

# **Bitwarden Web App and Network Security Report**

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

# Table of Contents

<b>Bitwarden Web App and Network Security Report</b>	<b>1</b>
<b>Table of Contents</b>	<b>2</b>
<b>Summary</b>	<b>3</b>
<b>Issues</b>	<b>4</b>
EXT.L.01: TLS 1.0 and TLS 1.1 in Use (Low)	4
WEB.M.01: Cross-Site Scripting via External Resource (Medium)	4
WEB.M.02: Cleartext Storage of Sensitive Information in Memory (Medium)	4
WEB.M.03: Audit / Event Log Tampering (Medium)	5
WEB.L.01: Unauthorized Access to Premium Features (Low)	5
WEB.L.02: Session Token in URL (Low)	5
WEB.L.03: Missing Integrity Check For External Resources (Low)	6
WEB.L.04: Weak Password Policy for Vault Exports (Low)	6
WEB.BPI.01: Disable Software Name and Version Reporting (Info)	6
WEB.BPI.02: Restrict File Uploads (Info)	6
WEB.BPI.03: Remove Private IP Addresses from API Responses (Info)	6

# Summary

In July 2024, Bitwarden engaged with cybersecurity firm Fracture Labs to perform penetration testing and a dedicated audit of the Bitwarden web application and its related network components. A team of testers from Fracture Labs were tasked with preparing and executing the audit over two weeks to reach total coverage of the system under review.

Eleven issues were discovered during the audit. Six issues were resolved post-assessment. Three issues were determined not feasible to address. One issue is under planning and research.

This report was prepared by the Bitwarden team to cover the scope and impact of the issues found during the assessment and their resolution steps. For completeness and transparency, a copy of the Findings section within the report delivered by Fracture Labs has also been attached to this report.

# Issues

## [EXT.L.01: TLS 1.0 and TLS 1.1 in Use \(Low\)](#)

Status: Issue was fixed post-assessment.

An errant DNS entry was removed. The backend host was never available and didn't resolve, but Azure defaults happened to be on that returned the deprecated TLS response.

## [WEB.M.01: Cross-Site Scripting via External Resource \(Medium\)](#)

Status: Issue was fixed post-assessment.

Pull requests:

- <https://github.com/bitwarden/server/pull/4747>

A Content Security Policy was added as a header on all responses from the icons service.

## [WEB.M.02: Cleartext Storage of Sensitive Information in Memory \(Medium\)](#)

Status: Issue was fixed post-assessment.

Pull requests:

- <https://github.com/bitwarden/clients/pull/11781>

A window reload was implemented to prompt browsers to release memory used by the web application, however this is not guaranteed and the experience may vary depending on the browser used.

## WEB.M.03: Audit / Event Log Tampering (Medium)

Status: Issue was fixed post-assessment.

Documentation at <https://bitwarden.com/help/event-logs/> was updated to clarify event processing:

*Events are captured at both the Bitwarden client and server, with most events occurring at the client level. While server event capture is instantaneous and quickly processed, clients push event data to the server every 60 seconds, so you may observe small delays in the reporting of recent events. Furthermore, client events are ingested by said client communicating data via an API call, and this is retried until success. As a result, if the client cannot communicate to the API or is somehow modified to not send events then they will not be received and therefore processed.*

The event logging process itself was not changed.

## WEB.L.01: Unauthorized Access to Premium Features (Low)

Status: Accepted.

Bad actors are able to modify configuration locally to bypass controls around premium feature access and while this is somewhat complicated to perform and maintain, it is accepted by Bitwarden.

## WEB.L.02: Session Token in URL (Low)

Status: Accepted as an upstream limitation.

Bitwarden must provide a SignalR authentication token in query strings for notifications over WebSockets to work as expected. This cannot be avoided and is required by SignalR, as noted in [Microsoft documentation](#). Exposure is minimal and limited to basic information sent in the push payload.

## WEB.L.03: Missing Integrity Check For External Resources (Low)

Status: Issue under planning and research.

A third-party library used on the Bitwarden marketing website for GDPR compliance is being replaced with a provider that can provide SRI.

## WEB.L.04: Weak Password Policy for Vault Exports (Low)

Status: Issue was fixed post-assessment.

Pull requests:

- <https://github.com/bitwarden/clients/pull/10539>

A password strength indicator was added to the export process. No requirements were added to the password itself.

## WEB.BPI.01: Disable Software Name and Version Reporting (Info)

Status: Issue was fixed post-assessment.

CDN and edge network changes were made to omit certain response headers that revealed light details about those systems.

## WEB.BPI.02: Restrict File Uploads (Info)

Status: Accepted.

Bitwarden will not inspect file upload contents as it considers that an overstepping of its role.

## WEB.BPI.03: Remove Private IP Addresses from API Responses (Info)

Status: Issue was fixed post-assessment.

Pull requests:

- <https://github.com/bitwarden/server/pull/4771>

An API that's no longer used was removed.

# FINDINGS – EXTERNAL NETWORK

## ATTACK NARRATIVE

The primary objective of the external penetration test was to identify vulnerabilities or threats that could allow a threat actor to compromise systems, application data, or place the applications and infrastructure at risk. All the external testing was executed from an unauthenticated perspective simulating the actions of a threat actor who did not have access to any client systems or credentials. Bitwarden provided a list of three wild card domains and two subdomains to be scanned for vulnerabilities and misconfigurations from a Fracture Labs cloud-based attack box.

Fracture Labs began the assessment by performing open-source intelligence (OSINT) on the provided assets. This phase aimed to collect as much publicly available information as possible. The assessment team identified subdomains, associated IP addresses, WHOIS information, and other relevant data. This information provided a better understanding of the structure and exposure of Bitwarden’s external assets, resulting in a comprehensive list of 74 targets identified for further assessment. This list was reviewed and approved by Bitwarden before proceeding to the next phase.

Next, Fracture Labs performed a comprehensive port scan against the external hosts to discover open ports and services. Each open port was cataloged and the associated service technologies and version numbers were documented.

Using a mixture of manual and automated testing techniques, Fracture Labs analyzed these services for known vulnerabilities, default configurations, and potential misconfigurations, such as outdated software, improper configurations, and other security weaknesses. Fracture Labs also executed a custom Nessus vulnerability scan and manually verified each potential finding.

Additionally, screenshots of all web services were taken to help identify technology, services, and interesting targets. Fracture Labs performed directory and file brute-forcing to search for hidden directories, sensitive files, and other potentially exploitable resources.

Following this, the assessment team conducted a thorough review of TLS/SSL configurations of the identified hosts and discovered one system that utilized outdated Transport Layer Security (TLS) versions 1.0 and 1.1, which have been deprecated since 2021 (see [EXT.L.01 – TLS 1.0 and TLS 1.1 in Use](#)).

```
└─ $ sslscan appfunctions.bitwarden.net
Version: 2.1.4
OpenSSL 3.2.2 4 Jun 2024
Connected to 20.62.225.137
Testing SSL server appfunctions.bitwarden.net on port 443 using SNI name appfunctions.bitwarden.net
SSL/TLS Protocols:
SSLv2      disabled
SSLv3      disabled
TLSv1.0    enabled
TLSv1.1    enabled
TLSv1.2    enabled
TLSv1.3    enabled
```

The use of deprecated TLS versions poses security risks as they contain known vulnerabilities and weaknesses that may allow a threat actor to conduct machine-in-the-middle attacks, decrypt sensitive information, or compromise the integrity of data in transit. Additionally, regulatory and compliance standards mandate the discontinuation of these outdated TLS versions due to their inherent security risks.

Overall, the systems performed well and demonstrated resilience to potential attacks. The majority of the infrastructure exhibited robust security measures, with only one minor vulnerability identified. This indicates a strong security posture and effective defensive strategies in place, ensuring the protection of critical assets against malicious activities.



## HIGH-RISK FINDINGS

No high-risk findings were identified.

## MEDIUM-RISK FINDINGS

No medium-risk findings were identified.

## LOW-RISK FINDINGS

EXT.L.01 – TLS 1.0 AND TLS 1.1 IN USE			
	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	<i>Difficult to Reproduce</i>	Needs Special Circumstances	Easily Reproduced
Exploitability	<i>Expert, Advanced Skills</i>	Skilled	Novice
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	Limited	<i>Obvious, Published</i>

### RISK SUMMARY

Fracture Labs identified one system utilizing outdated Transport Layer Security (TLS) versions 1.0 and 1.1, which have been deprecated since 2021. The use of deprecated TLS versions poses security risks as they contain known vulnerabilities and weaknesses that may allow a threat actor to conduct machine-in-the-middle attacks, decrypt sensitive information, or compromise the integrity of data in transit. Additionally, regulatory and compliance standards mandate the discontinuation of these outdated TLS versions due to their inherent security risks.

### AFFECTED ASSETS

- <https://appfunctions.bitwarden.net>

## TECHNICAL DETAILS

---

Fracture Labs identified the vulnerability by enumerating the SSL/TLS versions and cipher suites supported by the server, which provided detailed information about the server's cryptographic configurations.

```
└─$ sslscan appfunctions.bitwarden.net
Version: 2.1.4
OpenSSL 3.2.2 4 Jun 2024
Connected to 20.62.225.137
Testing SSL server appfunctions.bitwarden.net on port 443 using SNI name appfunctions.bitwarden.net
SSL/TLS Protocols:
SSLv2      disabled
SSLv3      disabled
TLSv1.0    enabled
TLSv1.1    enabled
TLSv1.2    enabled
TLSv1.3    enabled
```

## REMIEDIATION RECOMMENDATIONS

---

Bitwarden should consider disabling TLS 1.0 and TLS 1.1 in favor of more secure and modern versions of TLS, preferably TLS 1.2 or TLS 1.3. Additionally, Bitwarden should regularly monitor and audit system configurations to ensure adherence to security best practices and timely updates in line with industry standards.

## ADDITIONAL RESOURCES

---

Deprecating TLSv1.0 and TLSv1.1

<https://datatracker.ietf.org/doc/rfc8996/>

Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations

<https://csrc.nist.gov/pubs/sp/800/52/r2/final>

# FINDINGS – WEB APPLICATION ASSESSMENT

## ATTACK NARRATIVE

Bitwarden provided a list of 25 web applications to be assessed for vulnerabilities. The primary goal of the web application testing was to find and exploit vulnerabilities that could lead to a compromise of customer data or Bitwarden infrastructure. The testing was performed against the production application instance from an external Fracture Labs attack box.

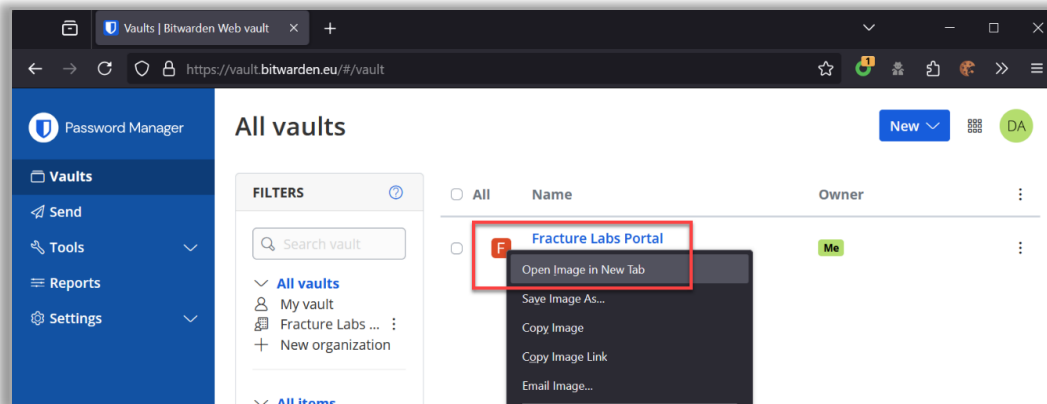
Fracture Labs began with unauthenticated testing of the web applications, looking for unpatched infrastructure, sensitive endpoints, and weaknesses that could allow unprotected access to application data and functions intended for authenticated users. No vulnerabilities were found during this phase.

Next, the assessment team performed directory and file brute-forcing to search for hidden directories, sensitive files, and other potentially exploitable resources. While no significant findings were identified, the team located a swagger file for `api.bitwarden.com` and documentation pertaining to the local vault management command line interface (CLI) API. Although the CLI and local vault management API were not in scope for this assessment, Fracture Labs intercepted API calls made by the CLI and observed many calls were similar to the ones performed by the web vault.

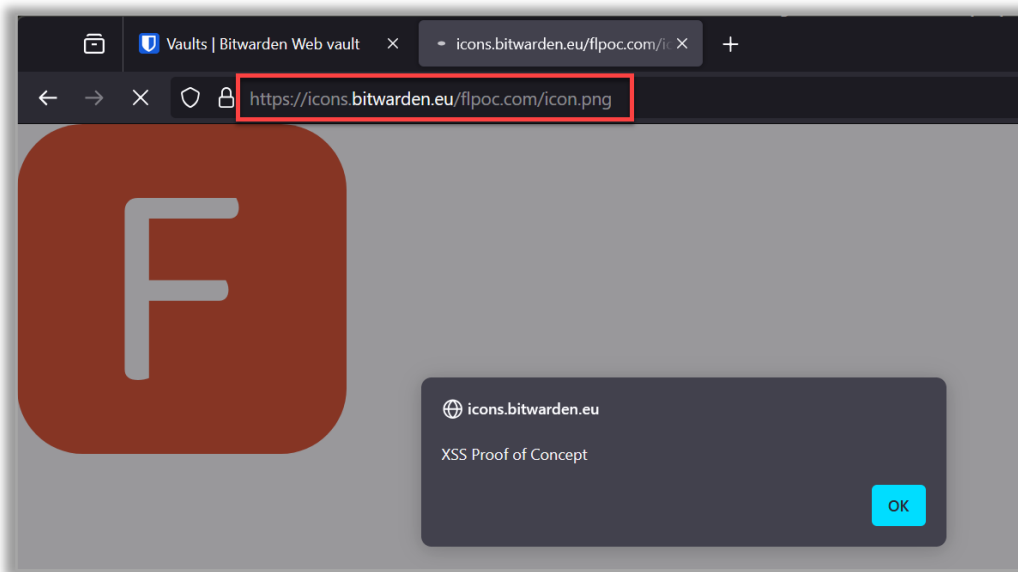
Common injection attacks, including Cross-Site Scripting, OS Injection, and SQL Injection, were thoroughly tested by manipulating POST bodies, GET parameters, headers, and cookie values. While the application handled most of the unexpected input properly, Fracture Labs identified that the web vault icon server could be abused to serve malicious SVG files from a Bitwarden domain to execute a Cross-Site Scripting (XSS) attack against other users (see [WEB.M.01- Cross-Site Scripting Via External Resource](#)).

The vault icon server accepted requests containing the domain of a targeted system as part of the URL and served the `favicon` file for that domain from cache if one was found. When the item was not found in cache, the icon server requested the webpage for that domain and parsed the HTML looking for a `favicon`. The `favicon` was then saved to the icon server cache and served from the icon server itself.

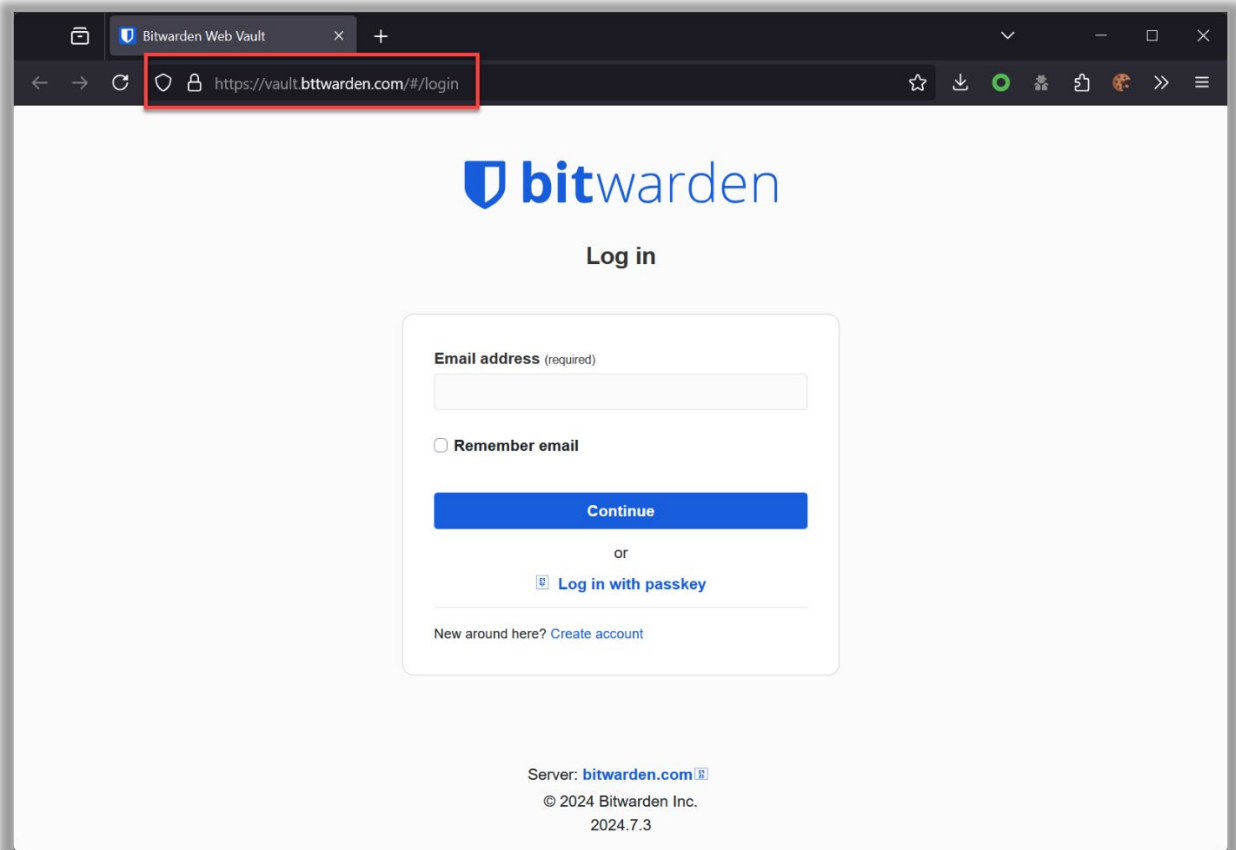
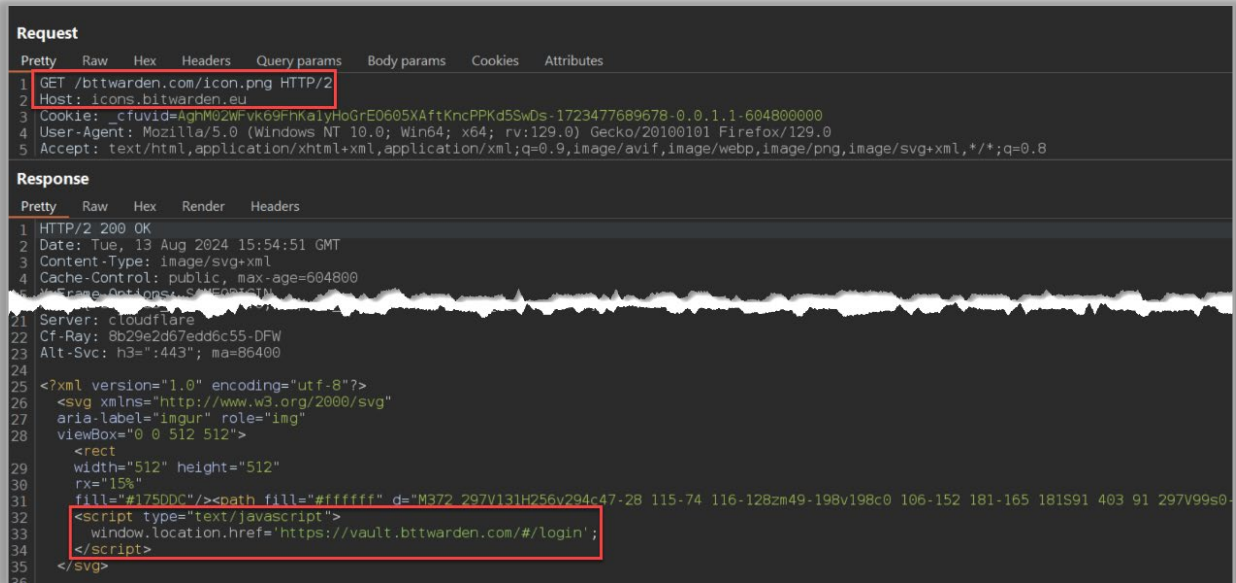
Fracture Labs set the URI field on a vault entry to `https://flpoc.com`, which caused the icon server to retrieve the malicious file and display it as an icon. Although modern browsers prevent JavaScript code execution when the `img` tag is used (as was the case in the web vault), the malicious code would execute if a user opened the image in a new tab or window.



The malicious JavaScript executed when browsing to the icon file, which was served by a Bitwarden domain.



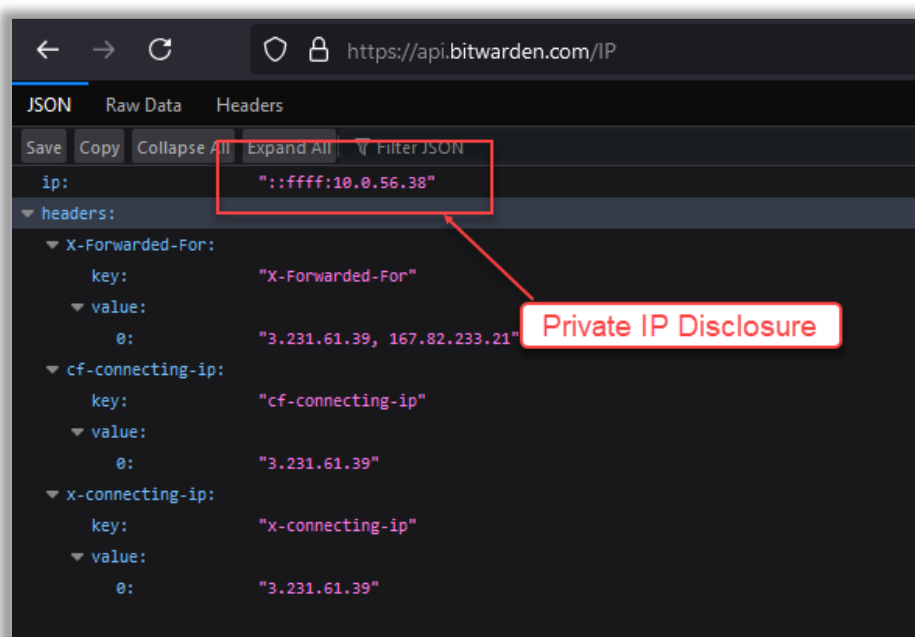
Furthermore, since the icon server did not require authentication, the link to the malicious icon file could be sent as part of a phishing campaign to abuse trust in the Bitwarden domain. Fracture Labs created a proof-of-concept attack in which a victim would be redirected from a valid Bitwarden domain (`hxxps://icons[.]bitwarden[.]eu/bttwarden[.]com/icon.png`) to a spoofed vault login page (`hxxps://vault[.]bttwarden[.]com/#/login`). A threat actor could abuse this to harvest credentials from a victim and decrypt vault secrets if the account did not have MFA enabled.



Fracture Labs attempted to abuse the icon server fetching feature to redirect the requestor to other URLs, including those local to the icon server. All attempts at such server-side request forgery (SSRF) attacks failed due to robust validation in the icon server code.

Next, Fracture Labs checked for potential information disclosure vulnerabilities by inspecting server responses for internal IP addresses, software version details, or other sensitive information that could aid a threat actor furthering attacks against Bitwarden assets.

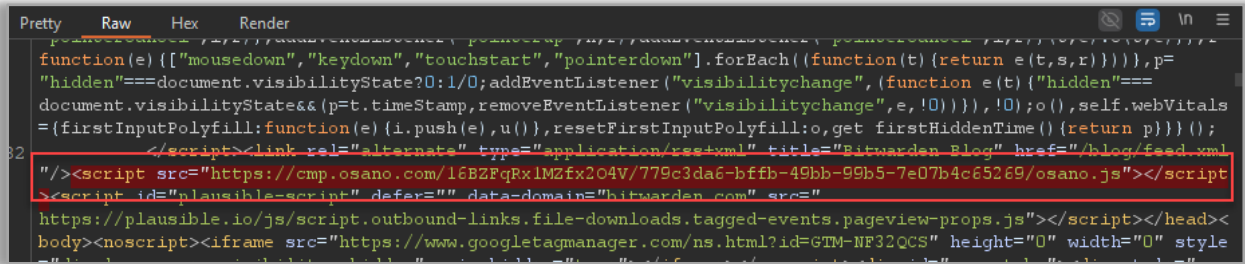
Fracture Labs discovered one API response that disclosed the private IP address of the server (see [WEB.BPI.03 – Remove Private IP Addresses from API Responses](#)). A threat actor could leverage this vulnerability to gain insights into the internal network architecture, which could help facilitate other targeted attacks against the infrastructure.



Also, Fracture Labs observed that the Bitwarden marketing site loaded an external script without performing subresource integrity (SRI) checks (see [WEB.L.03 – Missing Integrity Check for External Resources](#)). SRI is a security feature that allows browsers to verify that resources they fetch (e.g., from a CDN) are delivered without unexpected manipulation by providing a cryptographic hash that the browser can use to check the integrity of the fetched file.

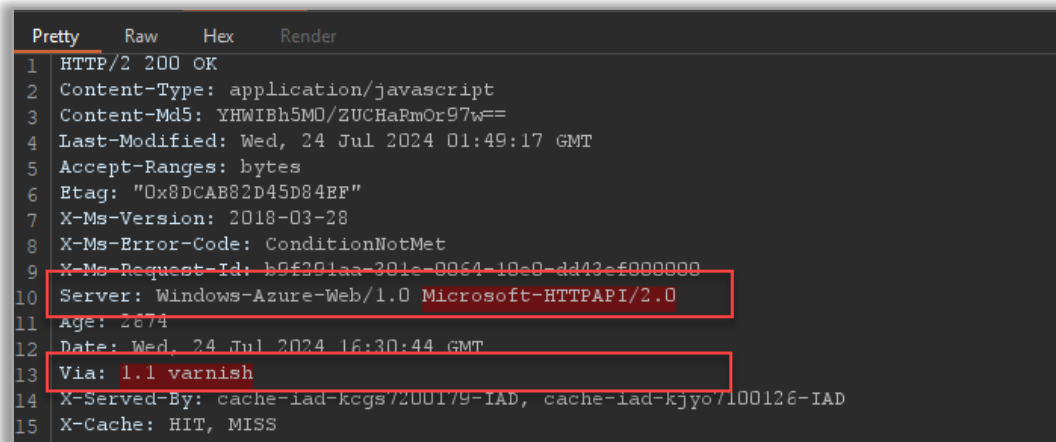
A threat actor with access to the resource files via a supply chain attack could inject malicious content into these resources, compromising the application's security. For example, in 2024, the Polyfill CDN used by hundreds of thousands of websites was hijacked and victim browsers were redirected to malicious and inappropriate websites. This highlights the importance of restricting usage to known-good versions of third-party assets.

This issue was discovered by examining the requests and responses between the application and the servers hosting these external resources. Specifically, the HTML source code of several pages included references to an external resource but lacked the integrity attribute in the `<script>` tag.



```
function(e) ([ "mousedown", "keydown", "touchstart", "pointerdown" ].forEach( (function(t) {return e(t,s,r)})) ), p=
"hidden"===document.visibilityState?0:1/0;addEventListener("visibilitychange", (function e (t) {"hidden"===
document.visibilityState&&(p=t.timestamp,removeEventListener("visibilitychange",e,!0)}),!0);o(),self.webVitals
=(firstInputPolyfill:function(e) {i.push(e),u()}),resetFirstInputPolyfill;o,get firstHiddenTime() {return p}});
32 </script><link rel="alternate" type="application/rss+xml" title="Bitwarden Blog" href="/blog/feed.xml
"/><script src="https://cmp.osano.com/16BZFqRxlM2fx204V/779c3da6-bffb-49bb-99b5-7e07b4c65269/osano.js"></script
<script id="plausible-script" defer="" data-domain="bitwarden.com" src="
https://plausible.io/js/script.outbound-links.file-downloads.tagged-events.pageview-props.js"></script></head><
body><noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-NF32QCS" height="0" width="0" style
```

The assessment team also identified several instances where the application name and version were disclosed (see [WEB.BPI.01 – Disable Software Name and Version Reporting](#)). Revealing this information in headers can provide valuable information to threat actors, allowing them to target specific vulnerabilities associated with that version.



```
1 HTTP/2 200 OK
2 Content-Type: application/javascript
3 Content-Md5: YHWIBh5MO/ZUCHaRmOr97w==
4 Last-Modified: Wed, 24 Jul 2024 01:49:17 GMT
5 Accept-Ranges: bytes
6 Etag: "0x8DCAB82D45D84BF"
7 X-Ms-Version: 2018-03-28
8 X-Ms-Error-Code: ConditionNotMet
9 X-Request-Id: b0f291aa-301e-0064-10e0-dd43e-f000000
10 Server: Windows-Azure-Web/1.0 Microsoft-HTTPAPI/2.0
11 Age: 1674
12 Date: Wed, 24 Jul 2024 16:30:44 GMT
13 Via: 1.1 varnish
14 X-Served-By: cache-iad-kogs/200179-IAD, cache-iad-kjyo/100126-IAD
15 X-Cache: HIT, MISS
```

Fracture Labs then focused on the authenticated testing of vault applications, testing the US and EU instances independently to ensure consistency across the regions.

Using self-registered test accounts, Fracture Labs mapped the applications by browsing them while capturing all traffic in a local intercepting proxy. A list of valid endpoints was created by fuzzing potential routes and filenames, but none provided access to anything not already known from browsing the application.

Access controls were thoroughly tested to ensure proper implementation of authorization mechanisms. The objective was to identify any Insecure Direct Object Reference vulnerabilities (IDORs) that might allow unauthorized access to information and vaults of other users or organizations. Despite various techniques to bypass authorization checks, no vulnerabilities were discovered, indicating effective prevention of unauthorized access to organizational data and vaults.

Next, Fracture Labs focused on the event log report found within the web vault. The report listed each action a user took on the vault for auditing purposes, but all client-side actions (such as viewing or modifying a vault item) were initiated from the user’s browser. This made the event logging susceptible to tampering by allowing a threat actor to intercept, modify, or drop log entries altogether (see [WEB.M.03 – Audit/Event Log Tampering](#)). This could lead to inaccurate or misleading audit trails, hindering effective incident response and forensic analysis.

To demonstrate this, Fracture Labs intercepted all traffic using an intercepting proxy and dropped the requests for log events, preventing the application from logging sensitive actions, such as viewing or modifying a password.

```

Request
-----
1 POST /events/collect HTTP/2
2 Host: vault.bitwarden.com
3 Content-Length: 131
4 Device-Type: 9
5 Accept-Language: en-US
6 Bitwarden-Client-Version: 2024.7.1
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer ...snipped...
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
10 Content-Type: application/json; charset=utf-8
11 Bitwarden-Client-Name: veb
12 Sec-Ch-Ua-Platform: "Windows"
13 Accept: */*
14 Origin: https://vault.bitwarden.com
15 Referer: https://vault.bitwarden.com/
16 Accept-Encoding: gzip, deflate, br
17 Priority: u=1, i
18
19 [{"type":1107,"cipherId":"6477e17f-bd2d-490d-a84b-b1b900390d99","date":"2025-07-30T02:18:41.035Z","organisationId":"au11"}]
    
```

Fracture Labs also discovered that deleting the JSON value and leaving it empty (in an invalid JSON format) from the user\*\_eventCollection\_events key in session storage broke the logging function as well.

Key	Value
user_667e6a6d-ccc2-4e32-a99d-b1b800db9540_eventCollection_events	
user_667e6a6d-ccc2-4e32-a99d-b1b800db9540_token_accessToken	"eyJhbGciOiJIUzI1NiIsImtpZCI6IjZlMzZDMzZDZD"



```
Unhandled error in angular
Error: Uncaught (in promise): SyntaxError: Unexpected end of JSON input
SyntaxError: Unexpected end of JSON input (at zone.umd.js:1250:35)
    at JSON.parse (<anonymous>)
    at Lf.get (window-storage.service.ts:25:35)
    at util.ts:11:37
    at Generator.next (<anonymous>)
    at main.b9c0aeb...js:3:526866
    at t (zone.umd.js:1339:25)
    at Bd (main.b9c0aeb...js:3:526611)
    at Xd (main.b9c0aeb...js:3:526904)
    at qd.<anonymous> (state-base.ts:107:18)
    at Generator.next (<anonymous>)
    at P (zone.umd.js:1250:35)
    at zone.umd.js:1157:21
    at zone.umd.js:1173:37
    at a (main.b9c0aeb...js:3:551007)
    at t.invoke (zone.umd.js:411:30)
    at Object.onInvoke (ng_zone.ts:443:29)
    at t.invoke (zone.umd.js:410:56)
    at r.run (zone.umd.js:165:47)
    at zone.umd.js:1314:38
    at t.invokeTask (zone.umd.js:445:35)
```

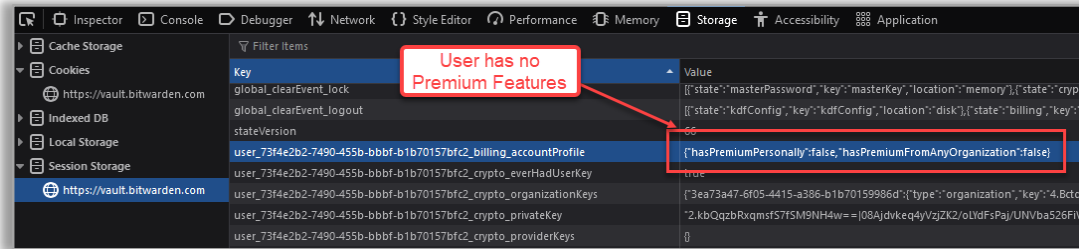
Also, Fracture Labs forged audit log entries by manually sending multiple modified requests.

Timestamp	Source	User	Action
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed security code for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Copied hidden field for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Copied password for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Auto-filled item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed password for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed Card Number for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Copied security code for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed password for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed hidden field for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed password for item 34e87119.
Aug 1, 2024, 10:57:57 AM	Web vault - Firefox	Delta Admin	Viewed password for item 34e87119.

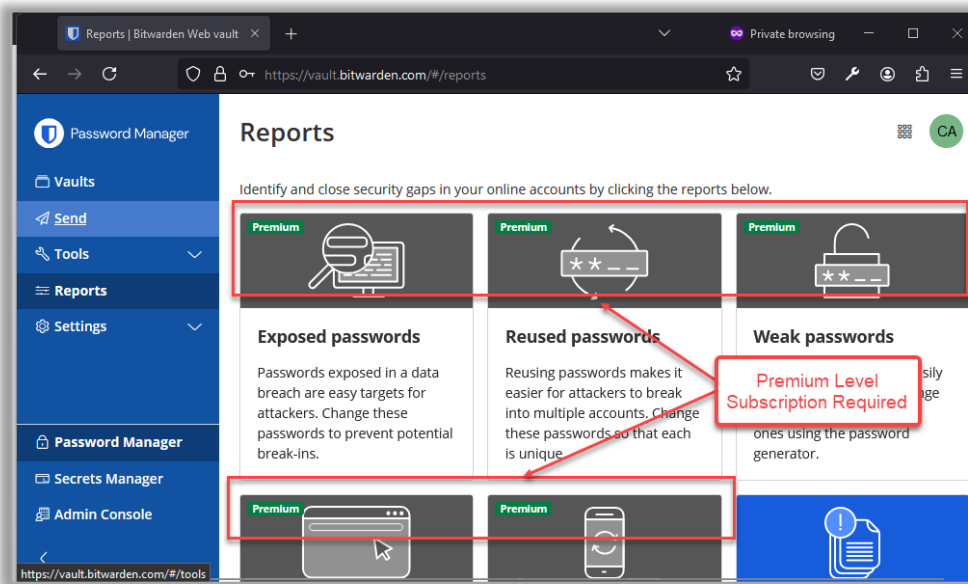
Next, Fracture Labs examined the browser’s local and session storage containers and found that the application stored user information and attributes in both. The data included information such as keys, profile, settings, and tokens. The profile attribute included two properties (`hasPremiumFromAnyOrganization` and `hasPremiumPersonally`) that the application used to determine a user's access to premium content without conducting backend validation (see [WEB.L.01 – Unauthorized Access to Premium Features](#)).

Fracture Labs executed a proof-of-concept attack that demonstrated it was possible for a user with a free account to obtain access to premium features by manipulating local storage values.

For example, Fracture Labs logged into the web vault with a free user account that was not associated with an organization that had a premium subscription. The team then navigated to the **Reports** section, opened the developer tools of the web browser, and inspected the browser's Session Storage. The settings confirmed the user did not have a premium account and was not associated with an organization that had a premium account.



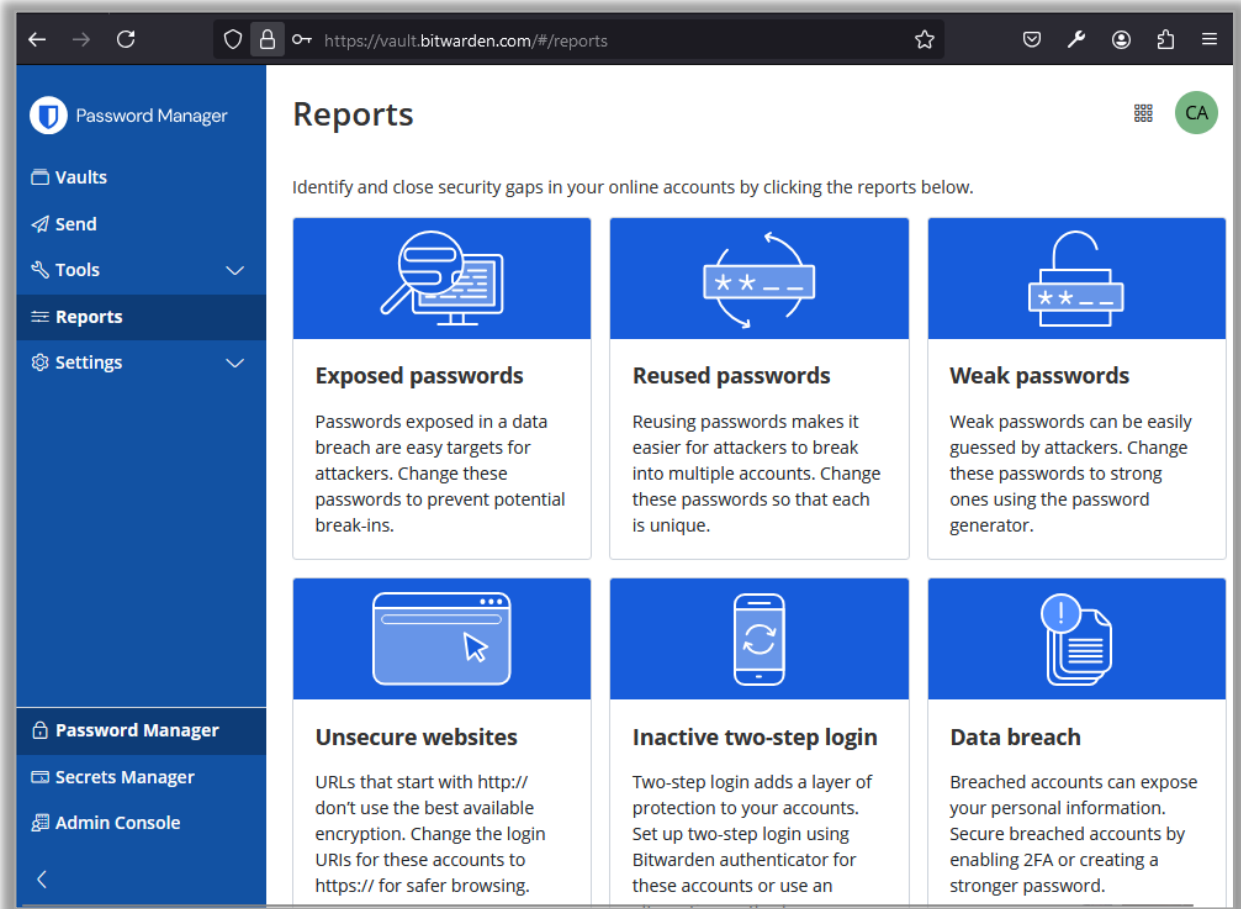
Also, premium features, such as *Exposed Passwords* and *Weak Passwords*, were not accessible without a premium-level subscription.

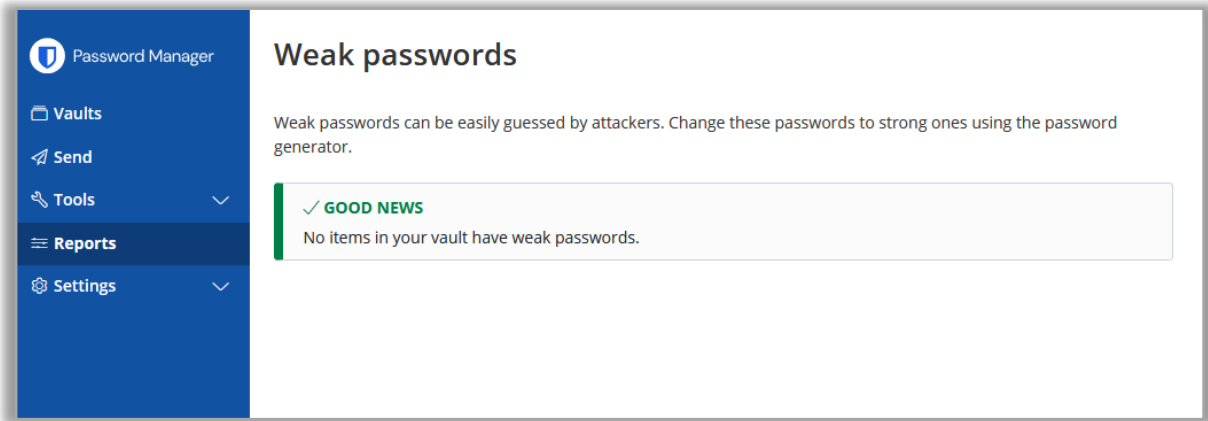


By setting the `hasPremiumPersonally` and `hasPremiumFromAnyOrganization` values to `true`, Fracture Labs was able to access premium features.

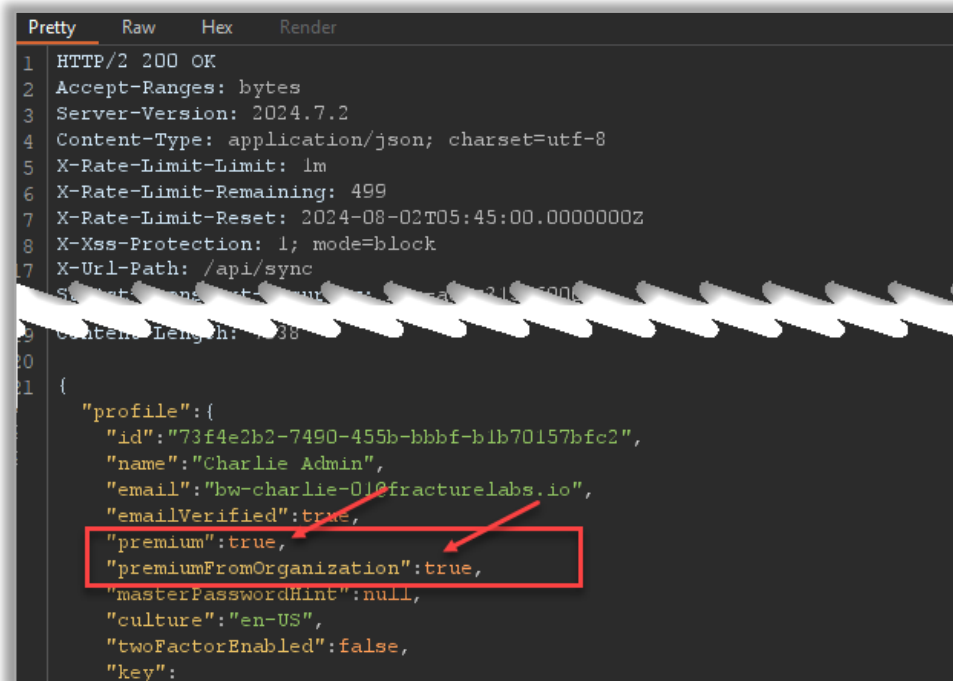
```

73f4e2b2-7490-455b-bbbf-b1b70157bfc2 [{"keys":{"cryptoSymmetricKey":{}}, "profile":{"userId":"73f4e2b2-7490-455b-bbbf-b1b70157bfc2","name":"Charlie Admin","email":"bw-charlie@fracturelabs.com","organizationId":"73f4e2b2-7490-455b-bbbf-b1b70157bfc2","state":{"state":"masterPassword","key":"masterKey","location":"memory"},{"state":"crypto","key":"cryptoKey","location":"memory"},{"state":"kdfConfig","key":"kdfConfig","location":"disk"},{"state":"billing","key":"accountBillingInfo","location":"disk"},{"state":"organizationKeys","key":"organizationKeys","location":"disk"}]}]
global_account_accounts [{"73f4e2b2-7490-455b-bbbf-b1b70157bfc2":{"name":"Charlie Admin","email":"bw-charlie@fracturelabs.com","organizationId":"73f4e2b2-7490-455b-bbbf-b1b70157bfc2","state":{"state":"masterPassword","key":"masterKey","location":"memory"},{"state":"crypto","key":"cryptoKey","location":"memory"},{"state":"kdfConfig","key":"kdfConfig","location":"disk"},{"state":"billing","key":"accountBillingInfo","location":"disk"},{"state":"organizationKeys","key":"organizationKeys","location":"disk"}]}]}
global_account_activeAccountId "73f4e2b2-7490-455b-bbbf-b1b70157bfc2"
global_account_activity [{"73f4e2b2-7490-455b-bbbf-b1b70157bfc2":{"state":{"state":"masterPassword","key":"masterKey","location":"memory"},{"state":"crypto","key":"cryptoKey","location":"memory"},{"state":"kdfConfig","key":"kdfConfig","location":"disk"},{"state":"billing","key":"accountBillingInfo","location":"disk"},{"state":"organizationKeys","key":"organizationKeys","location":"disk"}]}]}
global_clearEvent_lock [{"state":"masterPassword","key":"masterKey","location":"memory"},{"state":"crypto","key":"cryptoKey","location":"memory"},{"state":"kdfConfig","key":"kdfConfig","location":"disk"},{"state":"billing","key":"accountBillingInfo","location":"disk"},{"state":"organizationKeys","key":"organizationKeys","location":"disk"}]
global_clearEvent_logout [{"state":"kdfConfig","key":"kdfConfig","location":"disk"},{"state":"billing","key":"accountBillingInfo","location":"disk"},{"state":"organizationKeys","key":"organizationKeys","location":"disk"}]
stateVersion 00
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_billing_accountProfile {"hasPremiumPersonally":true,"hasPremiumFromAnyOrganization":true}
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_crypto_everHadUserKey true
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_crypto_organizationKeys [{"3ea73a47-6f05-4415-a386-b1b70159986d":{"type":"organization","key":"4.BctdpOdw7VjZK2/olYdFsPaj/UNVba526FIWI+KAdT2.kbQqzbRxqmsf57f5M9NH4w==|08Ajdvkeq4yVzjZK2/olYdFsPaj/UNVba526FIWI+KAdT2.kbQqzbRxqmsf57f5M9NH4w=="}]}]
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_crypto_privateKey [{"3ea73a47-6f05-4415-a386-b1b70159986d":{"type":"organization","key":"4.BctdpOdw7VjZK2/olYdFsPaj/UNVba526FIWI+KAdT2.kbQqzbRxqmsf57f5M9NH4w=="}]}]
    
```





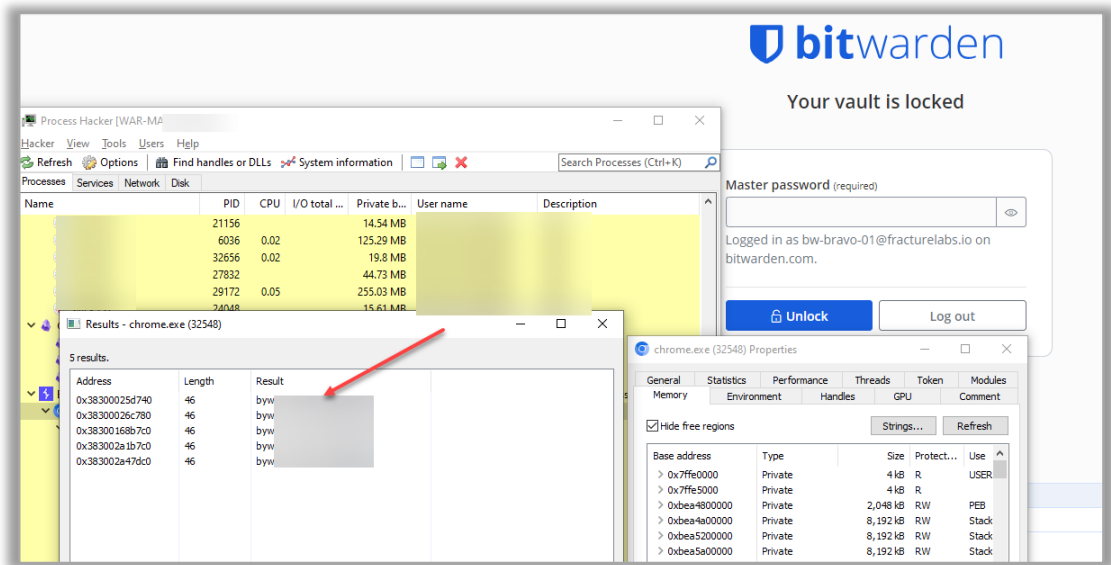
Fracture Labs also unlocked access to premium features by manipulating responses from the API.



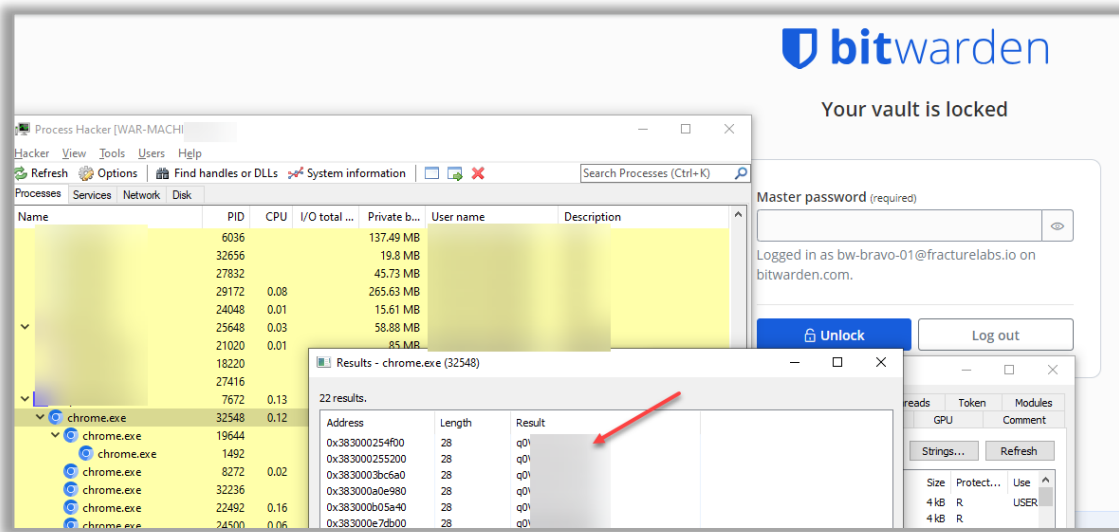
NOTE: Cure53 identified this vulnerability in a 2023 web application security test and Bitwarden accepted the risk as a low business risk.

Next, Fracture Labs focused on examining memory artifacts related to vault sessions and activity. The team discovered that browsers retained the master password and vault passwords in memory, even after the web vault was locked manually or via timeout (see [WEB.M.01 – Cleartext Storage of Sensitive Information in Memory](#)). A threat actor in a position to access the browser process memory could potentially extract the master password and gain full access to user’s Bitwarden vault.

Fracture Labs logged into the web vault and then locked the vault from the menu by clicking the “Lock now” option. Next, Fracture Labs accessed the memory contents of the browser process and searched for the master password strings in the memory contents.

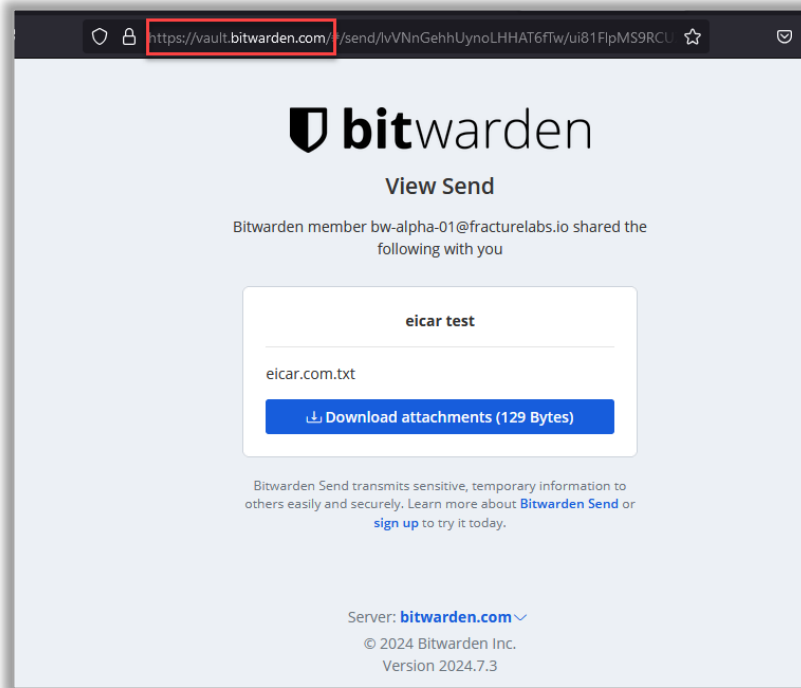


Additionally, Fracture Labs also found instances of passwords that were previously viewed.





Fracture Labs also discovered that the web vault allowed users to upload and share files without any restrictions (see [WEB.BPI.02 – Restrict File Uploads](#)). Although this functionality did not lead to a direct system compromise, a threat actor could still abuse it to upload and share malicious files, such as scripts or malware, with other users. While threat actors could use a myriad of file sharing services to distribute malware, abuse of this functionality could take advantage of victim’s trust in the [vault.bitwarden.com](#) or [vault.bitwarden.eu](#) domain names. To demonstrate this vulnerability, Fracture Labs successfully uploaded and shared [eicar.com.txt](#), a file deliberately identified as malicious by many antivirus vendors.



Next, Fracture Labs discovered that the encrypted Vault Export function did not require users to use a strong password for encrypted JSON files. While a password strength meter was displayed during this process, the application still allowed weak passwords, including single character passwords.

**Export vault**

**ⓘ EXPORTING INDIVIDUAL VAULT**  
Only the individual vault items associated with bw-alpha-01@fracturelabs.io will be exported. Organization vault items will not be included. Only vault item information will be exported and will not include associated attachments.

**Export from** (required)  
My vault

**File format** (required)  
.json (Encrypted)

**Export type**

**Account restricted**  
Use your account encryption key, derived from your account's username and Master Password, to encrypt the export and restrict import to only the current Bitwarden account.

**Password protected**  
Set a file password to encrypt the export and import it to any Bitwarden account using the password for decryption.

**File password** (required)  
• [Eye icon]

This password will be used to export and import this file

Weak

**Confirm file password** (required)  
• [Eye icon]

**Confirm format**

Fracture Labs continued to look for other ways to manipulate the application into performing unintended actions. Specific focus was put into the emergency access request feature, although no weaknesses were identified. Similarly, Fracture Labs attempted to perform password timing attacks against the web vault master password but was unable to gain any insight into timing patterns between valid and invalid passwords. Also, Fracture Labs compared the API calls observed during testing to the publicly available source code to ensure complete coverage. User enumeration attacks were also unsuccessful, as the application gracefully handled login attempts for invalid accounts. Finally, Fracture Labs attempted to exploit the file import functionality but did not identify any weaknesses.



## HIGH-RISK FINDINGS

No high-risk findings were identified.

## MEDIUM-RISK FINDINGS

WEB.M.01 – CROSS-SITE SCRIPTING VIA EXTERNAL RESOURCE			
	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	Difficult to Reproduce	<i>Needs Special Circumstances</i>	Easily Reproduced
Exploitability	Expert, Advanced Skills	<i>Skilled</i>	Novice
Affected Users	Few Users	<i>Some Users</i>	All Users
Discoverability	<i>Obscure</i>	Limited	Obvious, Published

### RISK SUMMARY

Fracture Labs identified that the web vault’s icon server could be abused to execute a Cross-Site Scripting (XSS) attack against other users by loading a malicious image file from an external resource via a trusted Bitwarden domain.

Cross-Site Scripting (XSS) is a type of injection-based web application vulnerability that occurs when threat actor-controlled input is dynamically added to a web page and subsequently treated as code instead of text. Typically, this allows threat actors to achieve JavaScript execution within the browser of a victim user. Successful exploitation of XSS vulnerabilities can result in malicious script execution, account hijacking, information theft, and execution of actions on a user's behalf.

The vault icon server accepted requests containing the domain of a targeted system as part of the URL and served the `favicon` file for that domain from cache if one was found. When the item was not found in cache, the icon server requested the webpage for that domain and parsed the HTML looking for a `favicon`. The `favicon` was then saved to the icon server cache and served from the icon server itself.

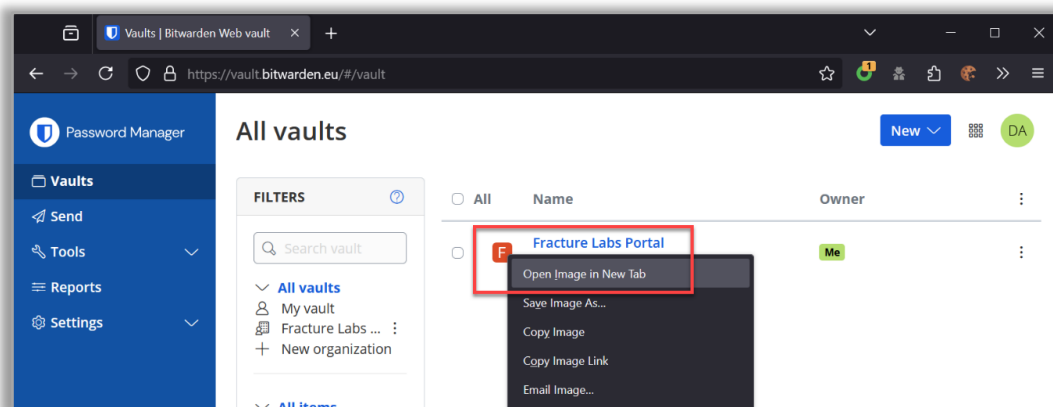
Fracture Labs created a proof-of-concept attack with a `favicon` that contained malicious JavaScript embedded within an SVG file. The malicious icon was then cached by the icon server and accessible via a Bitwarden URL. The JavaScript redirected the victim to a spoofed vault login page capable of capturing the user’s credentials when entered. Threat actors may be able to abuse this functionality to leverage the Bitwarden domain to execute malicious code or harvest user credentials.

## AFFECTED ASSETS

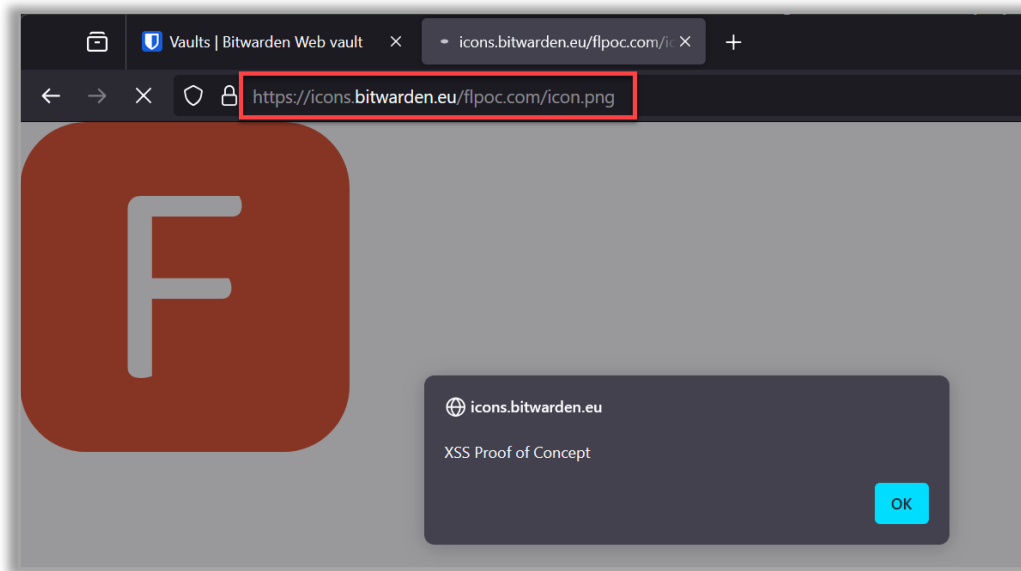
- [https://icons.bitwarden.\[com|eu\]/<domain>/icon.png](https://icons.bitwarden.[com|eu]/<domain>/icon.png)

## TECHNICAL DETAILS

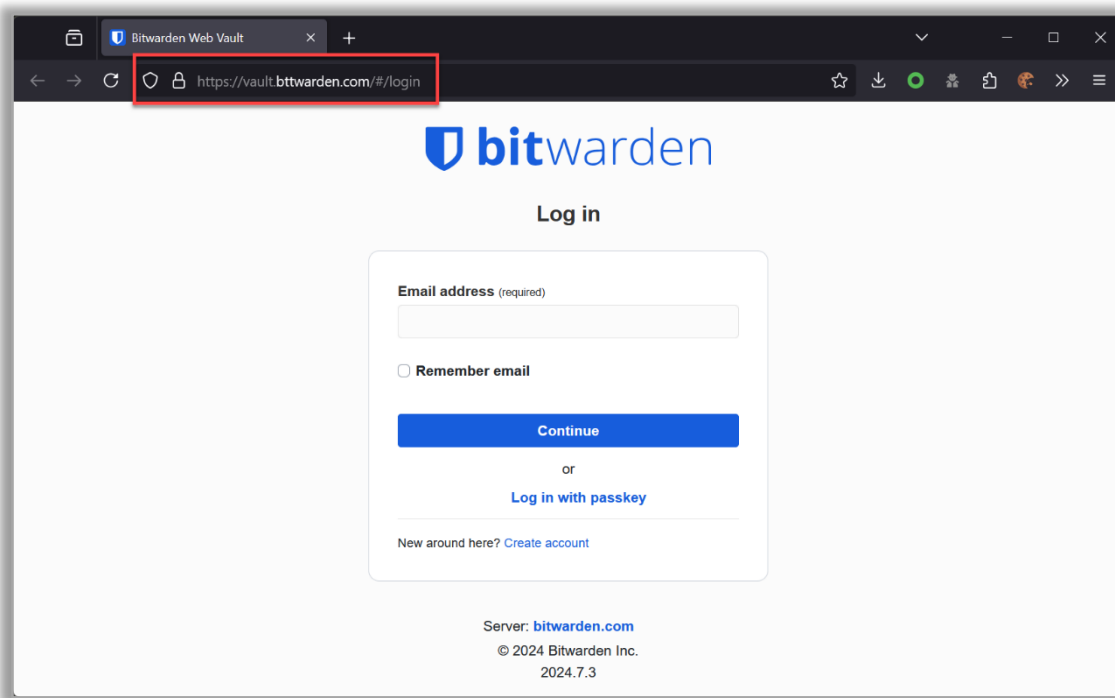
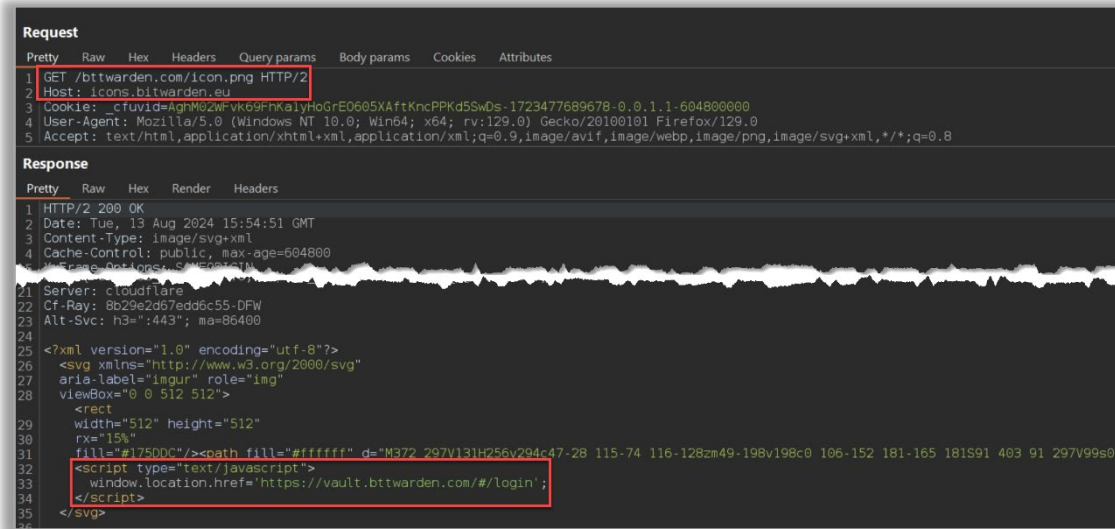
Fracture Labs set the URI field on a vault entry to <https://flpoc.com>, which caused the icon server to retrieve the malicious file and display it as an icon. Although modern browsers prevent JavaScript code execution when the `img` tag is used (as was the case in the web vault), the malicious code would execute if a user opened the image in a new tab or window.



The malicious JavaScript executed when browsing to the icon file, which was served by a Bitwarden domain.



Furthermore, since the icon server did not require authentication, the link to the malicious icon file could be sent as part of a phishing campaign to abuse trust in the Bitwarden domain. Fracture Labs created a proof-of-concept attack in which a victim would be redirected from a valid Bitwarden domain (`hxxps://icons[.]bitwarden[.]eu/bttwarden[.]com/icon.png`) to a spoofed vault login page (`hxxps://vault[.]bttwarden[.]com/#/login`). A threat actor could abuse this to harvest credentials from a victim and decrypt vault secrets if the account did not have MFA enabled.



## REMEDIATION RECOMMENDATIONS

---

Bitwarden should consider implementing a strong Content Security Policy (CSP) on the icon server that prohibits the execution of scripts. For example, Bitwarden could set the following header:

```
Content-Security-Policy: script-src none
```

For further protection, Bitwarden could consider rejecting SVG files that contain JavaScript. Since blacklisting can be difficult to implement and may be circumvented, this should be considered as a complementary control in addition to a strong CSP.

Finally, Bitwarden should consider scanning all SVG files currently cached by the icon server for embedded `<script>` tags. Bitwarden should remove all malicious SVGs, including the ones used for this security assessment (`*.flpoc.com` and `*.bitwarden.com`).

## ADDITIONAL RESOURCES

---

OWASP: A3 - Injection

[https://owasp.org/Top10/A03\\_2021-Injection](https://owasp.org/Top10/A03_2021-Injection)

OWASP: Content Security Policy Cheat Sheet

[https://cheatsheetseries.owasp.org/cheatsheets/Content\\_Security\\_Policy\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html)

WEB.M.02 – CLEARTEXT STORAGE OF SENSITIVE INFORMATION IN MEMORY			
	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	<i>Easily Reproduced</i>
Exploitability	Expert, Advanced Skills	<i>Skilled</i>	Novice
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

## RISK SUMMARY

Fracture Labs discovered that browsers retained the master password and vault passwords in memory, even after the web vault was locked manually or via timeout. Furthermore, passwords were discovered in memory, even after completely logging out of the vault. Bitwarden’s documentation appears to conflict with this observation, as it states master passwords and unencrypted data are not stored in memory while the vault is locked or logged out.

A threat actor with access to the browser process memory could extract the master password and gain full access to the user’s Bitwarden vault.

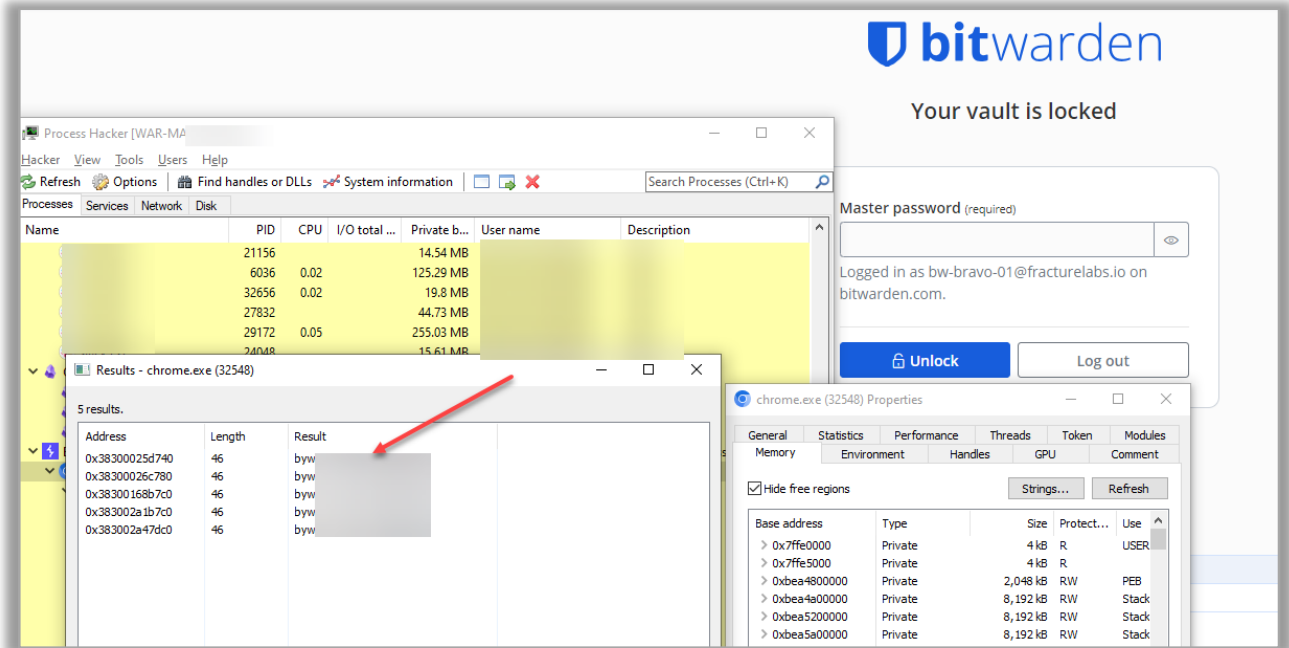
This issue was reproduced across common web browsers, including Brave, Chrome, Edge, and Firefox. However, no remnants of the master password were found in memory after the browser tab was closed.

## AFFECTED ASSETS

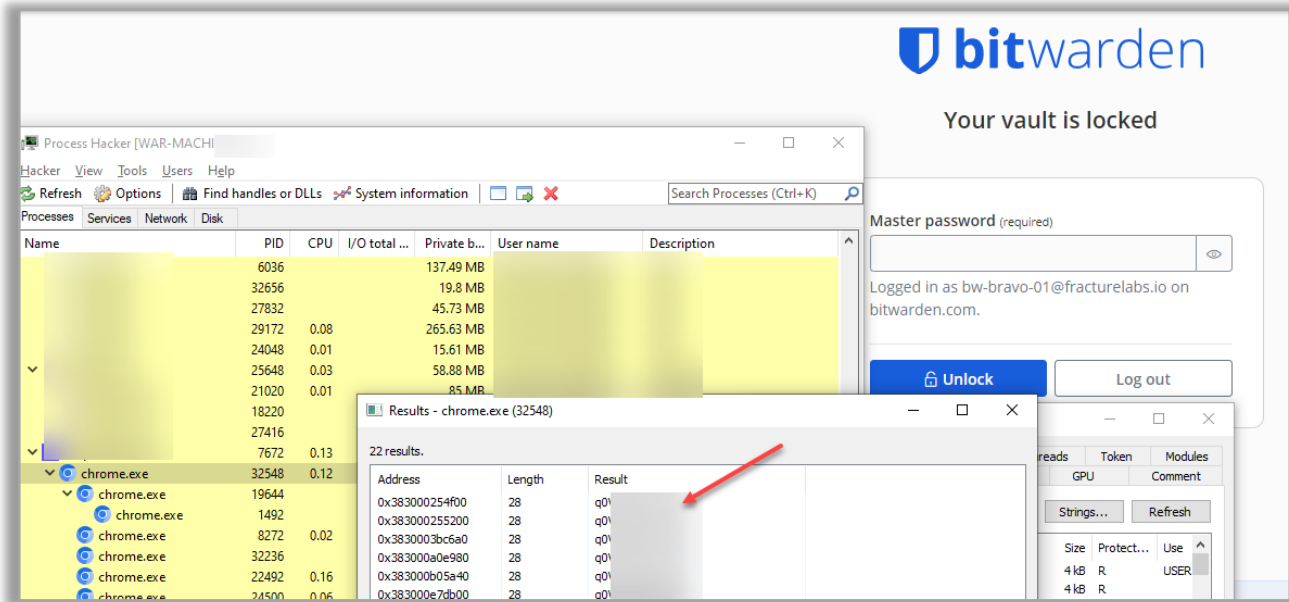
- [https://vault.bitwarden.\[com|eu\]](https://vault.bitwarden.[com|eu])

## TECHNICAL DETAILS

Fracture Labs logged into the web vault and then locked the vault from the menu by clicking the “Lock now” option. Next, Fracture Labs accessed the memory contents of the browser process and searched for the master password strings of the memory contents.



Additionally, Fracture Labs also found instances of passwords that were previously viewed.



Fracture Labs also dumped the memory of the browser process to disk and discovered plaintext master credentials.

```
$ strings firefox\ \2\).DMP | grep -i hungary
Hungary/M
region-name-hu = Hungary
  name: "HUNGARY (Rep.)",
hungary6
hungary6
hungary6
:bw-delta-01@fracturelabs.io:hungary6
hungary6
hungary6
hungary6
```

## REMEDIATION RECOMMENDATIONS

---

Bitwarden should consider implementing a process reload or refresh mechanism to clear sensitive values from memory when the vault is locked or logged out.

## ADDITIONAL RESOURCES

---

CWE-316: Cleartext Storage of Sensitive Information in Memory

<https://cwe.mitre.org/data/definitions/316.html>

Bitwarden: Is My Bitwarden Master Password Stored Locally

<https://bitwarden.com/help/security-faqs/#q-is-my-bitwarden-master-password-stored-locally>

<b>WEB.M.03 – AUDIT/EVENT LOG TAMPERING</b>			
	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	<i>Easily Reproduced</i>
Exploitability	Expert, Advanced Skills	<i>Skilled</i>	Novice
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

### RISK SUMMARY

Fracture Labs discovered that it was possible to subvert event logging by intercepting the log traffic and dropping or modifying the requests sent to the server.

The event log report listed each action a user took on the vault for auditing purposes, but all client-side actions (such as viewing or modifying a vault item) were initiated from the user’s browser. This made the event logging susceptible to tampering by allowing a threat actor to intercept, modify, or drop log entries altogether. This could lead to inaccurate or misleading audit trails, hindering effective incident response and forensic analysis.

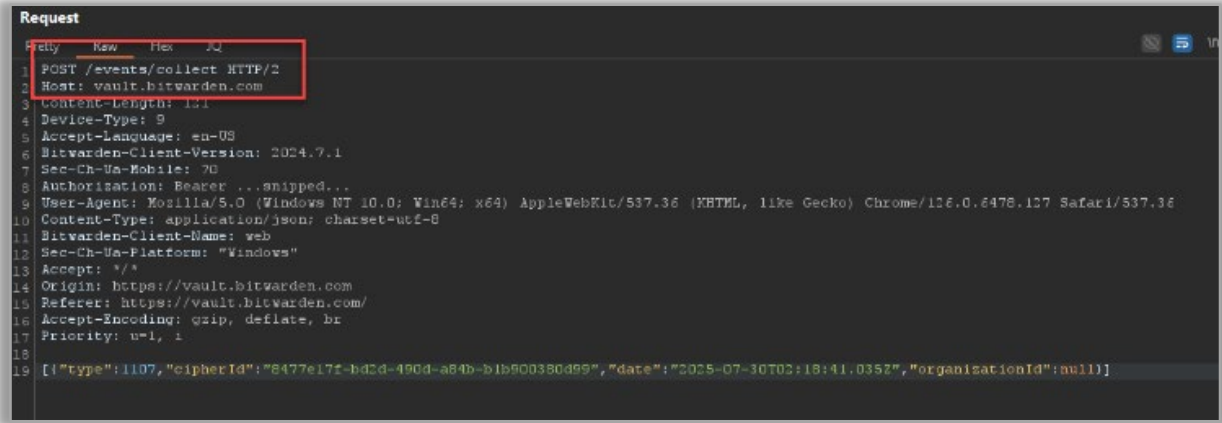
### AFFECTED ASSETS

- [https://vault.bitwarden.\[com|eu\]/events/collect](https://vault.bitwarden.[com|eu]/events/collect)

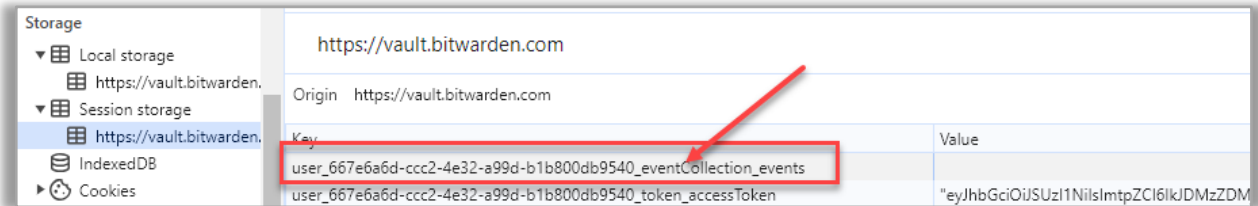
### TECHNICAL DETAILS

Fracture Labs intercepted all traffic using an intercepting proxy and dropped the requests for log events, preventing the application from logging sensitive actions such as viewing or modifying a password.

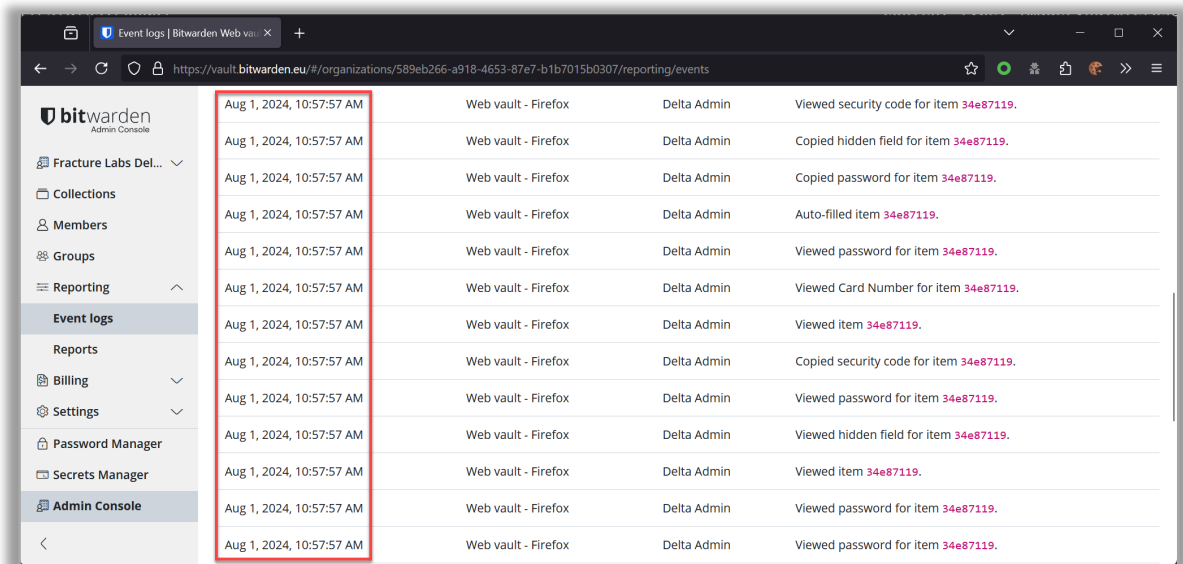




Fracture Labs also discovered that deleting the JSON value and leaving it empty (in an invalid JSON format) from the user\*\_eventCollection\_events key in session storage broke the logging function as well.



Also, Fracture Labs forged audit log entries by manually sending multiple modified requests.



## REMIEDIATION RECOMMENDATIONS

Bitwarden should consider implementing the log events to be handled by the back end rather than from the client which could prevent tampering with the log events. Due to the architecture of the web vault, this may not be feasible. In which case, Bitwarden may want to advise users that the event report relies upon user-reported data and may not be suitable for legal forensics or auditing purposes.

## ADDITIONAL RESOURCES

MITRE ATT&CK – T1562: Impair Defenses

<https://attack.mitre.org/techniques/T1562/>

CWE-223: Omission of Security-relevant Information

<https://cwe.mitre.org/data/definitions/223.html>

## LOW-RISK FINDINGS

WEB.L.01 – UNAUTHORIZED ACCESS TO PREMIUM FEATURES			
	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	<i>Easily Reproduced</i>
Exploitability	Expert, Advanced Skills	<i>Skilled</i>	Novice
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

### RISK SUMMARY

Fracture Labs examined the browser’s local and session storage containers and found that the application stored user information and attributes in both. The data included information such as keys, profile, settings, and tokens. The profile attribute included two properties (*hasPremiumFromAnyOrganization* and *hasPremiumPersonally*) that the application used to determine a user's access to premium content without conducting backend validation.

Fracture Labs executed a proof-of-concept attack that demonstrated it was possible for a user with a free account to obtain access to premium features by manipulating local storage values. While threat actors could access premium features without a subscription, this did not allow unauthorized access to sensitive data.

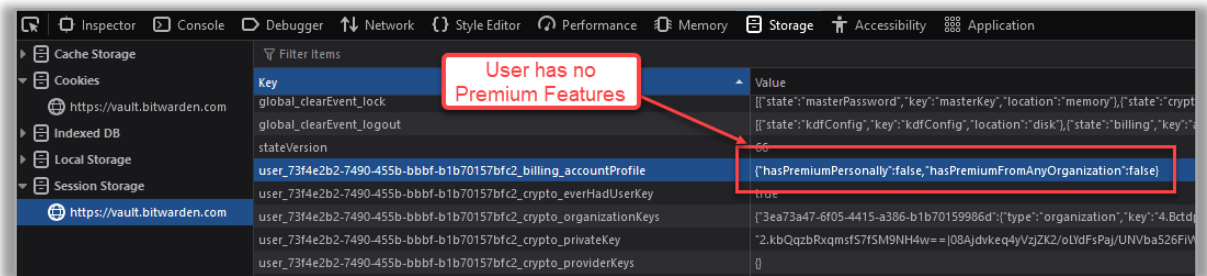
NOTE: Cure53 identified this vulnerability in a 2023 web application security test and Bitwarden accepted the risk as a low business risk.

### AFFECTED ASSETS

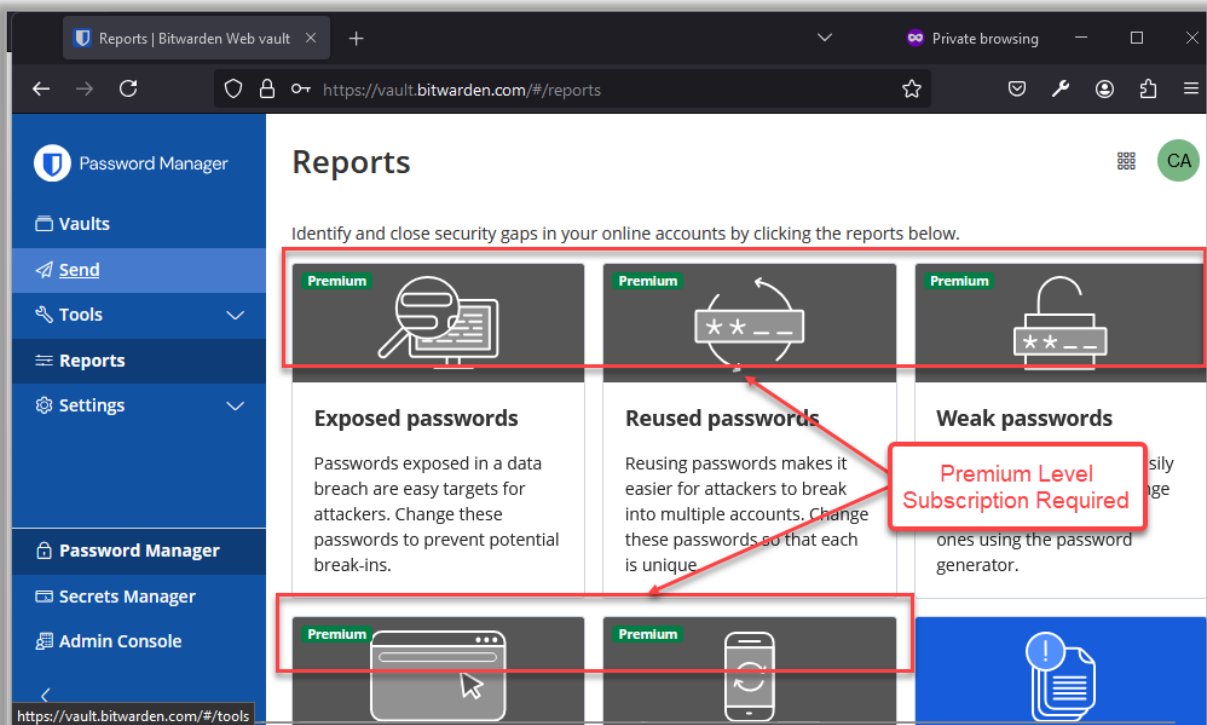
- [https://vault.bitwarden.\[com|eu\]](https://vault.bitwarden.[com|eu]) (Session Storage)
  - `user_{user-id}_billing_accountProfile`
    - *hasPremiumFromAnyOrganization*
    - *hasPremiumPersonally*
- [https://vault.bitwarden.\[com|eu\]/api/accounts/profile](https://vault.bitwarden.[com|eu]/api/accounts/profile)
- [https://vault.bitwarden.\[com|eu\]/api/sync?excludeDomains=true](https://vault.bitwarden.[com|eu]/api/sync?excludeDomains=true)

## TECHNICAL DETAILS

Fracture Labs logged into the web vault with a free user account that was not associated with an organization that had a premium subscription. The team then navigated to the **Reports** section, opened the developer tools of the web browser, and inspected the browser's Session Storage. The settings confirmed the user did not have a premium account and was not associated with an organization that had a premium account.



Also, premium features, such as *Exposed Passwords* and *Weak Passwords*, were not accessible without a premium-level subscription.

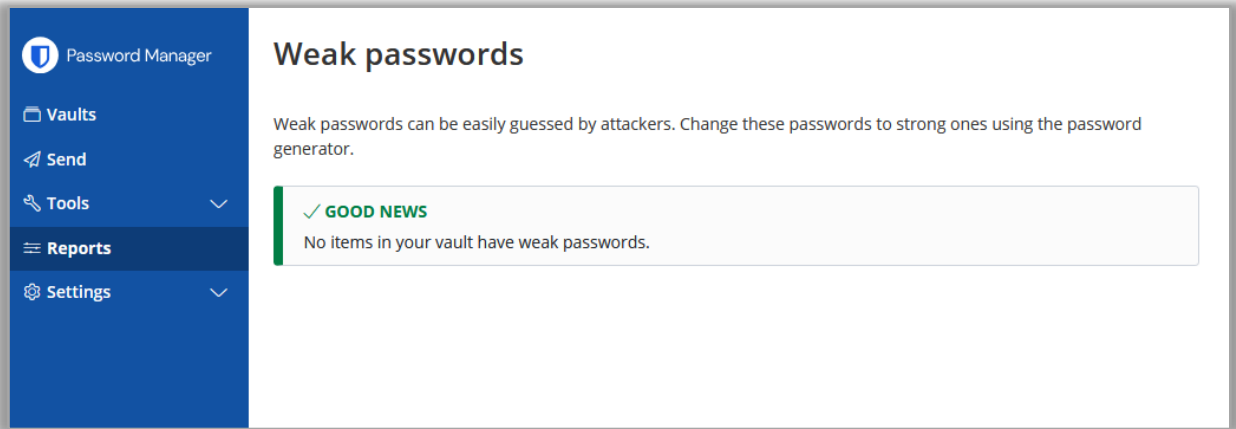


By setting the `hasPremiumPersonally` and `hasPremiumFromAnyOrganization` values to `true`, Fracture Labs was able to access premium features.

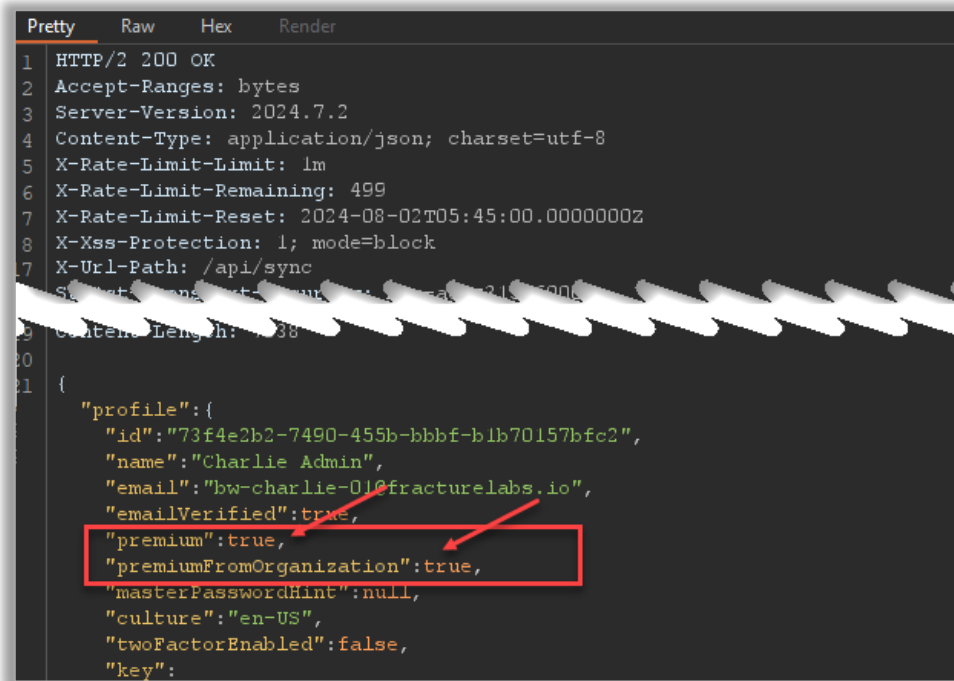
```
73f4e2b2-7490-455b-bbbf-b1b70157bfc2 [{"keys":{"cryptoSymmetricKey":{}}, "profile":{"userId":"73f4e2b2-7490-455b-bbbf-b1b70157bfc2","name":"Charlie Admin","email":"bw-charlie@fracturelabs.com","organizationId":"73f4e2b2-7490-455b-bbbf-b1b70157bfc2","hasPremiumPersonally":true,"hasPremiumFromAnyOrganization":true}]}
global_account_accounts [{"73f4e2b2-7490-455b-bbbf-b1b70157bfc2":{"name":"Charlie Admin","email":"bw-charlie@fracturelabs.com","organizationId":"73f4e2b2-7490-455b-bbbf-b1b70157bfc2","hasPremiumPersonally":true,"hasPremiumFromAnyOrganization":true}]}
global_account_activeAccountId "73f4e2b2-7490-455b-bbbf-b1b70157bfc2"
global_account_activity [{"73f4e2b2-7490-455b-bbbf-b1b70157bfc2":"2024-07-29T19:26:26.475Z"}]
global_clearEvent_lock [{"state":"masterPassword","key":"masterKey","location":"memory"},{"state":"crypto","key":"cryptoSymmetricKey"}]
global_clearEvent_logout [{"state":"kdfConfig","key":"kdfConfig","location":"disk"},{"state":"billing","key":"accountBillingInfo"}]
stateVersion 00
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_billing_accountProfile [{"hasPremiumPersonally":true,"hasPremiumFromAnyOrganization":true}]
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_crypto_everHadUserKey true
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_crypto_organizationKeys [{"3ea73a47-6f05-4415-a386-b1b70159986d":{"type":"organization","key":"4.BctdpOdw7Ym8KzVzjZK2/oLYdFsPaj/UNVba526FIWI+KAdT2.kbQqzbRxqmsf57f5M9NH4w==|08Ajdvkeq4yVzjZK2/oLYdFsPaj/UNVba526FIWI+KAdT2.kbQqzbRxqmsf57f5M9NH4w=="}]}
user_73f4e2b2-7490-455b-bbbf-b1b70157bfc2_crypto_privateKey [{"key":"cryptoSymmetricKey","value":"..."}]
```

The screenshot shows the Bitwarden Reports interface. The left sidebar contains navigation options: Password Manager, Vaults, Send, Tools, Reports (selected), Settings, Password Manager, Secrets Manager, and Admin Console. The main content area is titled 'Reports' and includes the following sections:

- Exposed passwords:** Passwords exposed in a data breach are easy targets for attackers. Change these passwords to prevent potential break-ins.
- Reused passwords:** Reusing passwords makes it easier for attackers to break into multiple accounts. Change these passwords so that each is unique.
- Weak passwords:** Weak passwords can be easily guessed by attackers. Change these passwords to strong ones using the password generator.
- Unsecure websites:** URLs that start with http:// don't use the best available encryption. Change the login URIs for these accounts to https:// for safer browsing.
- Inactive two-step login:** Two-step login adds a layer of protection to your accounts. Set up two-step login using Bitwarden authenticator for these accounts or use an authenticator app.
- Data breach:** Breached accounts can expose your personal information. Secure breached accounts by enabling 2FA or creating a stronger password.



Fracture Labs also unlocked access to premium features by manipulating responses from the API.



## REMEDIATION RECOMMENDATIONS

---

Bitwarden should consider enforcing authorization for premium features on the backend when possible. The frontend should not be the sole source of truth for such validations. The application should avoid storing permissions in client-side storage.

While Bitwarden could potentially remediate this by proxying all premium features through an API with authorization controls, that design could result in potential privacy concerns with the partial hashes going through Bitwarden servers.

## ADDITIONAL RESOURCES

---

API8:2023 Security Misconfiguration

<https://owasp.org/API-Security/editions/2023/en/0xa8-security-misconfiguration/>

Broken Access Control

[https://owasp.org/www-community/Broken\\_Access\\_Control](https://owasp.org/www-community/Broken_Access_Control)

	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	<i>Difficult to Reproduce</i>	Needs Special Circumstances	Easily Reproduced
Exploitability	<i>Expert, Advanced Skills</i>	Skilled	Novice
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

### RISK SUMMARY

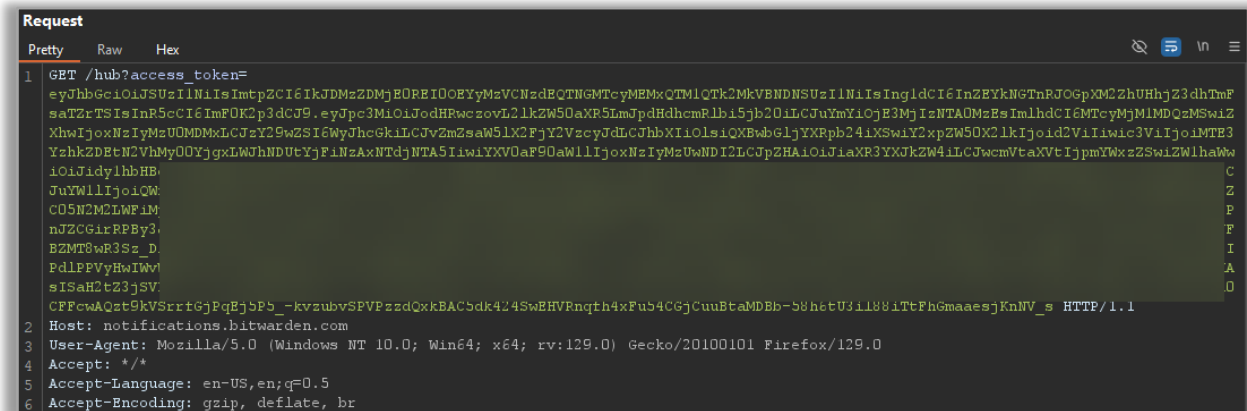
Fracture Labs observed session tokens used to identify and authenticate users passed within the URL to a notification service. Sensitive information within the URL may be logged in various locations like web browsers or proxy devices. Passing session tokens in the URL increases the risk of a threat actor capturing them through vectors, such as social engineering, shoulder surfing, and browser history or logs. A captured session token value could ultimately be used to compromise a user’s session.

### AFFECTED ASSETS

- [https://notifications.bitwarden.\[com|eu\]/hub](https://notifications.bitwarden.[com|eu]/hub)

### TECHNICAL DETAILS

Fracture Labs analyzed proxied web traffic and observed that the application implemented session tokens in the URL when making calls to the <https://notifications.bitwarden.com/hub> endpoint.





## REMEDIATION RECOMMENDATIONS

---

Bitwarden should consider using an alternative mechanism for transmitting session tokens, such as HTTP cookies, headers, or request bodies. Furthermore, the requests should be submitted using the **POST** method as this will help prevent session values from being captured by the previously stated logging mechanisms.

## ADDITIONAL RESOURCES

---

OWASP Broken Access Control

[https://owasp.org/www-community/Broken\\_Access\\_Control](https://owasp.org/www-community/Broken_Access_Control)

CWE-598: Use of GET Request Method with Sensitive Query Strings

<https://cwe.mitre.org/data/definitions/598.html>

WEB.L.03 – MISSING INTEGRITY CHECK FOR EXTERNAL RESOURCES			
	Low	Medium	High
Damage Potential	Trivial Info	<i>Sensitive Info, Defacement</i>	Admin, Full Trust
Reproducibility	<i>Difficult to Reproduce</i>	Needs Special Circumstances	Easily Reproduced
Exploitability	<i>Expert, Advanced Skills</i>	Skilled	Novice
Affected Users	Few Users	Some Users	<i>All Users</i>
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

## RISK SUMMARY

Fracture Labs observed that the Bitwarden marketing site loaded an external script without performing subresource integrity (SRI) checks. SRI is a security feature that allows browsers to verify that resources they fetch (e.g., from a CDN) are delivered without unexpected manipulation by providing a cryptographic hash that the browser can use to check the integrity of the fetched file.

A threat actor with access to the resource files via a supply chain attack or with the ability to intercept network communication could inject malicious content into these resources, compromising the application's security. For example, in 2024, the Polyfill CDN used by hundreds of thousands of websites was hijacked and victim browsers were redirected to malicious and inappropriate websites. This highlights the importance of restricting usage to known-good versions of third-party assets.

## AFFECTED ASSETS

- <https://bitwarden.com>
  - `script src="https://cmp.osano.com/16BZFqRx1MZfx204V/779c3da6-bffb-49bb-99b5-7e07b4c65269/osano.js">`

## TECHNICAL DETAILS

Fracture Labs discovered this issue by examining the requests and responses between the application and the servers hosting these external resources. Specifically, the HTML source code of several pages included references to an external resource but lacked the integrity attribute in the `<script>` tag. This attribute is used to verify that the resources being loaded have not been altered.

```

function(e) {[ "mousedown", "keydown", "touchstart", "pointerdown" ].forEach((function(t) (return e(t,s,r))))}, p=
"hidden"===document.visibilityState?0:1/0;addEventListener ("visibilitychange", (function e(t) ("hidden"===
document.visibilityState&& (p=t.timeStamp, removeEventListener ("visibilitychange", e, !0))) , !0);o(), self.webVitals
=(firstInputPolyfill:function(e) (i.push(e), u()), resetFirstInputPolyfill:o, get firstHiddenTime() (return p))) ();
32 </script><link rel="alternate" type="application/rss+xml" title="Bitwarden Blog" href="/blog/feed.xml
"/><script src="https://cmp.osano.com/16B2FqRx1M2fx204V/779c3da6-bffb-49bb-99b5-7e07b4c65269/osano.js"></script
><script id="plausible-script" defer="" data-domain="bitwarden.com" src="
https://plausible.io/js/script.outbound-links.file-downloads.tagged-events.pageview-props.js"></script></head><
body><noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-NF32QCS" height="0" width="0" style
="display: none; visibility: hidden" aria-hidden="true"></iframe></noscript><div id="__gatsby"><div style="
outline: none" tabindex="-1" id="gatsby-focus-wrapper"><style data-emotion="css 16lms4d">.css-16lms4d:where (
.dark, .dark *) {--tw-bg-opacity:1;background-color:rgb(40 42 49 / var(--tw-bg-opacity));}</style><div class="
css-16lms4d"><style data-emotion="css 1jrd2qn">.css-1jrd2qn(position:absolute;left:0px;right:0px;top:0px;
z-index:0;display:none;width:100%;}@media (min-width: 1024px) {.css-1jrd2qn(display:block;)}.css-1jrd2qn figure (
width:100%;).css-1jrd2qn img(width:100%;).css-1jrd2qn svg(width:100%;fill:none;)</style><div class="css-1jrd2qn
"><style data-emotion="css ovfuqv">.css-ovfuqv(position:relative;max-height:800px;overflow:hidden;).css-ovfuqv
">after(content:var(--tw-content);position:absolute;left:0px;right:0px;bottom:0px;overflow:visible;background-image:

```

## REMEDIATION RECOMMENDATIONS

Bitwarden should consider implementing Subresource Integrity (SRI) for all external resources. SRI helps ensure that files loaded from third-party servers have not been tampered with by including a cryptographic hash in the integrity attribute of the resource’s HTML tag. This way, if the resource content does not match the specified hash, the browser will block it from loading. An example on how to implement this for an external script is below:

```

<script src="https://example.com/resource.js" integrity="sha384-
oqVuAfXRRKap7fdgcCY5uykM6+R9GhXrXvDvnoBL5P+RYvjxVLEDhxo0c1Lg8KMoy9"
crossorigin="anonymous"></script>

```

In scenarios where SRI is not feasible (for example with dynamic Google Fonts), Bitwarden should consider self-hosting static versions of the files.

## ADDITIONAL RESOURCES

### Subresource Integrity - Security on the web

[https://developer.mozilla.org/en-US/docs/Web/Security/Subresource\\_Integrity](https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity)

### Subresource Integrity (SRI)

<http://owasp.org/www-community/controls/SubresourceIntegrity>

### Polyfill Supply Chain Attack Hits Over 100k Websites

<https://www.securityweek.com/polyfill-supply-chain-attack-hits-over-100k-websites/>

<b>WEB.L.04 – WEAK PASSWORD POLICY FOR VAULT EXPORTS</b>			
	Low	Medium	High
Damage Potential	<i>Trivial Info</i>	Sensitive Info, Defacement	Admin, Full Trust
Reproducibility	<i>Difficult to Reproduce</i>	Needs Special Circumstances	Easily Reproduced
Exploitability	Expert, Advanced Skills	Skilled	<i>Novice</i>
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

## RISK SUMMARY

Fracture Labs discovered that the encrypted Vault Export function did not require users to use a strong password to create encrypted JSON files. While a password strength meter was displayed during this process, the application still allowed weak passwords, including single character passwords.

A threat actor with access to the exported vault secrets may be able to brute force it to gain access to the credentials and secrets of other systems.

## AFFECTED ASSETS

- [https://vault.bitwarden.\[com|eu\]/#/tools/export](https://vault.bitwarden.[com|eu]/#/tools/export)

## TECHNICAL DETAILS

Fracture Labs exported a vault to an encrypted JSON file and specified a weak password. While a password meter indicated the password was weak, the system still allowed a password containing only a single character.

**Export vault**

ⓘ **EXPORTING INDIVIDUAL VAULT**  
Only the individual vault items associated with bw-alpha-01@fracturelabs.io will be exported. Organization vault items will not be included. Only vault item information will be exported and will not include associated attachments.

Export from (required)  
My vault

File format (required)  
.json (Encrypted)

Export type

Account restricted  
Use your account encryption key, derived from your account's username and Master Password, to encrypt the export and restrict import to only the current Bitwarden account.

Password protected  
Set a file password to encrypt the export and import it to any Bitwarden account using the password for decryption.

File password (required)  
• [eye icon]  
This password will be used to export and import this file  
Weak

Confirm file password (required)  
• [eye icon]

Confirm format

## REMEDATION RECOMMENDATIONS

---

Bitwarden should consider enforcing strong password requirements before exporting vaults.

## ADDITIONAL RESOURCES

---

OWASP: Authentication - Implement Proper Password Strength Controls

[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html#implement-proper-password-strength-controls](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#implement-proper-password-strength-controls)

CWE-521: Weak Password Requirements

<https://cwe.mitre.org/data/definitions/521.html>

# BEST PRACTICE IMPROVEMENT OPPORTUNITIES

## WEB.BPI.01 – DISABLE SOFTWARE NAME AND VERSION REPORTING

### RISK SUMMARY

Fracture Labs observed several instances of information disclosures via HTTP headers that provided insight into software names and versions. A threat actor could leverage this information disclosure vulnerability to launch targeted attacks against the application. The IP address and header values could also be logged in an index such as *Shodan.io*, which would increase the likelihood of attack should that version of software contain a vulnerability in the future.

### AFFECTED ASSETS

- [http\[s\]://admin.bitwarden.\[com|eu\]](http[s]://admin.bitwarden.[com|eu])
- [http\[s\]://artifacts.bitwarden.com/soap](http[s]://artifacts.bitwarden.com/soap)
- [http\[s\]://api.bitwarden.\[com|eu\]](http[s]://api.bitwarden.[com|eu])
  - [/IP](#)
  - [/public/collections](#)
- [http\[s\]://bitwarden.com](http[s]://bitwarden.com)
- [http\[s\]://billing.bitwarden.\[com|eu\]](http[s]://billing.bitwarden.[com|eu])
- [http\[s\]://events.bitwarden.\[com|eu\]](http[s]://events.bitwarden.[com|eu])
- [http\[s\]://icons.bitwarden.net](http[s]://icons.bitwarden.net)
- [http\[s\]://identity.bitwarden.\[com|eu\]](http[s]://identity.bitwarden.[com|eu])
- [http\[s\]://notifications.bitwarden.\[com|eu\]](http[s]://notifications.bitwarden.[com|eu])
- [http\[s\]://scim.bitwarden.\[com|eu\]/alive](http[s]://scim.bitwarden.[com|eu]/alive)
- [http\[s\]://push.bitwarden.\[com|eu\]](http[s]://push.bitwarden.[com|eu])
- [http\[s\]://sso.bitwarden.com](http[s]://sso.bitwarden.com)
- [http\[s\]://vault.bitwarden.\[com|eu\]](http[s]://vault.bitwarden.[com|eu])
  - [/api](#)
  - [/api/config/app/vendor.7e42beb4591ce52a4f26.js](#)
  - [/app/vendor.e740196f063808bfc118.js](#)
  - [/theme\\_head.4cb181fc19f2a308ba73.js](#)
  - [/styles.31d6cfe0d16ae931b73c.js](#)
- [http\[s\]://vault.euqa.bitwarden.pw/api/devices/knowndevices](http[s]://vault.euqa.bitwarden.pw/api/devices/knowndevices)
  - [/api/config](#)
  - [/app/vendor.e740196f063808bfc118.js](#)
  - [/styles.31d6cfe0d16ae931b73c.js](#)

- [http\[s\]://vault.qa.bitwarden.pw](http[s]://vault.qa.bitwarden.pw)
  - </api/devices/knowndevices>
  - </api/config>
  - </app/vendor.e740196f063808bfc118.js>
  - </styles.31d6cfe0d16ae931b73c.js>
- [http\[s\]://send.bitwarden.com](http[s]://send.bitwarden.com)

## TECHNICAL DETAILS

Fracture Labs analyzed proxied web traffic and observed several instances of information disclosures via HTTP headers that provided insight into software names and versions.

```
1 HTTP/2 200 OK
2 Content-Type: application/javascript
3 Content-Md5: YHWIBh5M0/ZUCHaRmOr97w==
4 Last-Modified: Wed, 24 Jul 2024 01:49:17 GMT
5 Accept-Ranges: bytes
6 Etag: "0x8DCAB82D45D84EF"
7 X-Ms-Version: 2018-03-28
8 X-Ms-Error-Code: ConditionNotMet
9 X-Ms-Request-Id: b0f201aa-201e-0064-10e0-d443-e0000000
10 Server: Windows-Azure-Web/1.0 Microsoft-HTTPAPI/2.0
11 Age: 2674
12 Date: Wed, 24 Jul 2024 16:30:44 GMT
13 Via: 1.1 varnish
14 X-Served-By: cache-iad-kcgs/200179-IAD, cache-iad-kjyo/100126-IAD
15 X-Cache: HIT, MISS
16 X-Cache-Hits: 158, 0
17 X-Timer: S1721838644.461589,VSO,VE11
18 X-Url-Path: /app/vendor.7e42beb4591ce52a4f26.js
19 Strict-Transport-Security: max-age=31536000
20 Content-Length: 2585983
```

## REMEDIATION RECOMMENDATIONS

Software name and versions in HTTP response headers are commonly part of the default configuration of applications and servers. Bitwarden should ensure that web servers, applications, and other services do not disclose version numbers in banners, error messages, or page footers. This can usually be configured in the application or server settings.

## ADDITIONAL RESOURCES

OWASP A05:2021 – Security Misconfiguration

[https://owasp.org/Top10/A05\\_2021-Security\\_Misconfiguration](https://owasp.org/Top10/A05_2021-Security_Misconfiguration)

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

<https://cwe.mitre.org/data/definitions/200.html>

## WEB.BPI.02 – RESTRICT FILE UPLOADS

### RISK SUMMARY

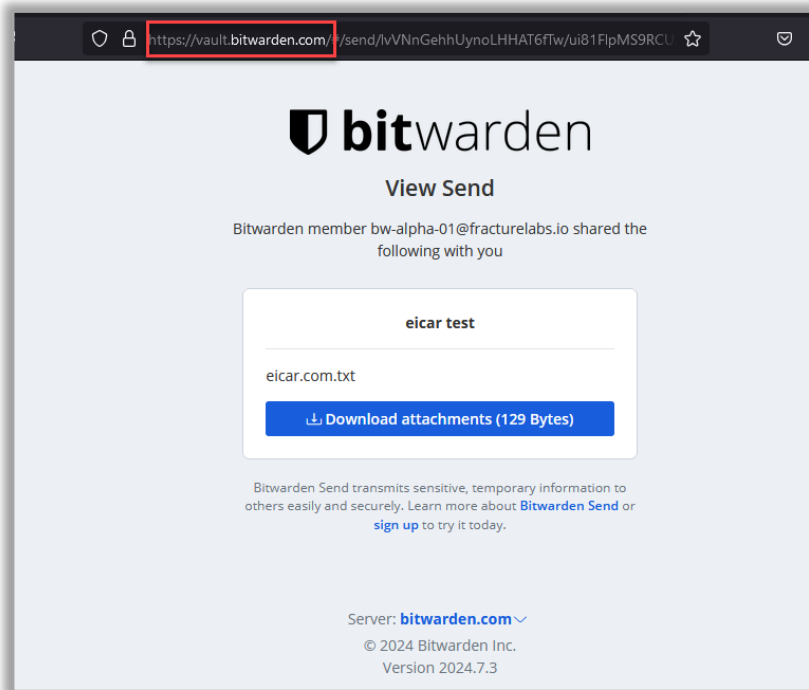
Fracture Labs discovered that the application allowed users to upload and share files without any restrictions. Although this functionality did not lead to a direct system compromise, a threat actor could still abuse it to upload and share malicious files, such as scripts or malware, with other users. While threat actors could use a myriad of file sharing services to distribute malware, abuse of this functionality could take advantage of victim’s trust in the `vault.bitwarden.com` or `vault.bitwarden.eu` domain names.

### AFFECTED ASSETS

- `https://vault.bitwarden.[com|eu]/send`

### TECHNICAL DETAILS

Fracture Labs successfully uploaded and shared `eicar.com.txt`, a file deliberately identified as malicious by many antivirus vendors.





## REMEDIATION RECOMMENDATIONS

---

Bitwarden should consider enforcing file type restrictions and perform deep content inspection to filter out potentially malicious content.

## ADDITIONAL RESOURCES

---

CWE-434: Unrestricted Upload of File with Dangerous Type

<https://cwe.mitre.org/data/definitions/434.html>

Unrestricted File Upload

[https://owasp.org/www-community/vulnerabilities/Unrestricted\\_File\\_Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload)

WEB.BPI.03 – REMOVE PRIVATE IP ADDRESSES FROM API RESPONSES			
	Low	Medium	High
Damage Potential	<i>Trivial Info</i>	Sensitive Info, Defacement	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	<b>Easily Reproduced</b>
Exploitability	Expert, Advanced Skills	Skilled	<b>Novice</b>
Affected Users	<i>Few Users</i>	Some Users	All Users
Discoverability	Obscure	<i>Limited</i>	Obvious, Published

## RISK SUMMARY

Fracture Labs discovered a private IP address disclosed by an API response message. A threat actor could leverage this vulnerability to gain insights into the internal network architecture, which could help facilitate other targeted attacks against the infrastructure.

## AFFECTED ASSETS

- [https://api.bitwarden.\[com|eu\]](https://api.bitwarden.[com|eu])

## TECHNICAL DETAILS

Fracture Labs discovered the vulnerability by browsing to the URL and inspecting the responses.

