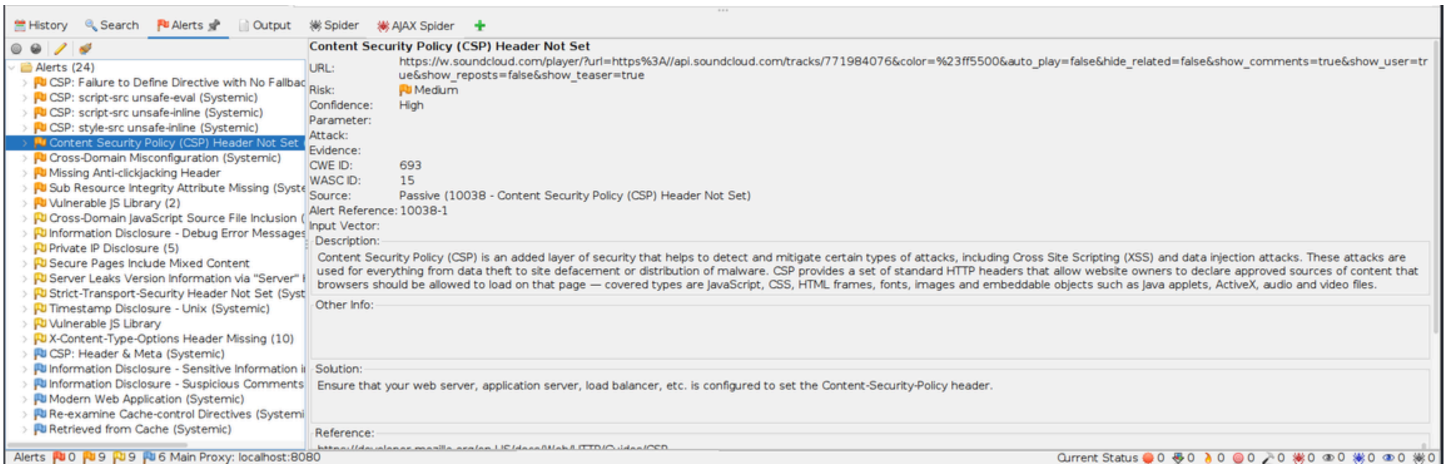
Risk Level:
Medium

Affected URL:
https://owasp.org/www--site-theme/assets/js/js.cookie.min.js

Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross-Site-Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page covered types are Javascript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files

Impact:
This issue may negatively impact user trust and website credibility.

Remediation Steps:
Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.



```
Alerts (24)
  CSP: Failure to Define Directive with No Fallbac
  CSP: script-src unsafe-eval (Systemic)
  CSP: script-src unsafe-inline (Systemic)
  CSP: style-src unsafe-inline (Systemic)
  Content Security Policy (CSP) Header Not Set
  Cross-Domain Misconfiguration (Systemic)
  Missing Anti-clickjacking Header
  Sub Resource Integrity Attribute Missing (Syste
  Vulnerable JS Library (2)
  Cross-Domain JavaScript Source File Inclusion (
  Information Disclosure - Debug Error Messages
  Private IP Disclosure (5)
  Secure Pages Include Mixed Content
  Server Leaks Version Information via "Server" I
  Strict-Transport-Security Header Not Set (Syst
  Timestamp Disclosure - Unix (Systemic)
  Vulnerable JS Library
  X-Content-Type-Options Header Missing (10)
  CSP: Header & Meta (Systemic)
  Information Disclosure - Sensitive Information i
  Information Disclosure - Suspicious Comments
  Modern Web Application (Systemic)
  Re-examine Cache-control Directives (Systemi
  Retrieved from Cache (Systemic)
```

```
Content Security Policy (CSP) Header Not Set
URL:             https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=false&hide_related=false&show_comments=true&show_user=tr
                 ue&show_reposts=false&show_teaser=true
Risk:            Medium
Confidence:      High
Parameter:
Attack:
Evidence:
CWE ID:          693
WASC ID:         15
Source:          Passive (10038 - Content Security Policy (CSP) Header Not Set)
Alert Reference: 10038-1
Input Vector:
Description:
 Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are
 used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that
 browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
Other Info:

Solution:
 Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Reference:
```

```
Alerts 0 9 9 6 Main Proxy: localhost:8080                Current Status 0 0 0 0 0 0 0 0 0 0
```

3. Missing Anti-clickjacking Header

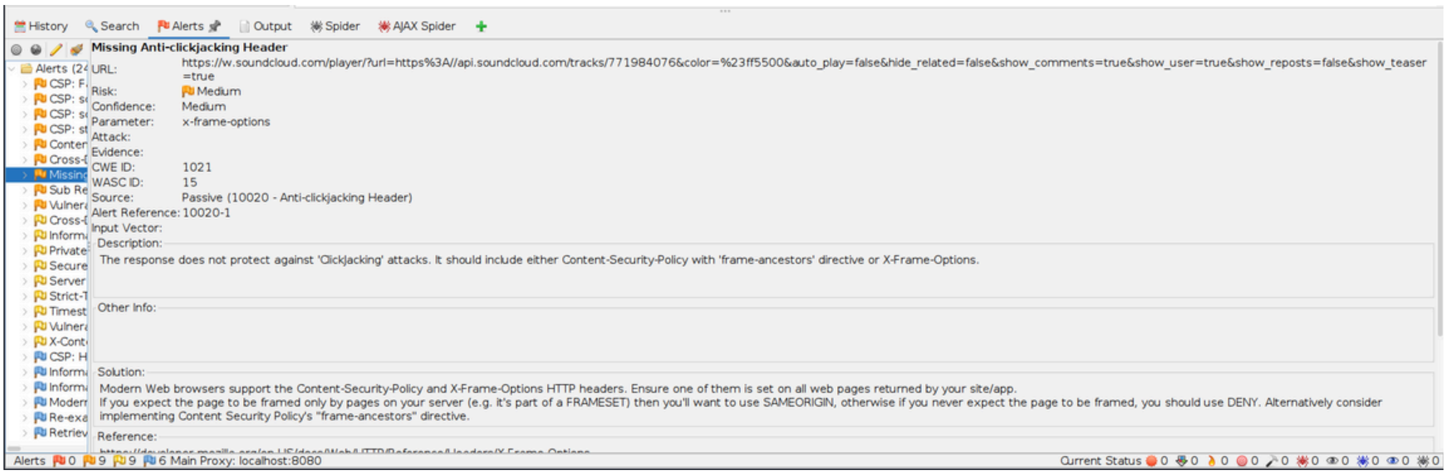
Risk Level:
Medium

Affected URL:
https://w.soundcloud.com/player/?url=https://api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=false&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true

Description:
The response does not protect against ‘Clickjacking’ attacks. It should include either Content-Security-Policy with ‘frame-ancestors’ directive or X-Frame Options.

Impact:
This vulnerability could expose the organization to compliance and reputational risks.

Remediation Steps:
Modern web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. If you expect the page to be framed only by pages on your server (e.g. it’s part of a FRAMESET) then you’ll want to use SAMEORIGIN, otherwise, if you never expect the page to be framed, you should use DENY. Alternatively, consider implementing Content Security Policy’s “frame-ancestors” directive.



Missing Anti-clickjacking Header

```
URL:             https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=false&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser
                 =true
Risk:            Medium
Confidence:      Medium
Parameter:       x-frame-options
Attack:
Evidence:
CWE ID:          1021
WASC ID:         15
Source:          Passive (10020 - Anti-clickjacking Header)
Alert Reference: 10020-1
Input Vector:
```

Description:
The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Other Info:

Solution:
Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.
If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

Reference:

Alerts 0 9 9 6 Main Proxy: localhost:8080   Current Status 0 0 0 0 0 0 0 0 0 0

4. X-Content-Type-Options Header Missing

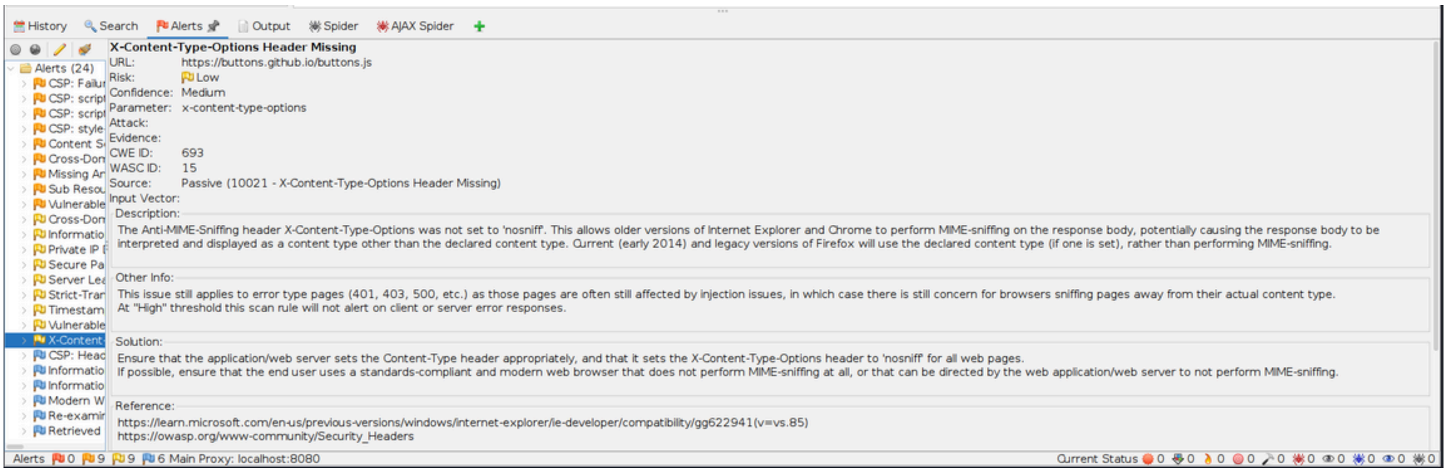
Risk Level:
Low

Affected URL:
https://buttons.github.io/buttons.js

Description:
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to ‘nosniff’. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Impact:
If left unresolved, this issue may increase the overall attack surface of the application.

Remediation Steps:
Ensure that the application/web server sets the Content-Type header appropriately and that it sets the X-Content-Type-Options header to ‘nosniff’ for all web pages. If possible, ensure that the end user uses a standard-compliant and modern web browser that does not perform MIME-sniffing at all or that can be directed by the web application/web server to not perform MIME-sniffing.



X-Content-Type-Options Header Missing
URL: https://buttons.github.io/buttons.js
Risk: Low
Confidence: Medium
Parameter: x-content-type-options
Attack:
Evidence:
CWE ID: 693
WASC ID: 15
Source: Passive (10021 - X-Content-Type-Options Header Missing)
Input Vector:

Description:
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Other Info:
This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.
At "High" threshold this scan rule will not alert on client or server error responses.

Solution:
Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.
If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

Reference:
https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
https://owasp.org/www-community/Security_Headers

Alerts 0 9 9 6 Main Proxy: localhost:8080 Current Status 0 0 0 0 0 0 0 0 0 0

5. Information Disclosure—Sensitive Information in URL

Risk Level:
Informational

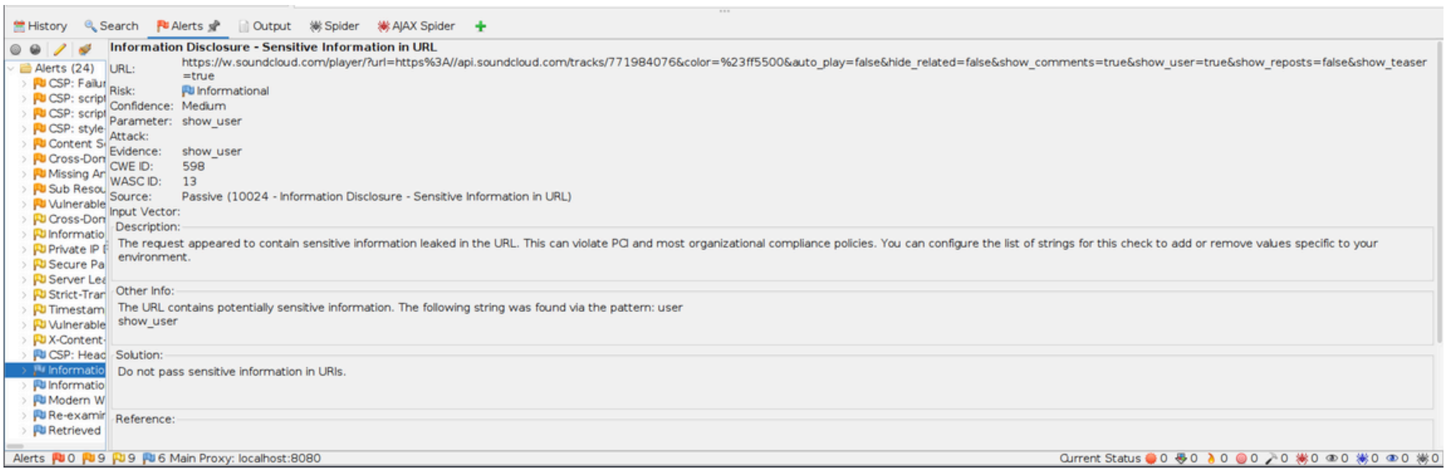
Affected URL:
https://w.soundcloud.com/player/?url=https://api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=false&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true

Description:
The request appeared to contain sensitive information leaked in the URL. This can violate PCI and most organizational compliance policies. You can configure the list of strings for this check to add or remove values specific to your environment

Evidence:
show_user

Impact:
This may lead to privacy and compliance concerns if sensitive parameters are exposed.

Remediation Steps:
Do not pass sensitive information in URIs



Information Disclosure - Sensitive Information in URL

URL:	https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=false&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true
Risk:	Informational
Confidence:	Medium
Parameter:	show_user
Attack:	
Evidence:	show_user
CWE ID:	598
WASC ID:	13
Source:	Passive (10024 - Information Disclosure - Sensitive Information in URL)
Input Vector:	

Description:
The request appeared to contain sensitive information leaked in the URL. This can violate PCI and most organizational compliance policies. You can configure the list of strings for this check to add or remove values specific to your environment.

Other Info:
The URL contains potentially sensitive information. The following string was found via the pattern: user
show_user

Solution:
Do not pass sensitive information in URIs.

Reference:

Alerts 0 9 9 6 Main Proxy: localhost:8080   Current Status 0 0 0 0 0 0 0 0 0 0

Conclusion

This vulnerability assessment identified several security misconfigurations primarily related to missing or improperly configured security headers. No high-risk or critical vulnerabilities were observed during the assessment.

Although the identified issues do not indicate immediate compromise, they may increase the risk of client-side attacks and information exposure if left unaddressed. Implementing the recommended remediation steps will improve the overall security posture of the application and align it with industry best practices.

The assessment was limited to passive and non-intrusive testing and should be periodically repeated to ensure continued security as the application evolves.

GitHub Repository Reference

All assessment documentation, screenshots, and the final report have been documented in a public GitHub repository.

Repository Name: FUTURE_CS_01
Repository Link: *https://github.com/sohelmj16/FUTURE_CS_01*

The repository includes the report PDF, supporting evidence, and a README file describing the tested website, scope, and tools used.