Name	Rohan Kumar
Mobile Number	9508029539
Email	rm803629@gmail.com

Legend	
Severity Definitions	
High	This risk level indicates that successful exploitation of the vulnerability may result in a significant impact to the confidentiality, integrity or availability of the information accessible through the application or even the backend resources like databases, operating systems, etc. It may also lead to damage of reputation.
Medium	This risk level indicates that successful exploitation of the vulnerability may reveal information about the application and its underlying infrastructure that may be used by an attacker in conjunction with another vulnerability to gain further access.
Low	This risk level indicates that successful exploitation of the vulnerability may result in little or no loss of sensitive information, but may enable an attacker to gain enough information regarding the application and its underlying infrastructure, which he/she may use to narrow down the attack approach.

Vulnerability Assessment Findings		
Sr. No.	**Findings**	**Risk Rating**
<Sr. No>	<Name of the Finding>	<High, Medium or Low>
	Below are some added finding for reference.	
1	**Content Security Policy (CSP) Header Not Set**	**High**
2	**Missing Anti-clickjacking Header**	**Meduim**
3	**Vulnerable JS Library(jQuery v3.1.1)**	**Meduim**
4	**Cookie without SameSite Attribute**	**Low**
5	**X-Content-Type-Options Header Missing**	**Low**
6	**Charset Mismatch (Header Versus Meta Charset)**	**informational**
7	**Information Disclosure - Sensitive Information in URL**	**informational**
8	**Information Disclosure - Suspicious Comments**	**informational**
9	**Modern Web Application**	**informational**
10	**Session Management Response Identified**	**informational**

Sr. No.	Findings	Severity	Evidences	Description
<Sr. No>	**<Name of the Finding>**	**<High, Medium or Low>**	<Page Number of attached POCs>	<Description of the Finding>

Below are some added finding for reference.

1	**Content Security Policy (CSP) Header Not Set**	**High**	A1	The application does not implement a **Content Security Policy (CSP)** header. CSP is a security mechanism that helps detect and mitigate attacks such as **Cross-Site Scripting (XSS)** and **data injection**.

2	**Missing Anti-clickjacking Header**	**High**	A2	The application does not implement headers to prevent **clickjacking**. Clickjacking is an attack where a malicious website embeds the target site in an invisible or disguised iframe, tricking users into performing unintended actions (like clicking buttons, submitting forms, or approving transactions). Without anti-clickjacking protections, attackers can hijack user interactions and compromise sensitive operations.
3	**Vulnerable JS Library**	Meduim	A3	The application is using JavaScript libraries with **known vulnerabilities**. Attackers can exploit these flaws to execute arbitrary code, perform XSS, bypass input validation, or steal sensitive information. Outdated libraries expand the attack surface and undermine overall application security.

4	**Cookie without SameSite Attribute**	Low	A4	The application issues cookies without the **SameSite** attribute. SameSite is a browser protection that controls whether cookies are sent with cross-site requests. Without it, cookies may be exposed to **Cross-Site Request Forgery (CSRF)** attacks, where an attacker tricks a user's browser into sending authenticated requests without their knowledge.
5	**X-Content-Type-Options Header Missing**	Low	A5	The application does not send the **X-Content-Type-Options** header. This header prevents browsers from performing **MIME-sniffing** of content types. Without it, some browsers may interpret files as a different MIME type than declared, which can lead to **Cross-Site Scripting (XSS)** or file execution vulnerabilities.

6	**Charset Mismatch (Header Versus Meta Charset)**	Informational	The character set declared in the **HTTP response header** does not match the charset declared in the **HTML meta tag**. This mismatch can cause browsers to interpret the page differently, leading to inconsistent rendering. In some cases, it may open avenues for encoding-related **XSS attacks**.
7	**Information Disclosure - Sensitive Information in URL**	Informational	Sensitive data such as **tokens and session identifiers** were observed in query parameters. URLs are often logged in server logs, browser history, proxy logs, and may also be leaked via referrer headers to third-party domains.

8	**Information Disclosure - Suspicious Comments**	Informational	The application contains HTML/JS comments that may disclose sensitive information about backend logic, credentials, or system details. Such information could aid attackers in exploiting the system.
9	**Modern Web Application**	Informational	The application uses modern single-page application frameworks. While this is not a vulnerability by itself, these frameworks rely heavily on client-side logic which may hide attack surfaces not visible in server-side testing.

10	**Session Management Response Identified**	Informational	The application uses **session cookies** for user authentication and state management. While not inherently a vulnerability, improper session cookie configurations may weaken session security.

Web Application Security Testing		
Affected Systems / URL	**Vulnerable Parameter(s)**	**Impact**
<Affected URL>	<Vulnerable Parameter if any otherwise put NA>	<Impact of the Finding>
http://192.168.0.104:8080/	N/A (applies to the entire application response headers).	Higher risk of Cross-Site Scripting (XSS) attacks exists since malicious scripts can run unchecked, leading to significant vulnerabilities. These injected scripts could potentially steal sensitive data such as cookies and tokens. Furthermore, attackers may exploit this risk by loading untrusted external content, including malware and phishing payloads. This situation not only poses a threat to user data but also allows for site defacement or unauthorized redirection, undermining the integrity and trustworthiness of the affected website.

http://192.168.0.104:8080/	N/A (applies to the entire response).	An attacker can create a malicious webpage that loads the target application in a hidden or transparent iframe. They can then trick a logged-in user into clicking on invisible elements of the application. This could lead to the user unintentionally performing sensitive actions, such as deleting account data, changing their password, or authorizing financial transactions
http://192.168.0.104:8080/	N/A (applies to library imports).	Cross-Site Scripting (XSS) attacks can be executed through vulnerable library functions, allowing attackers to exploit known bugs and bypass client-side security controls. This vulnerability poses significant risks, as it can lead to session hijacking, data theft, or even malware injection. The threat is particularly heightened when the vulnerable library is widely used across the application, increasing the potential impact of the attack. (jQuery v3.1.1)

http://192.168.0.104:8080/	PHPSESSID (session cookie).	1.Attackers can leverage CSRF to perform unauthorized actions on behalf of an authenticated user. 2.May allow data modification, unauthorized transactions, or privilege escalation. 3.Reduces overall cookie security if combined with missing Secure or HttpOnly flags.
http://192.168.0.104:8080/	N/A (applies to all responses).	1.Attackers may upload files with malicious content disguised as a safe file type (e.g., .jpg interpreted as .html). 2.Browsers may execute malicious content leading to XSS attacks. 3.Increases risk of content spoofing and drive-by attacks.

http://192.168.0.104:8080/	N/A	1.Pages may render incorrectly in some browsers. 2.Can be abused in rare cases to bypass input validation and inject malicious content.
http://192.168.0.104:8080/	authz_token	1.Leakage of tokens in logs, bookmarks, analytics tools. 2.May allow attackers to hijack sessions. 3.Increases overall risk of information disclosure.

http://192.168.0.104:8080/	N/A	1.Attackers may learn about internal functionality, hidden files, or credentials. 2.Could lead to privilege escalation or targeted attacks.
http://192.168.0.104:8080/	N/A	1.If not securely coded, DOM-based XSS or logic flaws may occur. 2.Complex client-side apps may bypass traditional security controls.

http://192.168.0.104:8080/	PHPSESSID	1.If cookies lack Secure, HttpOnly, or SameSite flags, attackers may steal sessions. 2.Weak session handling can lead to hijacking or replay attacks.

Recommendation	Reference
<Recommendation of the Finding>	<Referring URL of the Finding>

1.Add a **Content-Security-Policy** header in all HTTP responses. Example baseline: Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors 'none'; base-uri 'self';	https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

Option 1:Add one of the following headers in all HTTP responses: X-Frame-Options: DENY → Blocks all framing. X-Frame-Options: SAMEORIGIN → Allows framing only by same domain Option 2: Alternatively, use CSP with a frame-ancestors directive: Content-Security-Policy: frame-ancestors 'self'; Option 3: Test to ensure legitimate functionality (e.g., embedding allowed in trusted domains if required). Option 4: Escaping All User Supplied Input	https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html

1.Identify all JavaScript libraries in use (e.g., jQuery, Bootstrap, Owl Carousel, Magnific Popup). 2.Verify their versions against vulnerability databases (e.g., NVD , Snyk). 3.Upgrade libraries to the latest stable and secure versions. 4.Replace unsupported libraries with actively maintained alternatives. 5.Implement dependency management tools (e.g., npm, yarn, OWASP Dependency-Check) to monitor vulnerabilities continuously.	https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

1.Add the SameSite attribute to cookies. Use:
SameSite=Strict → Best protection, blocks all cross-site cookie sending.
SameSite=Lax → Allows cookies in top-level navigation only.

2.Always combine with: https://owasp.org/www-community/controls/SameSite
Secure flag → Ensures cookies are sent only over HTTPS.
HttpOnly flag → Prevents access from JavaScript.

3.Example secure cookie setting: Set-Cookie: PHPSESSID=<value>; Path=/; Secure; HttpOnly; SameSite=Strict

1.Set the header:
X-Content-Type-Options: nosniff

2.Apply this header for all static and dynamic resources (HTML, CSS, JS, images, etc.). https://owasp.org/www-project-secure-headers/

3.Ensure that correct MIME types are configured on the server.

1.Ensure charset is consistent across HTTP headers and HTML meta tags.

2.Use UTF-8 universally as it is widely supported.

https://owasp.org/www-community/vulnerabilities/Charset_Mismatch

1.Do not send sensitive data in query parameters.

2.Use POST body or Authorization headers instead.

3.Apply secure session management techniques.

https://portswigger.net/kb/issues/00400400_password-returned-in-url-query-string

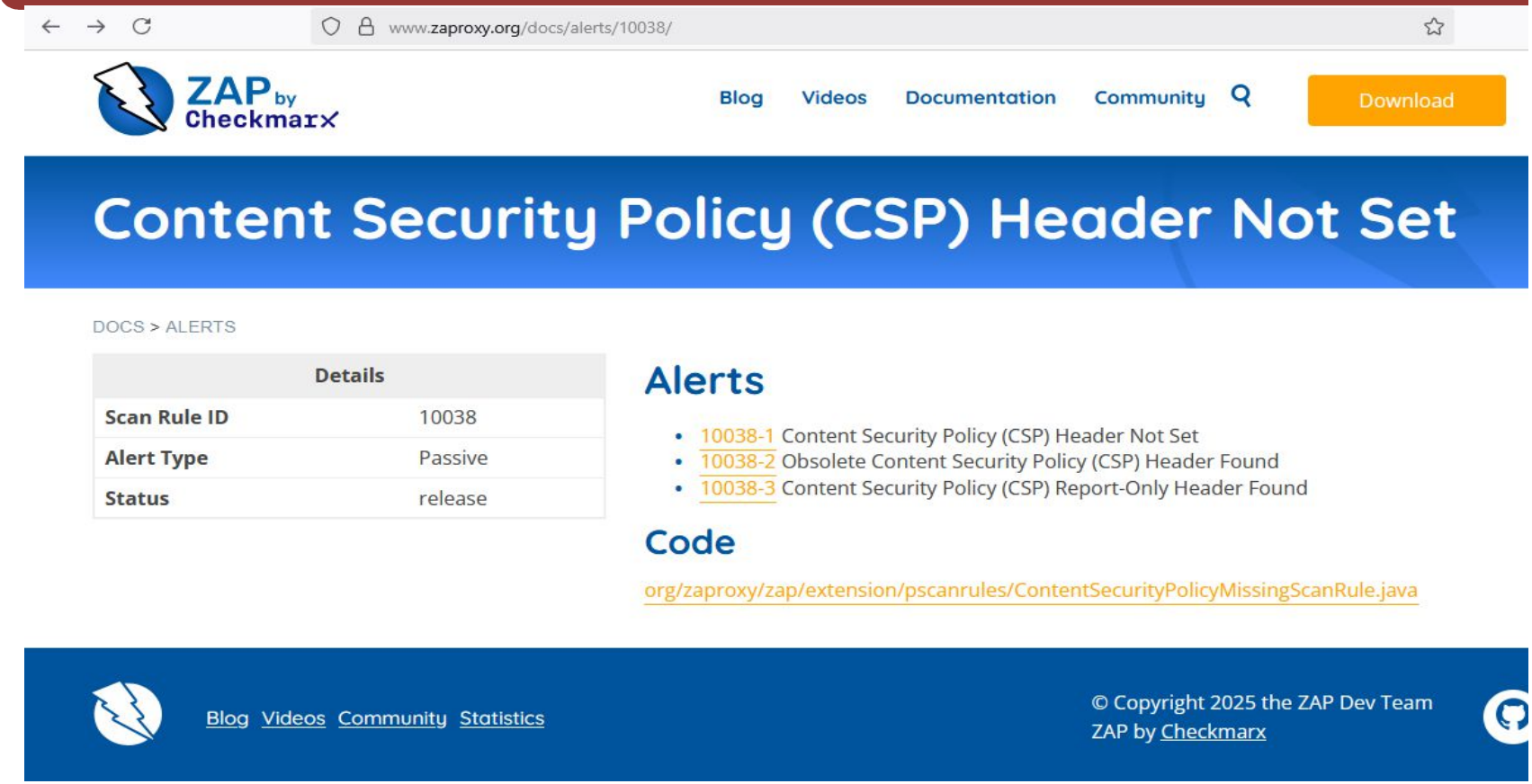
1.Remove unnecessary comments in production code. 2.Never expose credentials, API keys, or internal notes in client-side code.	https://owasp.org/www-community/vulnerabilities/Information_Leakage
1.Keep frameworks and dependencies updated. 2.Perform dedicated testing for DOM-based XSS and client-side security.	https://owasp.org/www-community/attacks/DOM_Based_XSS

1.Ensure session cookies have:
HttpOnly flag
Secure flag
SameSite attribute

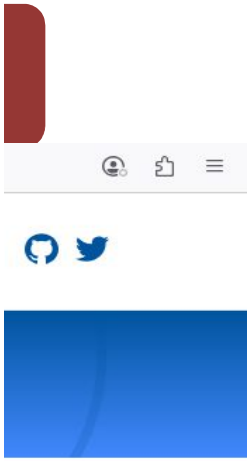
https://owasp.org/www-project-cheat-sheets/cheatsheets/Session_Management_Cheat_Sheet.html

2.Implement session timeout and regeneration on login.

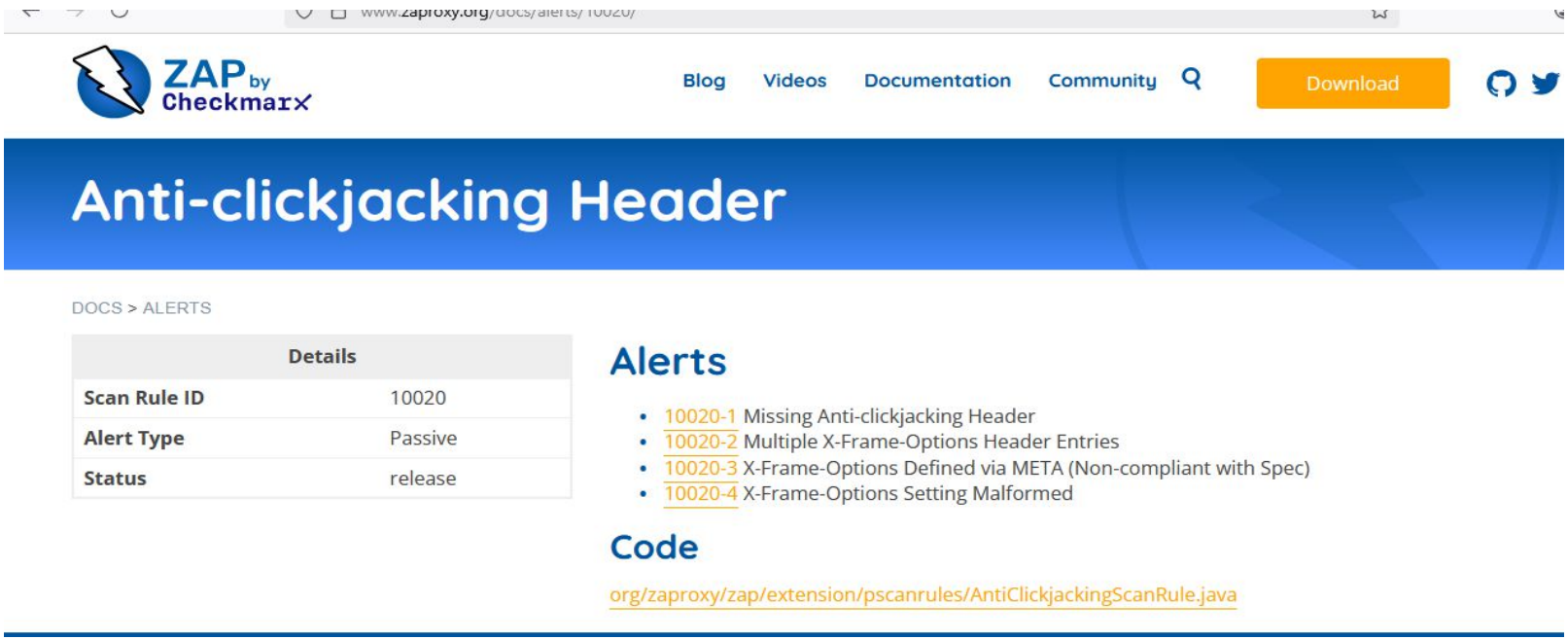
A1. Content Security Policy (CSP) Header Not Set



HTTP/1.1 200
Set-Cookie: PHPSESSID=4482C0D7F8F2ACCA0E0CF36AF6AD3C75; Path=/; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Date: Sun, 24 Aug 2025 10:34:37 GMT
Content-Length: 121899



A1. Missing Anti-clickjacking Header



HTTP/1.1 200
Content-Type: text/html;charset=ISO-8859-1
Date: Sun, 24 Aug 2025 10:34:37 GMT

A3. Valnerable JS Library

Vulnerable JS Library

DOCS > ALERTS

Details	
Alert ID	10003
Alert Type	Passive
Status	release
Risk	Medium
CWE	1395
WASC	
Technologies Targeted	All
Tags	CWE-1395 OWASP_2017_A09 OWASP_2021_A06 POLICY_DEV_STD POLICY_PENTEST POLICY_QA_STD
More Info	Scan Rule Help

Summary

The identified library appears to be vulnerable.

Solution

Upgrade to the latest version of the affected library.

Other Info

The identified library ExampleLibrary.js, version 13.3.7 is vulnerable.

References

- https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_C

Code

org/zaproxy/addon/retire/RetireScanRule.java

<script src="/vendor/jquery/jquery-1.12.4.min.js"></scrip

Caption for A3 sheet:

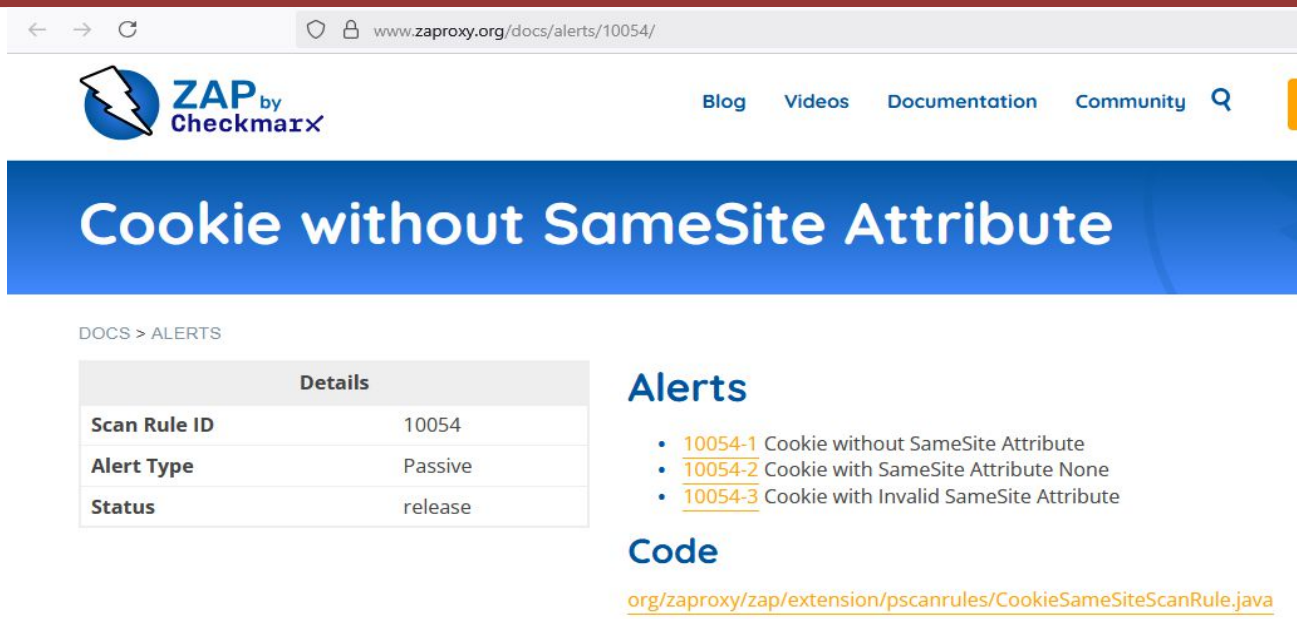
Outdated jQuery library (v1.12.4) included in the applicati

Components/

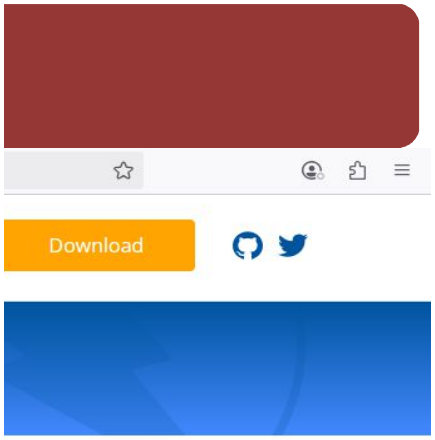
t>

on.

A4. Cookie without SameSite Attribute

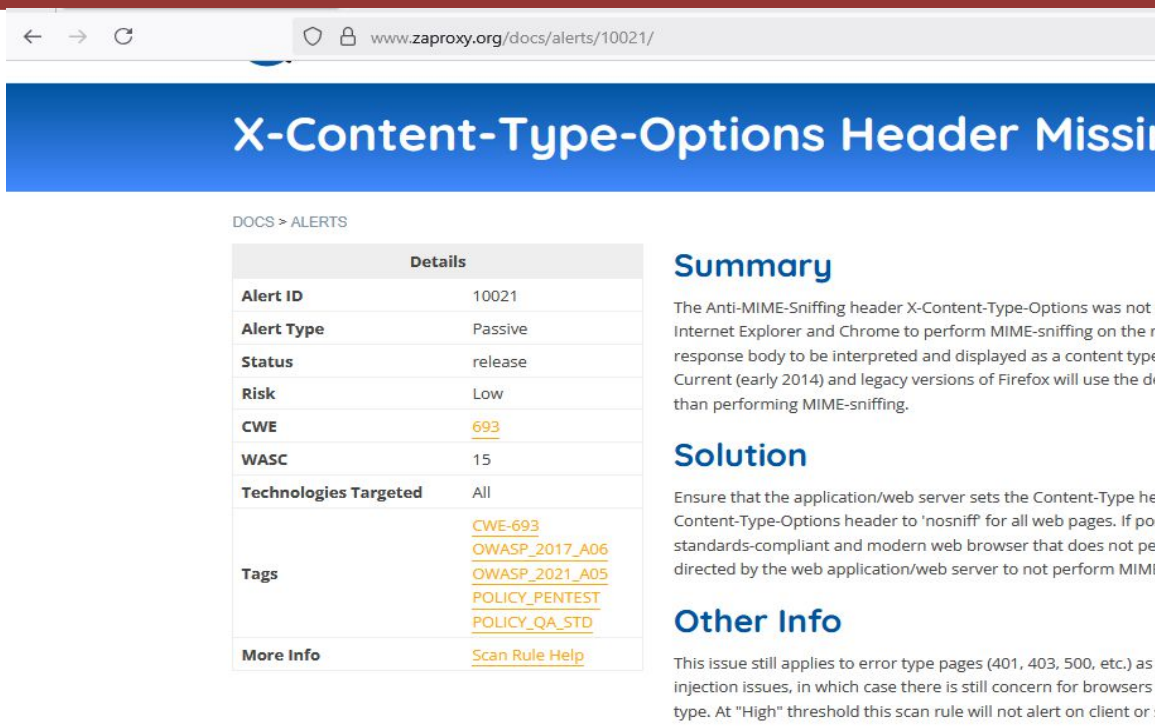


Set-Cookie: PHPSESSID=4482C0D7F8F2ACCA0E0CF36AF6A
Caption for A4 sheet:
Session cookie without SameSite attribute.”



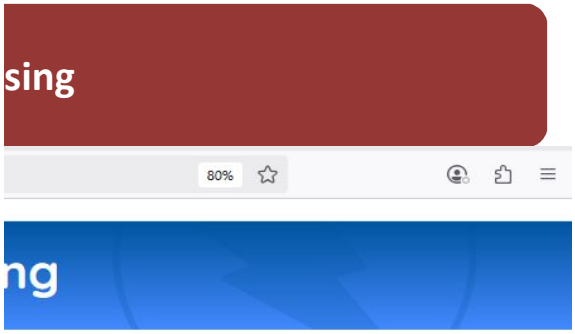
.D3C75; Path=/; HttpOnly

A5. X-Content -Type-Options Header Mis



HTTP/1.1 200
Content-Type: text/html;charset=ISO-8859-1

Caption for A5 sheet:
Response without X-Content-Type-Options header



set to 'nosniff'. This allows older versions of
response body, potentially causing the
e other than the declared content type.
eclared content type (if one is set), rather

eader appropriately, and that it sets the X-
ssible, ensure that the end user uses a
erform MIME-sniffing at all, or that can be
E-sniffing.

those pages are often still affected by
sniffing pages away from their actual content
server error responses.