Web Application Security Assessment Report

FullEnrich

Version 1.0

29th May 2024

Prepared By: Hoang Nhan Le

Prepared For: FullEnrich

Executive Summary

Context

The objective of this security assessment is to identify and exploit the vulnerabilities of the web application, assess the security risks and provide recommendations in order to mitigate the identified risks.

Project Scope

The **web application** assessment was performed in the **Dev** environment. The host of the target application was:

- app.dev.fullenrich.com

Key Findings

A total **four** low-risk findings were found and **seven** informational-risk issues also identified. The following is a summary of the main finding:

- **Strict Transport Security Not Enforced** allowed an attacker to force the web browser to communicate with the server over unencrypted HTTP protocol.

Summary of Findings

The web application has several low-priority vulnerabilities that could potentially impact the integrity and confidentiality of the information within the system. It might be advisable to address these issues to ensure the system's long-term reliability. Although these vulnerabilities are not an immediate threat to user or business security, resolving them would be beneficial if the improvement in the organization's security posture justifies the cost of the solution.

Strict Transport Security Not Enforced: Missing HTTP Strict Transport Security (HSTS) header may lead to man-in-the-middle downgrade attack. An attacker can force the browser to use an unencrypted HTTP protocol. Therefore, user data will be transferred in plain-text communication.

Strategic Recommendations

- **Configure the web application to use HTTP Strict Transport Security (HSTS) header** to avoid from man-in-the-middle downgrade attack.

Security Assessment History

Version 1.0

Initial Security Assessment (29th May 2024)

A security assessment report generated indicates the result of the first test. By providing this report, the assessor acknowledges that there are 11 security issues open in the application. It is recommended that remedial work be taken to ensure the future operation of the system.

Table of Contents

1 Overview

1.1 Assessment Methodology

Phase 1 - Information Gathering	The security assessment was started by collecting information of the target application from various sources. The information, which was publicly available on the Internet, includes the network infrastructure, domain name service, security systems in use, open services, etc. The information would help the consultants to understand the target environment and plan for further assessment.
Phase 2 - Vulnerability Identification & Prioritization	After information gathering, assessors would try to identify any vulnerability on the application through automated scanning tools and manual inspection. Numerous test cases were conducted. For the Web Application assessment, **OWASP Top 10 vulnerability** would be covered. ● Client-side attacks Examine if there is any insecure data handling, such as unencrypted data storage for sensitive information, insecure file caching, etc. ● Network-side attacks Examine the traffic between the application and server and identify if there is any information leakage (e.g. personal data leakage, etc.) ● Server-side attacks Examine the traffic between the application and backend server and identify if there is any possible data manipulation between the client and server (e.g. injection, data tampering, session hijacking, etc) The risks discovered will be correlated with the results in static security assessment to produce a more accurate result. The results were collected, reviewed and prioritized for further exploit.
Phase 3 - Research & Development	In this phase, the consultants conducted research on the vulnerabilities identified on the target application and developed the attack approaches, tools, scripts, etc. and prepared for exploiting the vulnerabilities.
Phase 4 - Exploitation	With the findings in the research and development phase, the consultants would then carry exploits on the target. This phase involved the use of real-world hacker tools and scripts to simulate attacks on the vulnerabilities. In this phase, a higher level of privileged or access to sensitive information can be achieved.
Phase 5 - Post-Exploitation	After exploitation, the consultants might gain privileged access to the target application. the consultants would explore further opportunities to see if it is possible to access other systems through the privileged access.
Phase 6 - Risk Analysis and Reporting	The results of the security assessment were documented in detail in this report. The risk rating of each vulnerability was assessed. The result and the recommendations for remediation will be documented in the report. To cater for different readers, the report will be clearly sectioned to consist of executive-level reporting and technical

	reporting. The consultants shall endeavor to produce a report that is concise, well-structured and contain of solid recommendations and reproducible results.

1.2 Type of Tests

The test can be conducted in black-box, grey-box or white-box approach. According to **Open Source Security Testing Methodology Manual (OSSTMM)**, the three types of tests are defined as follows:

Type	Description
Black-box (Blind)	The assessor engages the target with no prior knowledge of its defenses, assets, or channels. The target is prepared for the audit, knowing in advance all the details of the audit. A black-box audit primarily tests the skills of the assessor. The breadth and depth of a blind audit can only be as vast as the assessor's applicable knowledge and efficiency allows.
Grey-box	The assessor engages the target with limited knowledge of its defenses and assets and full knowledge of channels. The target is prepared for the audit, knowing in advance all the details of the audit. A grey-box audit tests the skills of the assessor. The nature of the test is efficiency. The breadth and depth depend upon the quality of the information provided to the assessor before the test as well as the assessor's applicable knowledge.
WHITE-BOX	The assessor engages the target with limited knowledge of its defenses and assets and full knowledge of channels. The target is notified in advance of the scope and time frame of the audit but not the channels tested or the test vectors. A white-box audit tests the skills of the assessor and the target's preparedness to unknown variables of agitation. The breadth and depth depend upon the quality of the information provided to the assessor and the target before the test as well as the assessor's applicable knowledge.

For more details, please refer to OSSTMM v3.

1.3 Risk Level Classifications

This section of the report details the severity classification system used during the assessment according to **Common Vulnerability Scoring System Version 3.1 (CVSS v3.1)**:

Severity Rating	Description
CRITICAL	These issues imply an immediate, easily accessible threat of large-scale total compromise. As such, they should be resolved as a matter of urgency to ensure the business is not operating with an excessive level of IT related business risk.
HIGH	These issues imply an immediate threat of system compromise. As such, they should be resolved as soon as possible to ensure the business is not operating with an excessive level of IT related business risk.
MEDIUM	These issues should be resolved in a timely manner where possible; however, they can often be mitigated in the short term until appropriate resolutions can be put in place.
LOW	These issues should be resolved if the improvement in the organization's security posture would justify the cost of the solution. In general, solutions to low severity issues should be implemented once higher severity issues have been addressed.
INFORMATIONAL	These issues are included in the report for completeness.

For more details, please refer to CVSS v3.1.

1.4 Detailed Engagement Data

Name	details
company name	FullEnrich
Test Type	Web Application Security Assessment
Target	app.dev.fullenrich.com
Environment	Dev
Method	Black-box
Test Accounts	Self-created
Test Dates	26th May 2024 – 29th May 2024
Functions In-scope	All accessible functions

1.5 Port Scanning Results

app.dev.fullenrich.com

Available Port	Protocol	Service
80	tcp	http
443	tcp	https
2052	tcp	clearvisn
2053	tcp	knetd
2082	tcp	infowave
2083	tcp	radsec
2086	tcp	gnunet
2087	tcp	eli
2095	tcp	nbx-ser
2096	tcp	nbx-dir
8080	tcp	http-proxy
8443	tcp	https-alt
8880	tcp	cddbp-alt

The assessor found many open ports on the remote host. They could be real open ports or false-positive if a firewall had filtered the ports. The administrator should investigate whether the services are running on the host. For the security best practice, all unused services should be disabled.

1.6 Result Summary

The following table presents the total number of vulnerabilities found, sorted by severity. **Common Vulnerability Scoring System Version 3.1 (CVSS v3.1)** would be covered:

Risk Level		COUNT
CRITICAL		0
HIGH		0
MEDIUM		0
LOW		4
INFORMATIONAL		7
Grand Total		11

1.7 Table of Findings

The following table presents the total number of findings.

1.7.1 Web Application

Risk ID	Vulnerability	Risk level
A1	Strict Transport Security Not Enforced	Low
A2	Ineffective Session Termination	Low
A3	Clickjacking: X-Frame-Options Header Missing	Low
A4	Missing X-Content-Type-Options Header	Low
A5	Content Security Policy (CSP) Not Implemented	Informational
A6	Referrer Policy Header Missing	Informational
A7	X-XSS-Protection Header Missing	Informational
A8	Permissions-Policy Header Not Implemented	Informational
A9	Subresource Integrity (SRI) Not Implemented	Informational
A10	SameSite Cookie Not Implemented	Informational
A11	Password Field with Autocomplete Enabled	Informational

2 Detailed Findings

This section of the document is technical in nature and provides further detail about the items already discussed, for the purposes of remediation and risk-assessment.

2.1 Web Application

A1 Strict Transport Security Not Enforced	Low
CVSS v3.1 Base Score: 3.7	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Host/URL:

https://app.dev.fullenrich.com/app/

Issue Description:

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

The following picture shows that the Strict-Transport-Security header was not present in the response:

Request

Pretty Raw Hex

```
1  GET /app/ HTTP/2
2  Host: app.dev.fullenrich.com
3  Cookie: __cf_bm=mB57ddAa.kd2693W3VOqyoVn1(
4  User-Agent: Mozilla/5.0 (Windows NT 10.0;
5  Accept: text/html,application/xhtml+xml,ap
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate, br
8  Upgrade-Insecure-Requests: 1
9  Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
13 X-Pwnfox-Color: pink
14 Priority: u=1
15 Te: trailers
```

Response

Pretty Raw Hex Render

```
1  HTTP/2 200 OK
2  Date: Tue, 28 May 2024 02:47:59 GMT
3  Content-Type: text/html; charset=utf-8
4  X-Do-App-Origin: 0176b672-e958-4c1c-a1a2-fcd
5  Cache-Control: private
6  X-Do-Orig-Status: 200
7  Cf-Cache-Status: MISS
8  Last-Modified: Tue, 28 May 2024 02:47:59 GMT
9  Vary: Accept-Encoding
10 Server: cloudflare
11 Cf-Ray: 88aaec52ef5c1841-EWR
12
13 <!DOCTYPE html>
14 <html lang="en">
15
```

Recommendation:

Enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, optionally add the 'preload' flag to the HSTS header and submit the domain for review by browser vendors.

Reference:

HTTP Strict Transport Security https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

HSTS Preload Form https://hstspreload.org/

A2 Ineffective Session Termination	Low

CVSS v3.1 Base Score: 3.7	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Host/URL:

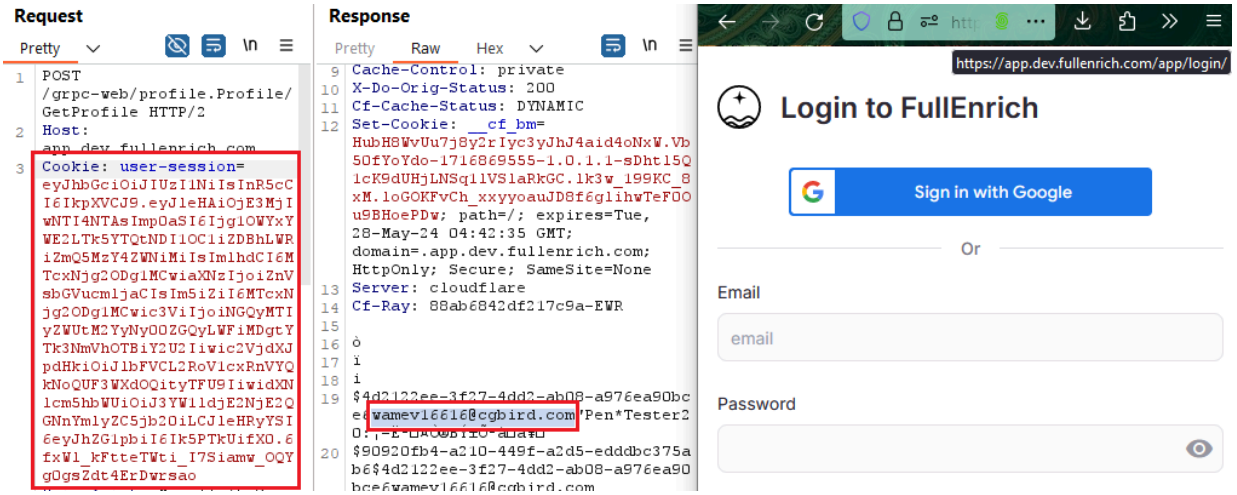
https://app.dev.fullenrich.com/app/

Issue Description:

The web applications failed to terminate authentication sessions when the user logged out of the applications. In the event that the previously authenticated session token was captured by an attacker, for example, by extracting the token from the user's workstation file system, the attacker could continue to use the stolen session token despite the user logging out of (terminating) that session. Likewise, if user's workstation were left unattended, it could result in the sessions to be hijacked by an attacker who has physical access to the workstation.

Although the logout function terminated the associated session on the client-side, the session remained valid on the server-side. Requests which were made after the logout function had been invoked and continued to be successful. This weakness might be caused by design and implementation levels and could be used by attackers to gain unauthorized access to the applications.

The assessor noticed the session cookie "user-session" could be reused after logging out.



Recommendation:

Ensure the application terminate the session on both the client and server side when logout.

Reference:

CWE-613: Insufficient Session Expiration https://cwe.mitre.org/data/definitions/613.html

A3 Clickjacking: X-Frame-Options Header Missing	Low

CVSS v3.1 Base Score: 3.1	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Host/URL:

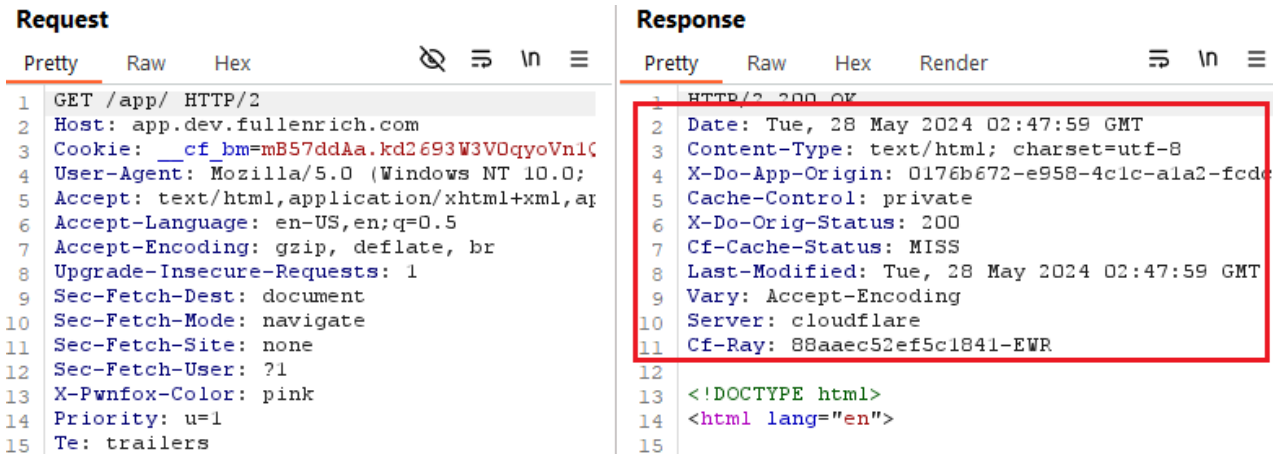
https://app.dev.fullenrich.com/app/

Issue Description:

Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

The server did not return an X-Frame-Options header which means that this website could be at risk of a clickjacking attack.

The following picture shows that the X-Frame-Options header was not present in the response:



Recommendation:

The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page inside a frame or iframe. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

Configure all pages of the web server to include an X-Frame-Options header. Consult Web references for more information about the possible values for this header.

Reference:

CWE-1021: Improper Restriction of Rendered UI Layers or Frames
https://cwe.mitre.org/data/definitions/1021.html

Clickjacking Defense
https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html

Clickjacking http://en.wikipedia.org/wiki/Clickjacking

The X-Frame-Options response header https://developer.mozilla.org/en-US/docs/HTTP/X-Frame-Options

A4 Missing X-Content-Type-Options Header	Low

CVSS v3.1 Base Score: 3.1	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Host/URL:

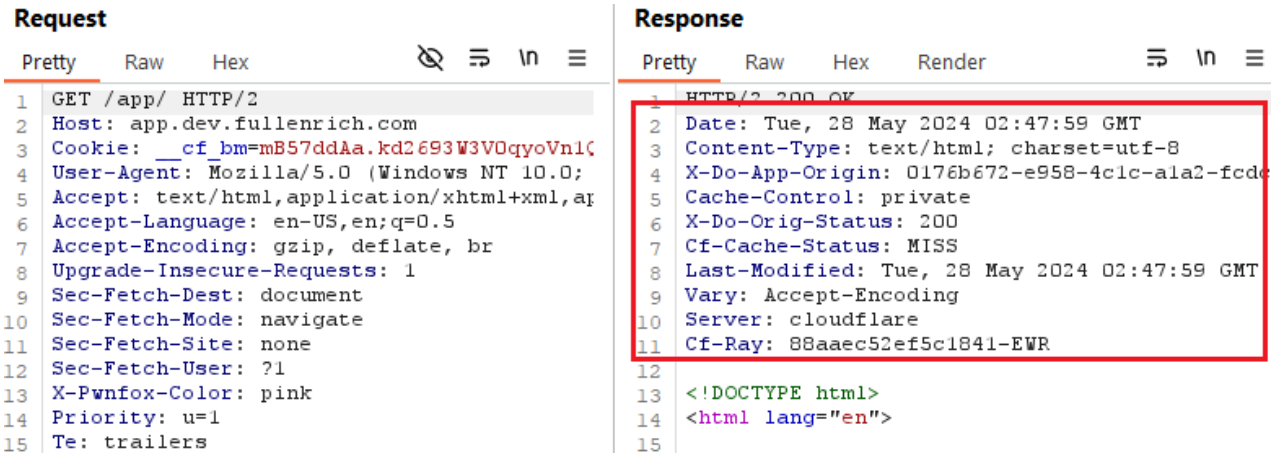
https://app.dev.fullenrich.com/app/

Issue Description:

The HTTP X-Content-Type-Options response header prevents the browser from MIME-sniffing a response away from the declared content-type.

The server did not return a correct “X-Content-Type-Options” header, which means that this website could be at risk of a Cross-Site Scripting (XSS) attack.

The following picture shows that the X-Content-Type-Options header was not present in the response:



Recommendation:

Configure the web server to include an “X-Content-Type-Options” header with a value of “nosniff”.

Reference:

X-Content-Type-Options
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

A5 Content Security Policy (CSP) Not Implemented	Informational
CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

Host/URL:

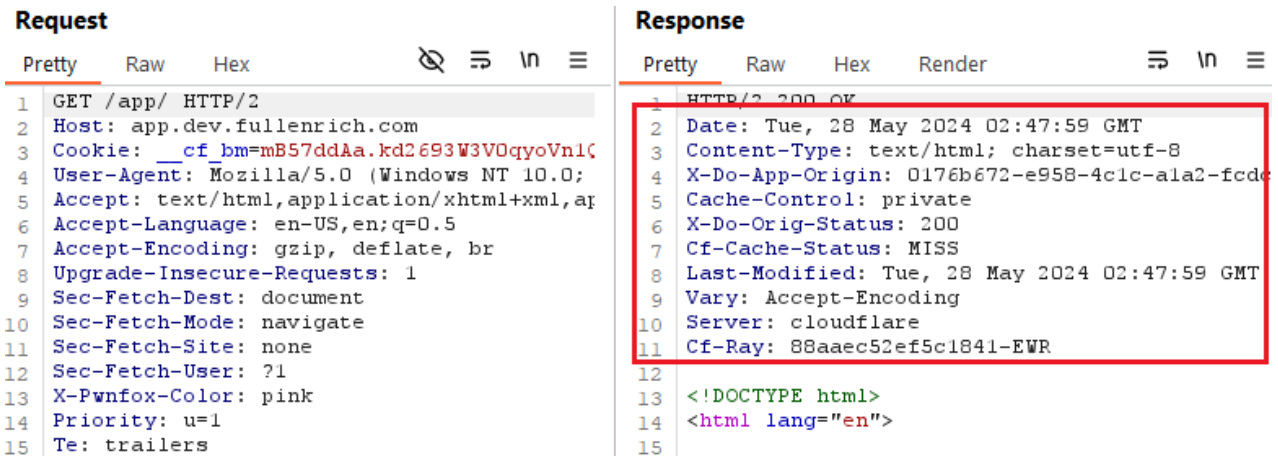
https://app.dev.fullenrich.com/app/

Issue 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.

It was detected that the web application did not implement Content Security Policy (CSP) as the CSP header is missing from the response.

The following picture shows that the Content-Security-Policy header was not present in the response:



Recommendation:

It is recommended to implement Content Security Policy (CSP) into the web application. Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control resources the user agent is allowed to load for that page.

Reference:

Content Security Policy (CSP) Not Implemented
https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/content-security-policy-csp-not-implemented/

Content Security Policy (CSP) https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

A6 Referrer Policy Header Missing	Informational
CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

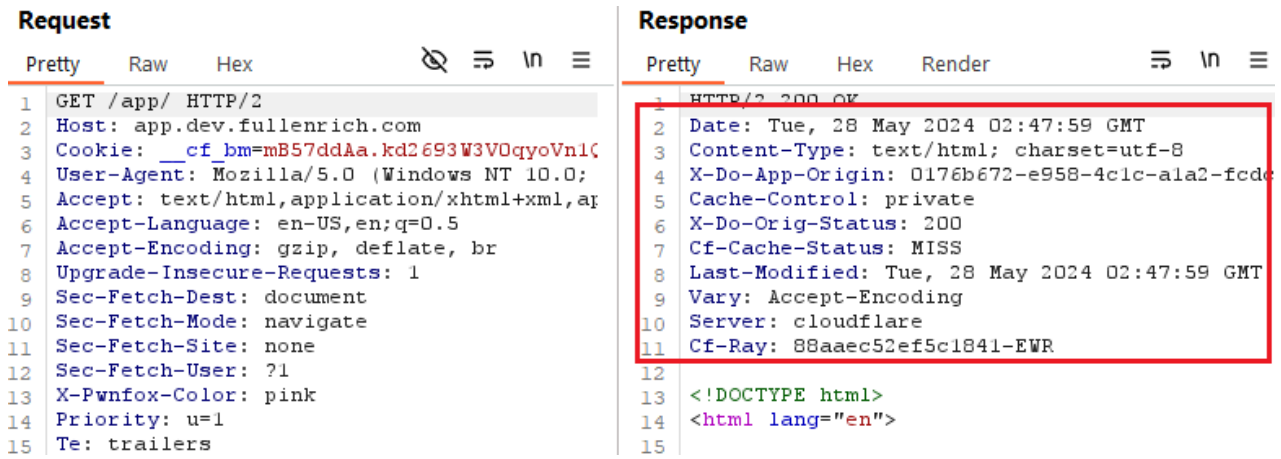
Host/URL:

https://app.dev.fullenrich.com/app/

Issue Description:

Referrer Policy controls behaviour of the Referrer header, which indicates the origin or web page URL the request was made from. There was no Referrer Policy header that may leak user's information to third-party sites.

The following picture shows that the Referrer-Policy header was not present in the response:



Recommendation:

Consider setting Referrer-Policy header to "strict-origin-when-cross-origin" or a stricter value.

Reference:

Referrer-Policy https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy

A7 X-XSS-Protection Header Missing	Informational
CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

Host/URL:

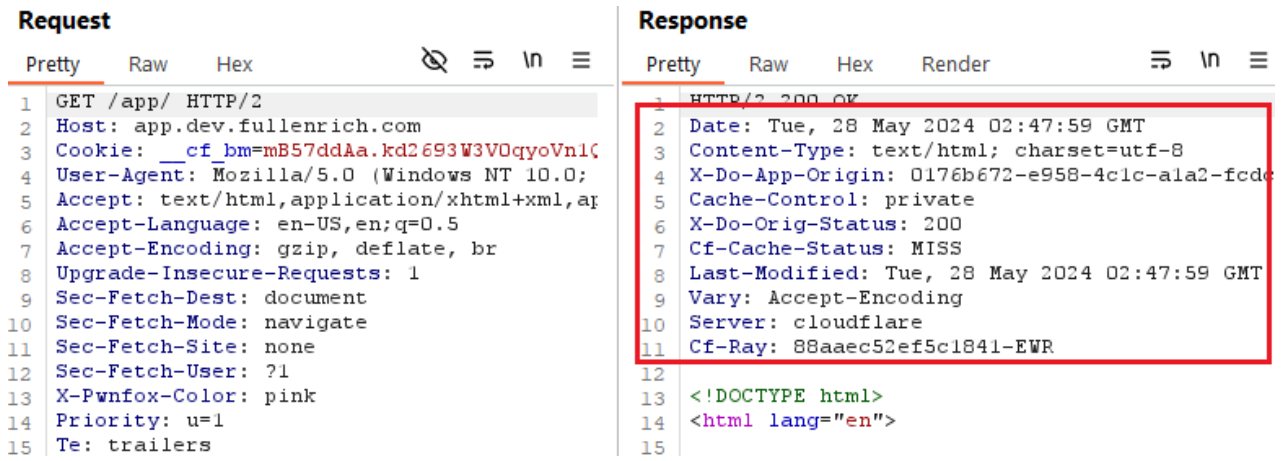
https://app.dev.fullenrich.com/app/

Issue Description:

The HTTP X-XSS-Protection response header is a feature of modern browsers that allows websites to control their XSS auditors.

The server is not configured to return a X-XSS-Protection header which means that any pages on this website could be at risk of a Cross-Site Scripting (XSS) attack. This URL is flagged as a specific example.

The following picture shows that the X-XSS-Protection header was not present in the response:



Recommendation:

Configure the application to include an “X-XSS-Protection” header with a value of “1; mode=block” on all pages.

Reference:

X-XSS-Protection https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

A8 Permissions-Policy Header Not Implemented	Informational

CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

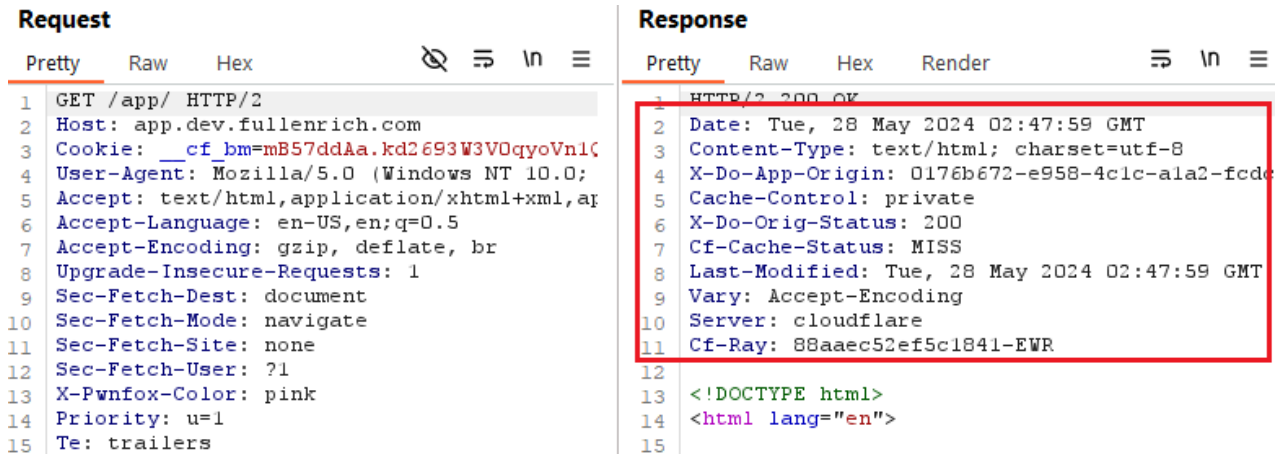
Host/URL:

https://app.dev.fullenrich.com/app/

Issue Description:

The Permissions-Policy header allows developers to selectively enable and disable use of various browser features and APIs.

The following picture shows that the Permissions-Policy header was not present in the response:



Recommendation:

Include Permissions-Policy header in the response.

Reference:

Permissions-Policy https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Permissions-Policy

Permissions-Policy https://www.validbot.com/header/Permissions-Policy.html

A9 Subresource Integrity (SRI) Not Implemented	Informational

CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

Host/URL:

https://app.dev.fullenrich.com/app/

Issue Description:

Subresource Integrity (SRI) is a security feature that enables browsers to verify that third-party resources they fetch (for example, from a CDN) are delivered without unexpected manipulation. It works by allowing developers to provide a cryptographic hash that a fetched file must match.

Third-party resources (such as scripts and stylesheets) can be manipulated. An attacker that has access or has hacked the hosting CDN can manipulate or replace the files. SRI allows developers to specify a base64-encoded cryptographic hash of the resource to be loaded. The integrity attribute containing the hash is then added to the <script> HTML element tag. The integrity string consists of a base64-encoded hash, followed by a prefix that depends on the hash algorithm. This prefix can either be sha256, sha384 or sha512.

The script loaded from the external URL specified in the Details section doesn't implement Subresource Integrity (SRI). It's recommended to implement Subresource Integrity (SRI) for all the scripts loaded from external hosts.

The assessor noticed that integrity attribute was not implemented in the following external script tags:

- https://r.wdfl.co/rw.js
- https://www.googletagmanager.com/gtag/js?id=G-Q3Q802382M

Recommendation:

Use the SRI Hash Generator link (from the References section) to generate a <script> element that implements Subresource Integrity (SRI).

For example, you can use the following <script> element to tell a browser that before executing the https://example.com/example-framework.js script, the browser must first compare the script to the expected hash, and verify that there's a match.

```
<script src="https://example.com/example-framework.js"
integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8
wC"
crossorigin="anonymous"></script>
```

Reference:

Subresource Integrity https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity

SRI Hash Generator https://www.srihash.org/

A10 SameSite Cookie Not Implemented	Informational

CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N

Host/URL:

https://app.dev.fullenrich.com/

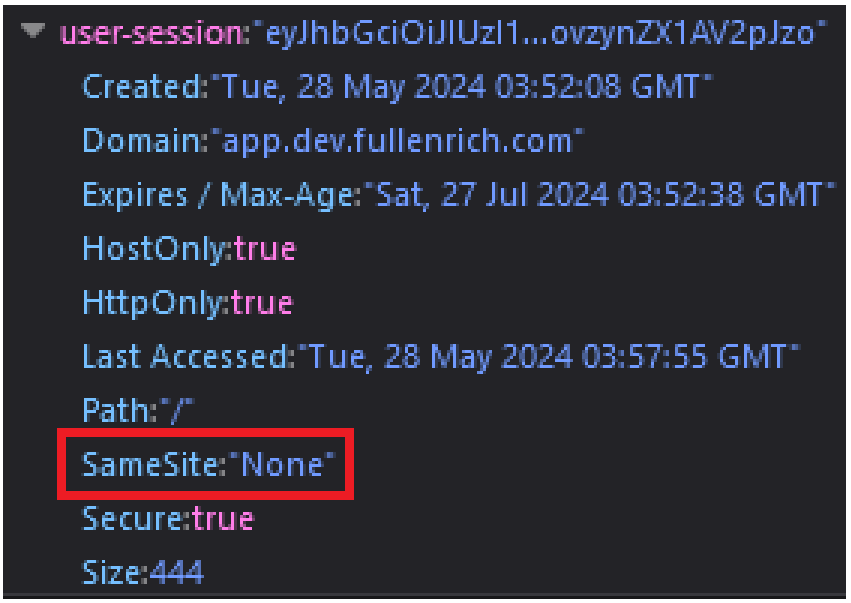
Issue Description:

Cookies are typically sent to third parties in cross origin requests. This can be abused to do CSRF attacks. Recently a new cookie attribute named SameSite was proposed to disable third-party usage for some cookies, to prevent CSRF attacks.

Same-site cookies allow servers to mitigate the risk of CSRF and information leakage attacks by asserting that a particular cookie should only be sent with requests initiated from the same registrable domain.

The following picture shows that the SameSite attribute was not implemented for the cookie:

- user-session



Recommendation:

The server can set a same-site cookie by adding the SameSite=... attribute to the Set-Cookie header. There are three possible values for the SameSite attribute:

- Lax: In this mode, the cookie will only be sent with a top-level get request.

  `Set-Cookie: key=value; SameSite=Lax`

- Strict: In this mode, the cookie will not be sent with any cross-site usage even if the user follows a link to another website.

  `Set-Cookie: key=value; SameSite=Strict`

- None: In this mode, the cookie will be sent with the cross-site requests. Cookies with SameSite=None must also specify the Secure attribute to transfer them via a secure context. Setting a SameSite=None cookie without the Secure attribute will be rejected by the browsers.

  ```
  Set-Cookie: key=value; SameSite=None; Secure
  ```

Reference:

CWE-1275: Sensitive Cookie with Improper SameSite Attribute
https://cwe.mitre.org/data/definitions/1275.html

A11 Password Field with Autocomplete Enabled	Informational

CVSS v3.1 Base Score: 0.0	**Vector String:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:N/I:N/A:N

Host/URL:

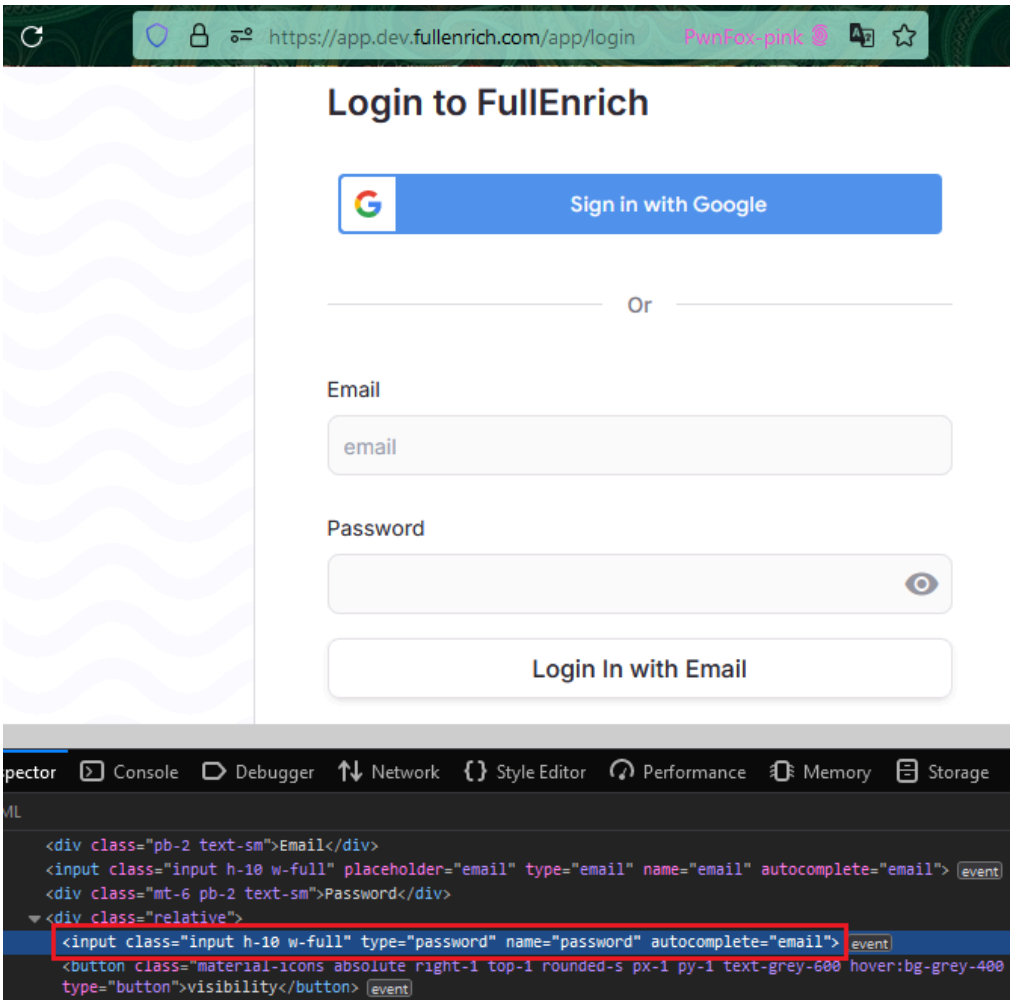
https://app.dev.fullenrich.com/app/login

Issue Description:

Most browsers have a facility to remember user credentials that are entered into HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application.

The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds a separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials.

The following picture shows that the autocomplete attribute was not set "off" in the HTML source code.



Recommendation:

To prevent browsers from storing credentials entered into HTML forms, include the attribute `autocomplete="off"` within the FORM tag (to protect all form fields) or within the relevant INPUT tags (to protect specific individual fields).

Please note that modern web browsers may ignore this directive. In spite of this there is a chance that not disabling autocomplete may cause problems obtaining PCI compliance.

Reference:

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
https://cwe.mitre.org/data/definitions/200.html

3 Appendix

3.1 Tools List

Tools Used	Description
Kali Linux	Pentest-focussed Linux distribution http://www.kali.org/
Nmap	Open source port scanner http://nmap.org/
acunetix	Web Vulnerability Scanner https://www.acunetix.com/vulnerability-scanner/
OWASP ZAP	Web vulnerability scanner https://www.zaproxy.org/
Nessus	Vulnerability scanning tool https://www.tenable.com/products/nessus/nessus-professional
Burp Suite Pro	Intercepting proxy and web application scanner http://portswigger.net/
Metasploit Framework	Exploit development and penetration testing framework http://www.metasploit.com/
FEROXBUSTER	Web content scanner https://www.kali.org/tools/feroxbuster/
NIKTO	Web server scanner https://tools.kali.org/information-gathering/nikto
sslscan	TLS/SSL scanner https://github.com/rbsec/sslscan
Firefox	Web browser https://www.mozilla.org/en-US/firefox/
Google Chrome	Web browser https://www.google.com/chrome/

3.2 Assessment Team

The following staff members are responsible for the assessment:

role	consultant	Title	qualification
Consultant	Hoang Nhan Le	Security Consultant	CREST CRT: 38099414 CREST CPSA: 38099414 OSCP: OS-101-050420