Bitwarden Web App and Network Security Report

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

Table of Contents

Summary

In July 2025, 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 one month to reach total coverage of the system under review.

Eleven issues were discovered during the audit. Six issues were resolved post-assessment. Four 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

2025.EXT.L.01: Unauthenticated Varnish Cache Purge

Status: Resolved post-assessment.

During the testing period, the affected hosts were undergoing a planned migration from a standard virtual server environment to a managed environment. At the time of testing, the managed environment's CDN policy had not yet been fully hardened, and the PURGE HTTP verb had not been disabled. Following completion of testing, the hosts were reverted to their original configuration, which is not susceptible to this issue. The managed environment will be reintroduced only after additional security validation is completed.

2025.EXT.L.02: Subdomain/DNS Misconfiguration

Status: Resolved mid-assessment.

This DNS record was a legacy artifact from prior QA testing. It was remediated as part of an internal Bitwarden initiative to improve DNS hygiene. As part of this effort, DNS records will be managed through standardized, automated, and auditable processes to prevent similar issues in the future.

2025.EXT.BPI.01: Standardize Certificate Lifecycle Controls

Status: Resolved post-assessment.

The hosts under the affected domain are used by Bitwarden's QA teams for internal testing purposes. Following completion of the assessment, this instance was updated to use certificates issued by Let's Encrypt. As part of planned improvements, access to all QA testing hosts will be restricted from public access once the appropriate controls are fully implemented.

2025.EXT.BPI.02: Restrict Access to Self-Hosted Development Environments

Status: Resolved post-assessment with additional improvements underway.

The affected hosts are used by Bitwarden's QA teams to test interactions between Bitwarden servers and clients, including mobile applications. These test scenarios require access from multiple devices and automated testing systems. Following completion of testing, the hosts were reviewed and unnecessary exposed services were removed, leaving only those required for

testing. In parallel, efforts are underway to migrate the environment to a more restrictive configuration that will be accessible only to authorized Bitwarden staff.

2025.WEB.M.01: Cleartext Storage of Sensitive Information in Memory

Status: Accepted.

Bitwarden is designed to minimize the presence of the master password in memory beyond what is strictly necessary for operation. During testing, it was observed that the master password could be found within the Chrome browser's memory. This occurrence was outside the sandboxed memory space of the Bitwarden browser extension itself and instead resided within Chrome's broader process memory. Bitwarden's control is limited to the memory managed by the extension and does not extend to how browsers internally handle or retain data.

As a result, Bitwarden has accepted this risk, as it is outside the scope of changes that can be made within the Bitwarden product. Addressing this behavior would require changes at the browser level and would similarly affect other browser-based password managers.

2025.WEB.M.02: Audit/Event Log Tampering

Status: Accepted.

This functionality depends on accurate reporting from the client. A malicious actor could potentially suppress or manipulate this data, resulting in incomplete or misleading records. However, because the relevant information necessarily exists on the client when the vault is decrypted, this logic cannot be enforced server side. To address this limitation, Bitwarden has provided additional clarification and context in its documentation, available at https://bitwarden.com/help/event-logs/#when-events-are-saved.

2025.WEB.L.01: Non-Time Constant Comparison of Security Token

Status: Resolved post-assessment.

Pull requests:

- https://github.com/bitwarden/server/pull/6279

Bitwarden updated the comparison logic to use the recommended `CoreHelpers.FixedTimeEquals()` method. This implementation leverages Microsoft's `CryptographicOperations.FixedTimeEquals()` function from the standard C# libraries to perform time constant comparisons, reducing the risk of potential timing-based attacks.

2025.WEB.L.02: Existing Session Not Invalidated After Manual Logout

Status: Accepted.

Bitwarden uses short lived JWT access tokens, with a 60 minute lifetime, issued by an OAuth 2.0 authorization server in a stateless architecture. Under this model, access tokens cannot be invalidated prior to expiration. This approach aligns with common OAuth 2.0 implementations. Transitioning to a stateful token architecture would require significant architectural changes and introduce additional complexity without providing sufficient security benefit. Bitwarden considers the 60-minute access token lifetime to both appropriately limit potential exposure and align with prevailing industry standards.

2025.WEB.BPI.01: Disable Software Name and Version Reporting

Status: Resolved post-assessment.

Bitwarden addressed this finding through multiple corrective actions. First, certain hosts were removed from the external network perimeter entirely. For the remaining hosts, access was restricted to route only through the CDN, with additional controls applied to limit exposure of sensitive headers. Over the longer term, Bitwarden plans to further restrict these hosts from public access altogether.

2025.WEB.BPI.02: Disable Verbose Error Messages

Status: Accepted.

As an open source platform, Bitwarden benefits from detailed error messages to support effective troubleshooting between servers and clients. Because the source code is publicly available, these messages do not expose sensitive implementation details, and they provide meaningful value to Bitwarden engineers, customers, and security researchers investigating issues through the bug bounty program.

In addition, Bitwarden logs errors server side for operational monitoring and reporting. Controls are in place to ensure that, while clients may receive detailed error information, any sensitive data is appropriately redacted or excluded from server side logs.

2025.WEB.BPI.03: Implement Fine-Grained API Scope Assignments

Status: Accepted, but under planning and research.

Fine grained API scope assignments have been requested by multiple customers. While no immediate remediation is planned for this finding, research and development efforts are underway to evaluate and introduce this capability in a future release.

FINDINGS – EXTERNAL NETWORK

ATTACK NARRATIVE

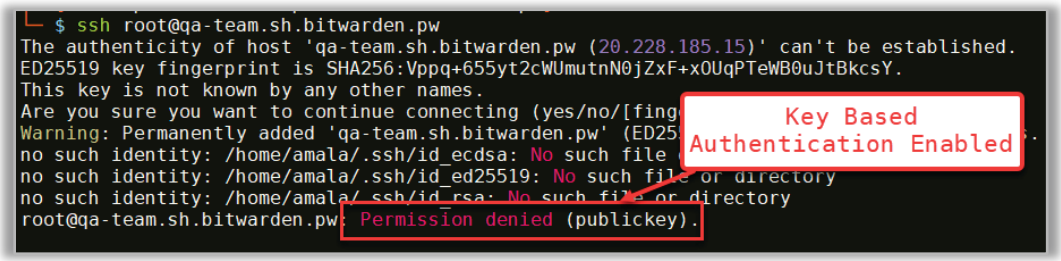
The primary objective of the external penetration test was to identify vulnerabilities or threats that could allow a malicious actor to compromise systems, application data, or place the applications and infrastructure at risk. All external testing was conducted from an unauthenticated perspective simulating the actions of a threat actor who did not have access to any client systems or credentials.

Bitwarden provided a comprehensive list of wildcard domains and subdomains to be scanned for vulnerabilities and misconfigurations from a Fracture Labs cloud-based attack box, including `*.bitwarden.com`, `*.bitwarden.eu`, `*.bitwarden.net` and `*.bitwarden.pw` domains.

Fracture Labs began the assessment by performing comprehensive open-source intelligence (OSINT) gathering on the provided target domains. This reconnaissance phase focused on collecting publicly available information about Bitwarden's external infrastructure footprint. Fracture Labs utilized automated and manual enumeration techniques to systematically discover subdomains across the authorized domains. The intelligence gathering activity revealed an extensive external infrastructure with numerous development, testing, and production environments across multiple geographic regions and cloud providers.

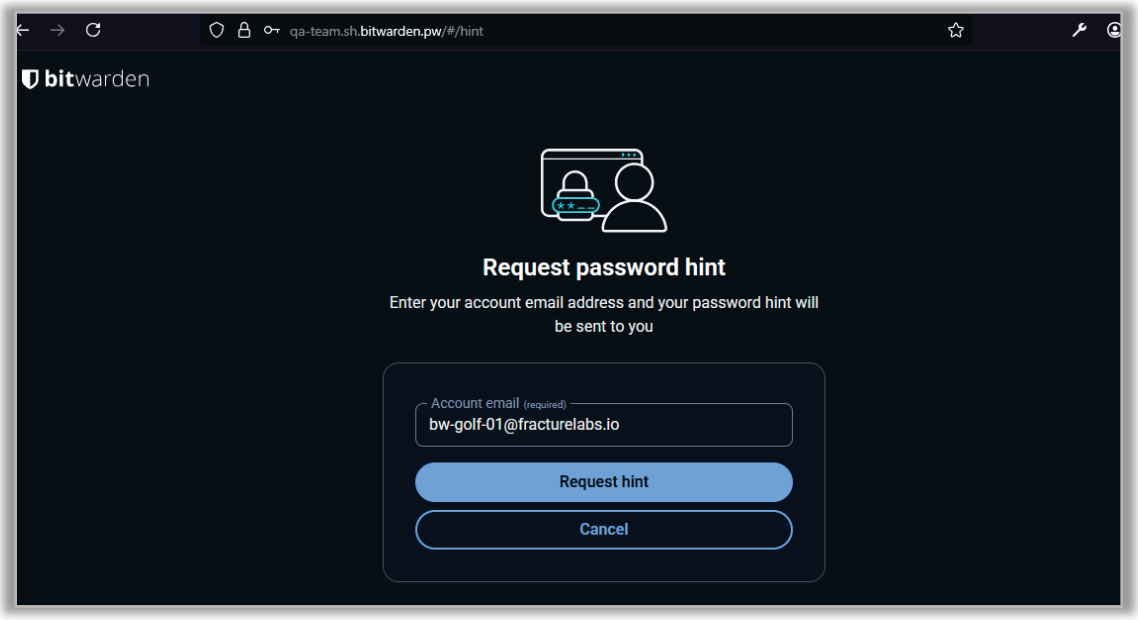
During the enumeration process, Fracture Labs discovered multiple interesting patterns in the subdomain structure. These included employee-named development instances under the `.sh.bitwarden.pw` domain space, environment-specific deployments such as `poc`, `poc2`, `qa`, `usdev`, `euqa`, and `eudevtest` environments, and Kubernetes controllers and load balancers across different deployment zones. The comprehensive enumeration resulted in identification of over 200 unique subdomains requiring further analysis.

Following the reconnaissance phase, Fracture Labs conducted port scanning across all identified targets to discover open services and potential attack vectors. The port scanning revealed consistent patterns across different types of infrastructure, with most web-accessible systems exposing TCP ports 80, 443, 8080, and 8443. Additionally, Fracture Labs identified numerous systems with SSH services running on TCP port 22, although these systems required public key authentication and properly rejected password-based authentication attempts.



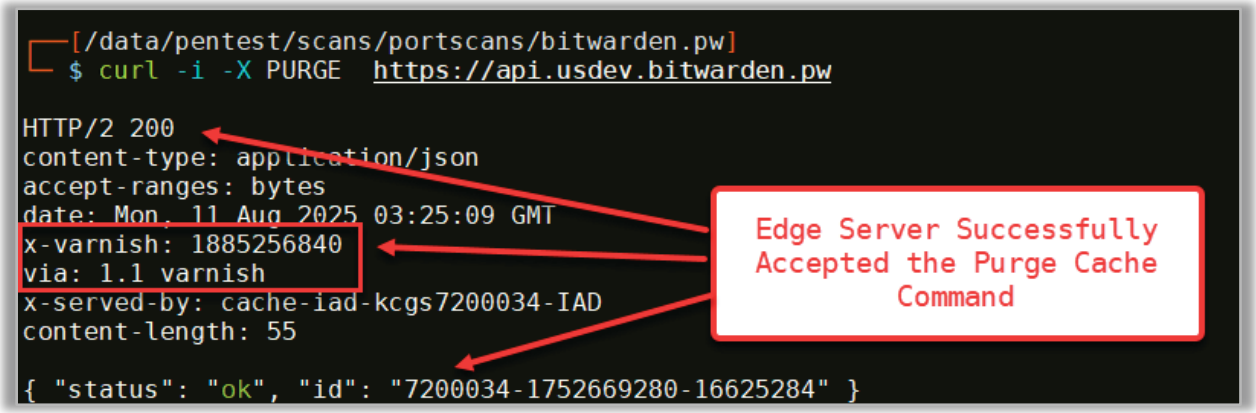
Fracture Labs identified a subdomain misconfiguration for a developmental domain, `localhost.bitwarden.pw`, which redirected to a domain unrelated to Bitwarden (see 2025.EXT.L.02 – Subdomain/DNS Misconfiguration). This external redirect introduced the risk of subdomain takeover, phishing, or brand misuse and warrants review of DNS and redirect rules.

Building on these findings, Fracture Labs performed detailed analysis of the numerous developmental `.sh.bitwarden.pw` subdomains, which appeared to be individual employee development or testing environments exposed (see 2025.EXT.BPI.02 – Restrict Access to Self-Hosted Development Environments). These instances presented self-hosted Bitwarden vault interfaces requiring authentication, with most displaying standard login pages requesting email and master password credentials. Accounts registered on `vault.bitwarden.com` were not valid on these instances.

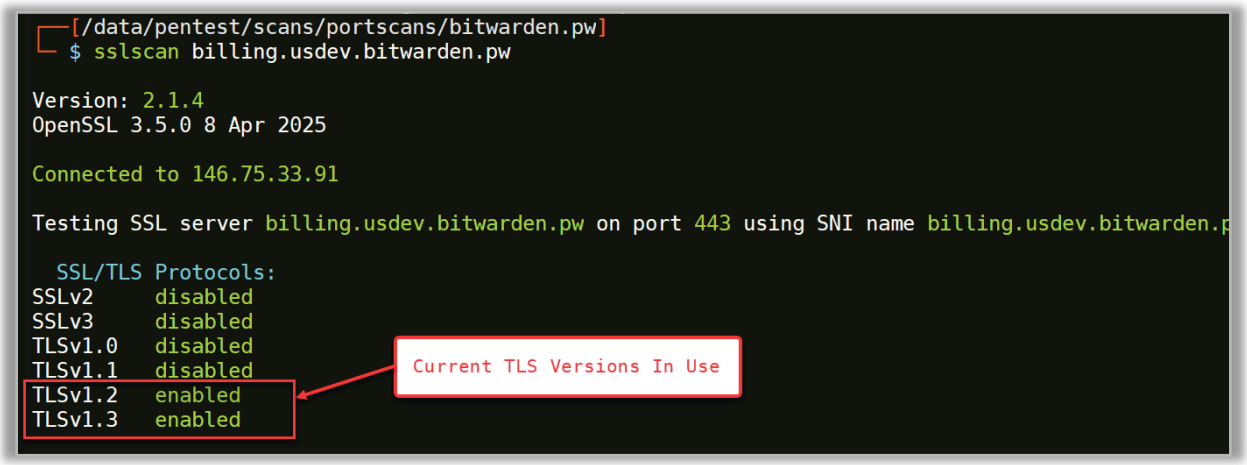


Fracture Labs also discovered that Varnish cache servers accepted unauthenticated `PURGE` requests, allowing external users to clear cached content without proper authorization (see 2025.EXT.L.01 – Unauthenticated Varnish Cache Purge). With no authentication required for cache purging, threat

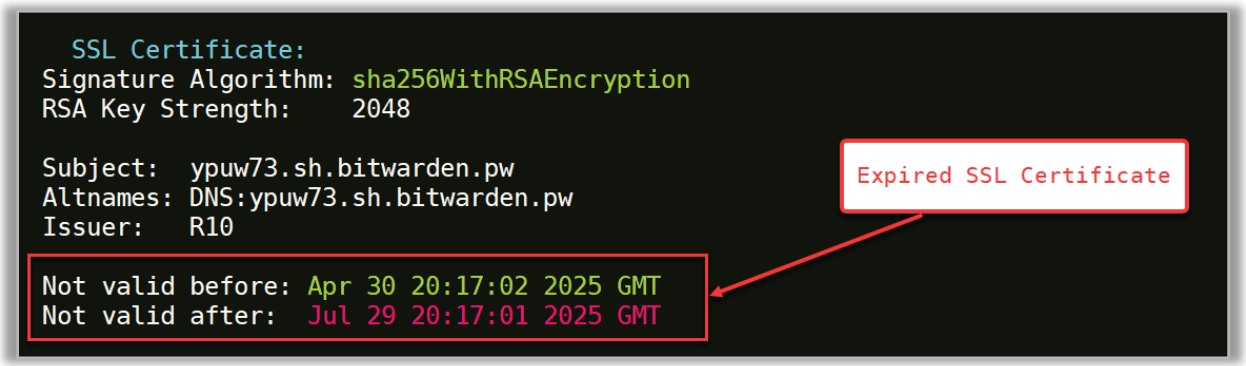
actors may be able to alter stored content, repeatedly clear the cache to slow down performance, or overload backend systems until they can no longer respond. These actions can disrupt normal operations and impact the application's availability and reliability.



Following this, Fracture Labs conducted comprehensive SSL/TLS configuration analysis across all accessible HTTPS services to identify potential cryptographic weaknesses or configuration issues. Fracture Labs discovered that Bitwarden's external infrastructure maintained strong cryptographic configurations with current TLS versions and appropriate cipher suites.



However, Fracture Labs discovered two development hosts utilizing expired SSL certificates (see 2025.EXT.BPI.01 - Standardize Certificate Lifecycle Controls Across All Environments).



Fracture Labs also tried various common attack techniques against the identified infrastructure, including directory and file enumeration, default credential testing, and common vulnerability exploitation. Extensive requests were made across the target infrastructure to find exposed administrative panels, configuration files, and other sensitive resources. No vulnerabilities were identified.

Lastly, Fracture Labs executed a vulnerability scan across all identified targets. The scan confirmed the results of manual testing and did not reveal any findings beyond those already identified.

Overall, the external infrastructure showed exceptional resilience to external attack vectors with robust access controls, proper authentication mechanisms, and strong cryptographic implementations. The systems performed well under comprehensive security testing and showed evidence of mature security practices throughout the development and deployment lifecycle. The minimal findings indicated a strong external security posture with effective defensive strategies protecting critical assets against external threats.

HIGH-RISK FINDINGS

No High-Risk findings were identified.

MEDIUM-RISK FINDINGS

No Medium-Risk findings were identified.

LOW-RISK FINDINGS

2025.EXT.L.01 – UNAUTHENTICATED VARNISH CACHE PURGE			
	Low	Medium	High
Damage Potential	***Trivial Info***	Sensitive Info, Defacement	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	***Easily Reproduced***
Exploitability	Expert, Advanced Skills	***Skilled***	Novice
Affected Users	Few Users	Some Users	***All Users***
Discoverability	Obscure	***Limited***	Obvious, Published

RISK SUMMARY

Fracture Labs identified that Varnish cache servers accepted unauthenticated `PURGE` requests, allowing external users to clear cached content without proper authorization. This misconfiguration enables threat actors to perform cache poisoning attacks, degrade application performance through cache clearing, and potentially cause denial of service conditions by forcing backend servers to regenerate content repeatedly. Unauthenticated cache purge capabilities can significantly impact application availability and performance while providing threat actors with a mechanism to manipulate cached content and disrupt normal operations.

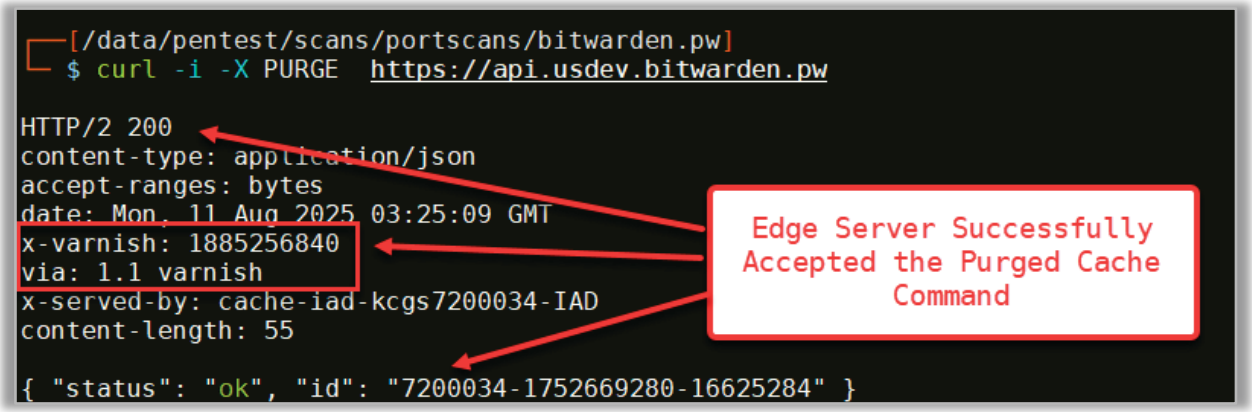
AFFECTED ASSETS

- The list of affected assets has been provided to Bitwarden separately

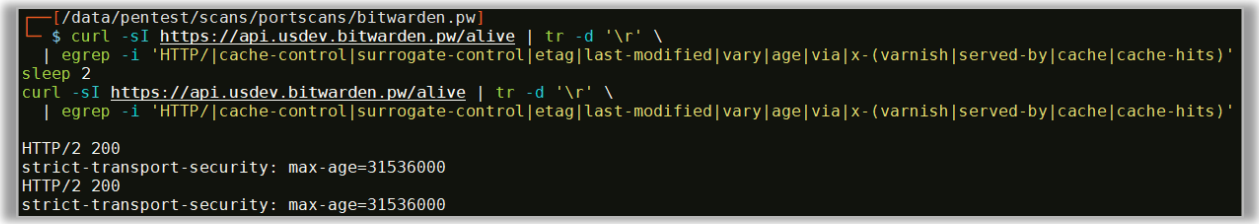
TECHNICAL DETAILS

Fracture Labs discovered that Varnish cache servers respond to HTTP `PURGE` requests without requiring authentication or authorization checks. This allows external users to clear specific cached objects or entire cache contents, forcing the application to regenerate content from backend servers.

Fracture Labs issued baseline requests to identify cache signals and confirm whether responses were being served from an intermediary cache. The response received *HTTP/2 200* with JSON body `{ "status": "ok", "id": "…" }` and cache headers such as `Via: 1.1 varnish`, `X-Varnish: <id>`, `X-Served-By: cache-…` which confirmed that an external, unauthenticated client could submit a `PURGE` instruction to the edge.



To demonstrate data-plane impact (MISS→HIT transition), Fracture Labs attempted to identify a cacheable object on the same host, starting with common candidates such as `/robots.txt`, `/favicon.ico`, `/assets/app.css`, `/static/app.js`, `/config`, `/IP`, `/alive` and a few others. No `Age/X-Cache/Via` indicators were returned for these API paths, indicating they were not cached at the edge.



Although the tested paths did not exhibit cache hits, the `PURGE` acceptance itself demonstrated a control weakness at the edge which should be addressed to prevent any elevation of origin loads of pages that may be cached in the future.

REMEDIATION RECOMMENDATIONS

Bitwarden should consider denying unauthenticated `PURGE/BAN` at the edge by limiting cache invalidation to trusted sources only.

ADDITIONAL RESOURCES

Purging and Banning

https://varnish-cache.org/docs/trunk/users-guide/purging.html

Access Control Lists

https://www.varnish-software.com/developers/tutorials/varnish-configuration-language-vcl/#access-control-lists

2025.EXT.L.02 – SUBDOMAIN/DNS MISCONFIGURATION [REMEDIATED]			
	Low	Medium	High
Damage Potential	***Trivial Info***	Sensitive Info, Defacement	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	***Easily Reproduced***
Exploitability	Expert, Advanced Skills	***Skilled***	Novice
Affected Users	Few Users	Some Users	***All Users***
Discoverability	Obscure	***Limited***	Obvious, Published

RISK SUMMARY

Fracture Labs observed that a development (non-Production) domain, `localhost.bitwarden.pw`, redirected to a domain unrelated to Bitwarden. This likely reflected a DNS misconfiguration or incomplete record decommissioning. No evidence of compromise was identified; however, redirecting from a Bitwarden owned subdomain introduces brand and trust risk.

The redirect appeared to be configured at the DNS or web server level and consistently directed users to the unrelated healthcare domain. This behavior suggests either misconfigured DNS records, abandoned cloud service configurations, or incomplete cleanup following infrastructure changes.

Threat actors could abuse this path for phishing or traffic hijacking if the destination is compromised or repointed in the future.

AFFECTED ASSETS

- localhost.bitwarden.pw (104.20.22.205)

TECHNICAL DETAILS

Fracture Labs identified the misconfiguration during external subdomain enumeration and testing. When accessing `localhost.bitwarden.pw` through both web browser and command-line tools, users are automatically redirected to an unrelated organization.

REMEDIATION RECOMMENDATIONS

Bitwarden should review DNS records for `localhost.bitwarden.pw` and others to ensure they point to Bitwarden infrastructure or remove the DNS record entirely if the subdomain is no longer needed. Additionally, Bitwarden should implement regular DNS auditing procedures to identify and remediate similar misconfigurations across their domain infrastructure.

REMEDIATION VERIFICATION RESULTS

Fracture Labs verified the DNS record no longer existed on August 22, 2025.

ADDITIONAL RESOURCES

Mitigate DNS Infrastructure Tampering

https://www.cisa.gov/resources-tools/resources/mitigate-dns-infrastructure-tampering

OWASP: Unvalidated Redirects and Forwards

https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

BEST-PRACTICE IMPROVEMENT OPPORTUNITIES

2025.EXT.BPI.01 – STANDARDIZE CERTIFICATE LIFECYCLE CONTROLS

RISK SUMMARY

Expired SSL/TLS certificates represent a significant security and availability risk that can disrupt service availability and compromise data transmission security. When certificates expire, legitimate users receive security warnings and may be unable to access services, while the expired state can force downgrades to unencrypted communications or acceptance of invalid certificates. This creates opportunities for machine-in-the-middle attacks, data interception, and service disruption that can impact both security posture and business operations.

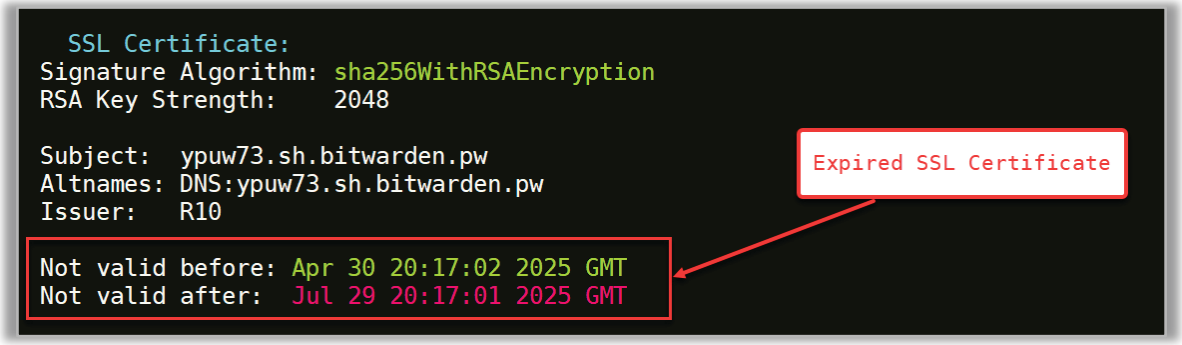
Fracture Labs found two development systems with expired SSL/TLS certificates. When accessing services with expired certificates, browsers display prominent security warnings advising users not to continue. These warnings can prevent legitimate access to services and may train users to ignore certificate warnings, reducing overall security awareness. In automated systems and API integrations, expired certificates can cause complete service failures and application errors.

AFFECTED ASSETS

- clienttest.sh.bitwarden.pw (172.172.208.197)
- ypuw73.sh.bitwarden.pw (20.127.103.176)

TECHNICAL DETAILS

Certificate expiration analysis revealed certificates that had passed their validity dates, rendering the associated services non-functional or requiring users to bypass security warnings.



REMEDIATION RECOMMENDATIONS

Bitwarden should obtain valid SSL/TLS certificates from a trusted Certificate Authority (CA). Additionally, Bitwarden should consider implementing a monitoring or alerting mechanism to notify administrators ahead of certificate expiration dates, minimizing service disruptions and potential security risks.

ADDITIONAL RESOURCES

CWE-298: Improper Validation of Certificate Expiration

https://cwe.mitre.org/data/definitions/298.html

OWASP Transport Layer Protection Cheat Sheet

https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

2025.EXT.BPI.02 – RESTRICT ACCESS TO SELF-HOSTED DEVELOPMENT ENVIRONMENTS

RISK SUMMARY

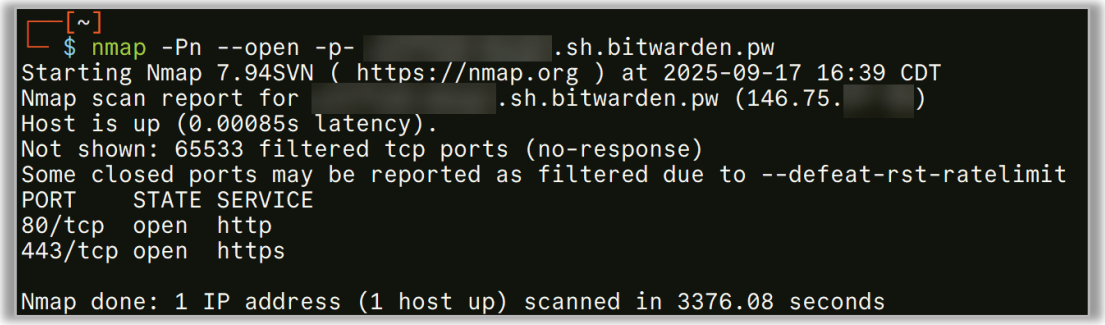
Fracture Labs identified self-hosted development infrastructure that was publicly accessible without any network-level access controls or IP restrictions. The development systems lacked proper network segmentation and were reachable directly from the Internet. Development infrastructure typically receives less frequent security updates and patch management compared to production systems, creating an expanded attack surface. A threat actor may be able to leverage these exposed systems as an initial foothold to conduct reconnaissance, extract sensitive configuration data, or pivot to otherwise inaccessible network resources.

AFFECTED ASSETS

- *.sh.bitwarden.pw

TECHNICAL DETAILS

The `.sh.bitwarden.pw` hosts were accessible from untrusted hosts.



REMEDIATION RECOMMENDATIONS

Bitwarden should consider implementing network-level access controls to restrict development infrastructure access to authorized IP ranges or VPN connections only. Development environments should be placed behind a firewall or within a separate network segment that requires explicit approval for external access. If not already done, Bitwarden should also consider establishing a regular patching schedule for development systems that aligns with production maintenance windows to ensure timely security updates are applied.

ADDITIONAL RESOURCES

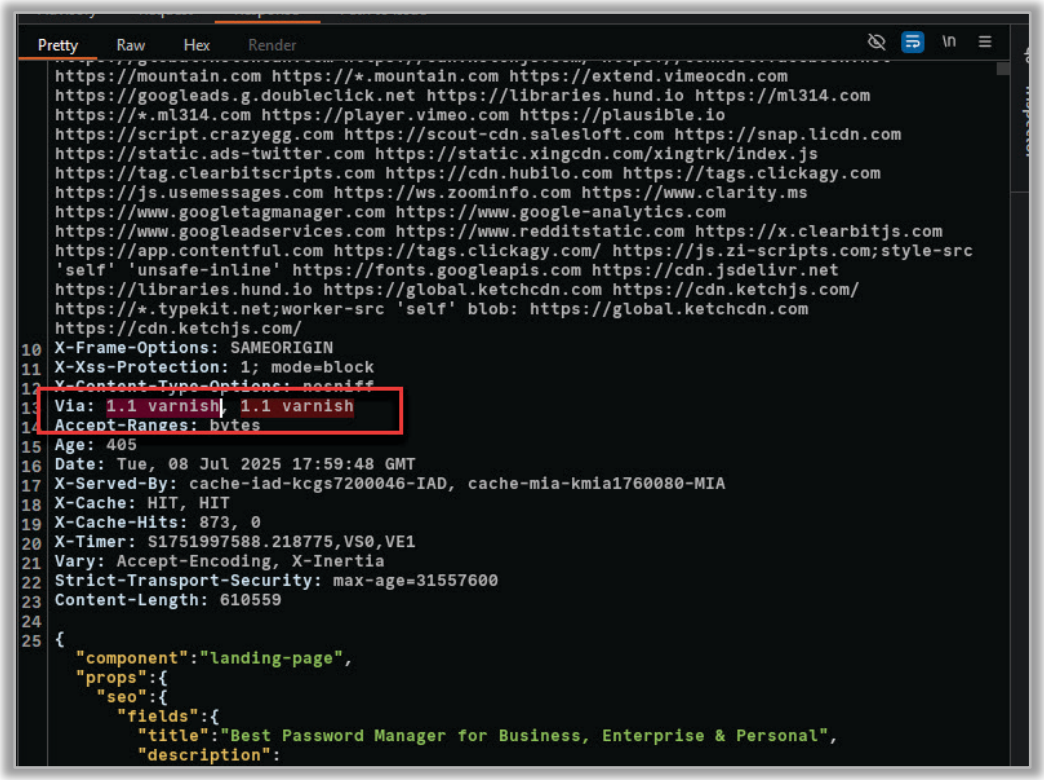
CWE-668: Exposure of Resource to Wrong Sphere
https://cwe.mitre.org/data/definitions/668.html

FINDINGS – WEB ASSESSMENT

ATTACK NARRATIVE

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 instances from an external Fracture Labs attack box.

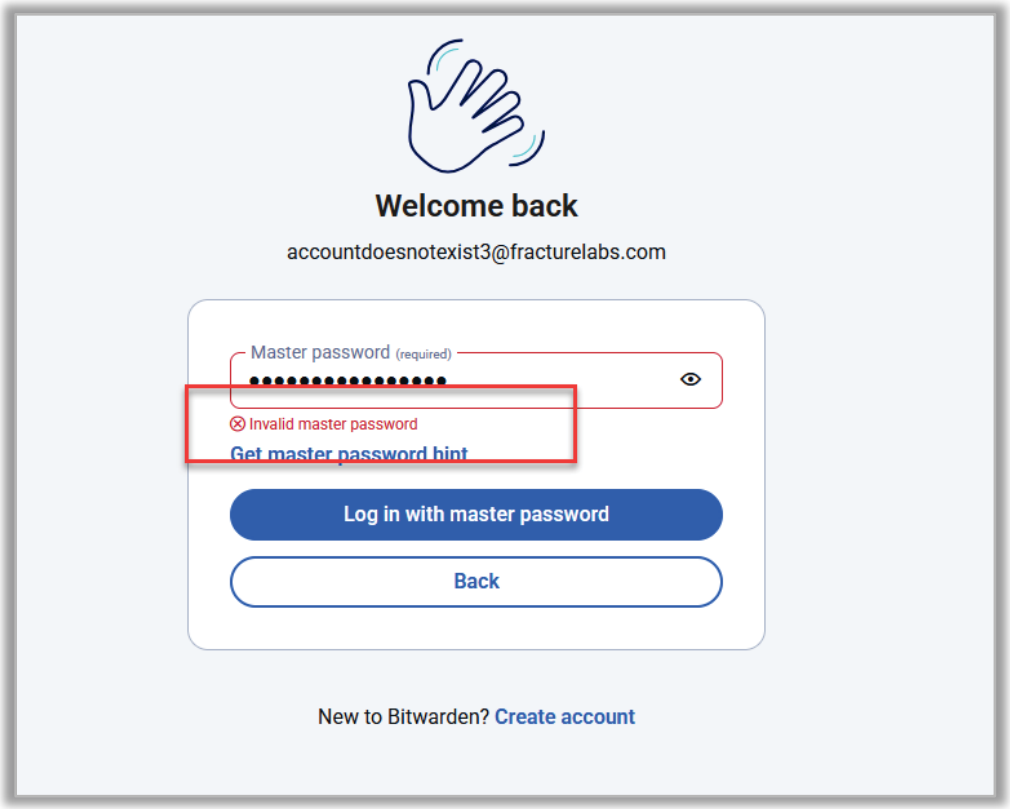
Fracture Labs began with unauthenticated testing of the web applications, focusing on identifying unpatched infrastructure, sensitive endpoints, and weaknesses that could allow unprotected access to application data and functions intended for authenticated users. Fracture Labs analyzed server responses and identified that HTTP response headers disclosed version information for infrastructure components, such as Varnish cache servers (see 2025.WEB.BPI.01 – Disable Software Name and Version Reporting).



While not directly exploitable, this information disclosure could assist threat actors in targeting known vulnerabilities associated with specific software versions.

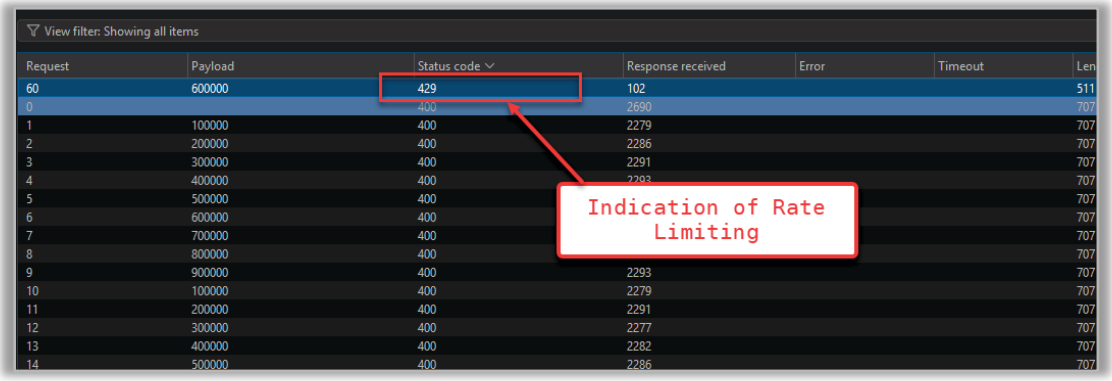
Next, Fracture Labs conducted directory enumeration and file brute-forcing to identify hidden resources, administrative interfaces, and sensitive endpoints. No vulnerabilities or misconfigurations were discovered.

Following this, Fracture Labs transitioned into the authenticated testing portion focusing primarily on `vault.bitwarden.com`. Using multiple test accounts across different organizational tiers, Fracture Labs evaluated authentication mechanisms for common weaknesses including credential stuffing, brute force vulnerabilities, and user enumeration. Fracture Labs discovered that the application demonstrated resilient authentication handling by allowing password entry for non-existing accounts before returning generic failure messages, effectively preventing direct user enumeration attacks.



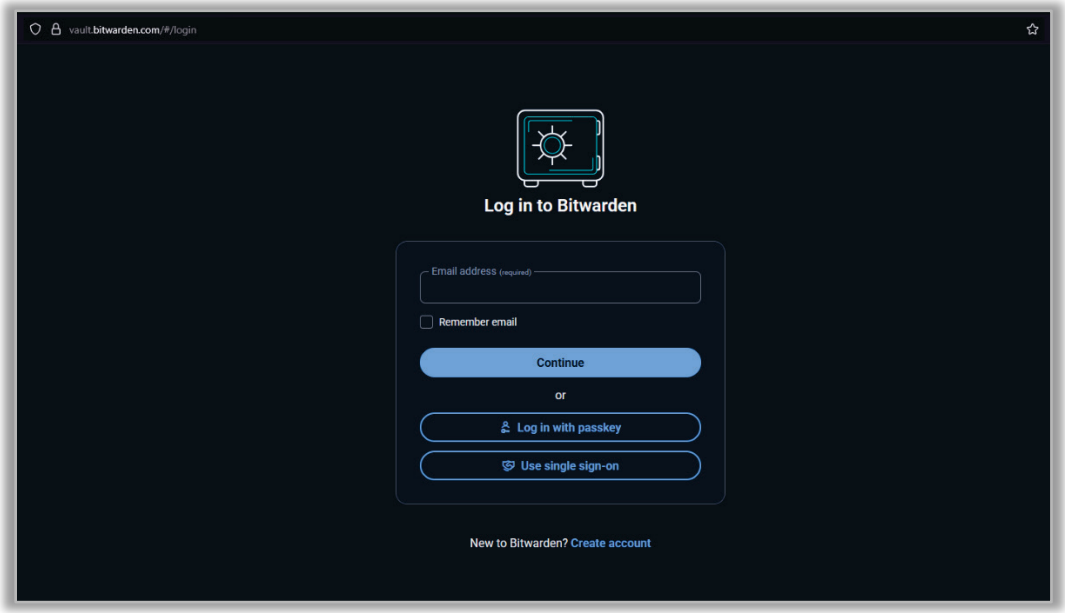
Fracture Labs also confirmed that multi-factor authentication (MFA) was available as an optional security control, appropriately allowing organizations to enforce their preferred security policies rather than mandating universal MFA implementation.

While evaluating the MFA implementation details through source code analysis, Fracture Labs discovered that token validation in both the `EmailTokenProvider` and `OtpTokenProvider` classes compared user-supplied tokens using `string.Equals(token, code)` which could allow threat actors to infer partial matches through repeated probes (see 2025.WEB.L.01 - Non-Time Constant Comparison of Security Token). Fracture Labs tried to leverage timing differences to guess codes but was unable to discern a consistent pattern, and no valid codes were identified. In addition, the application also invoked rate limiting after approximately 60 requests, further limiting the feasibility of the attack.

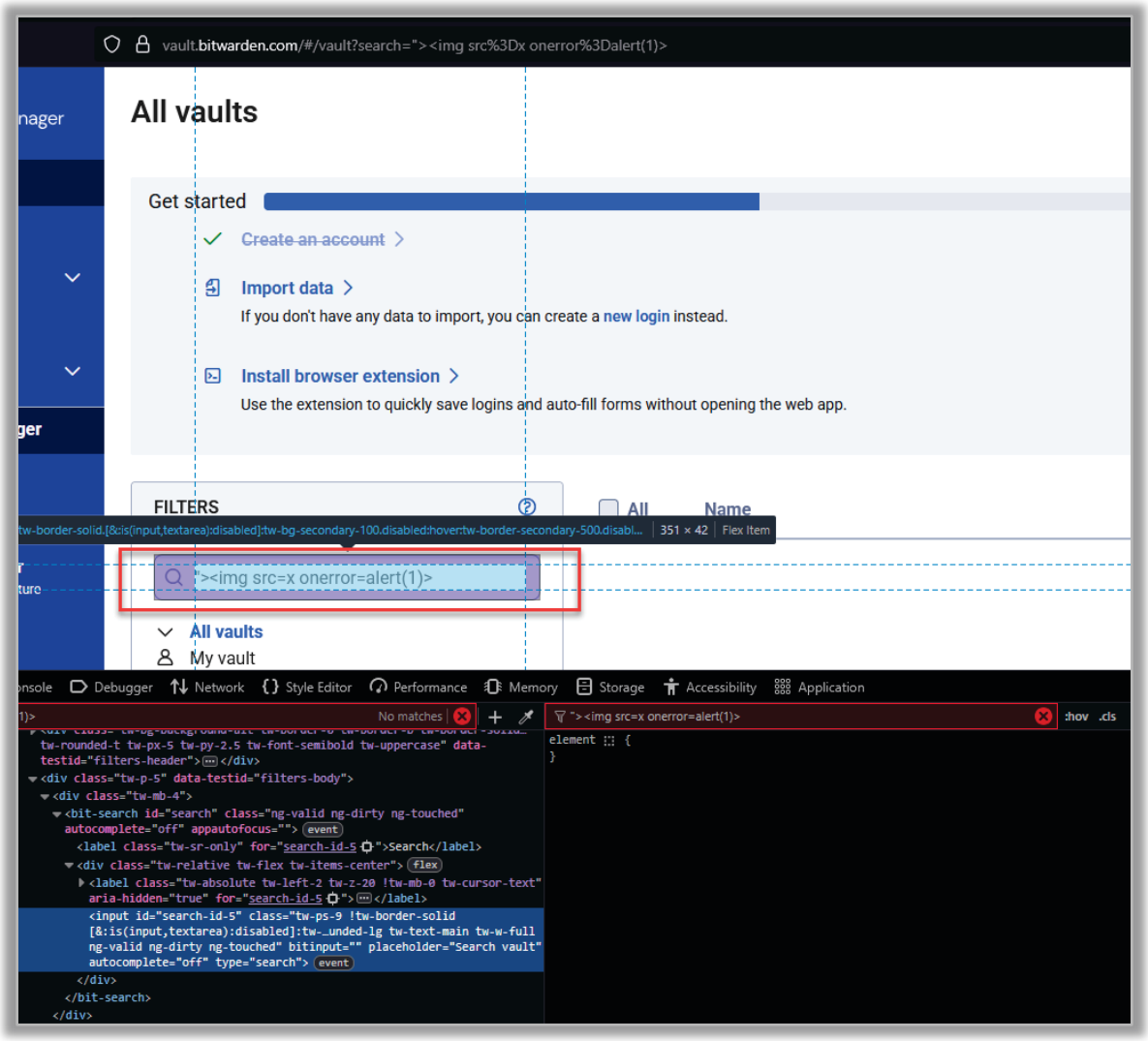


Next, Fracture Labs reviewed session management and found a mixed posture. The application used short-lived bearer tokens (about one hour) instead of traditional session cookies, which is good practice. However, the team also found that the API accepted tokens after a user logged out (see 2025.WEB.L.02 – Existing Session not Invalidated After Manual Logout) , so access continued until the token naturally expired. The one-hour lifetime limited the exposure, but logout did not immediately revoke tokens.

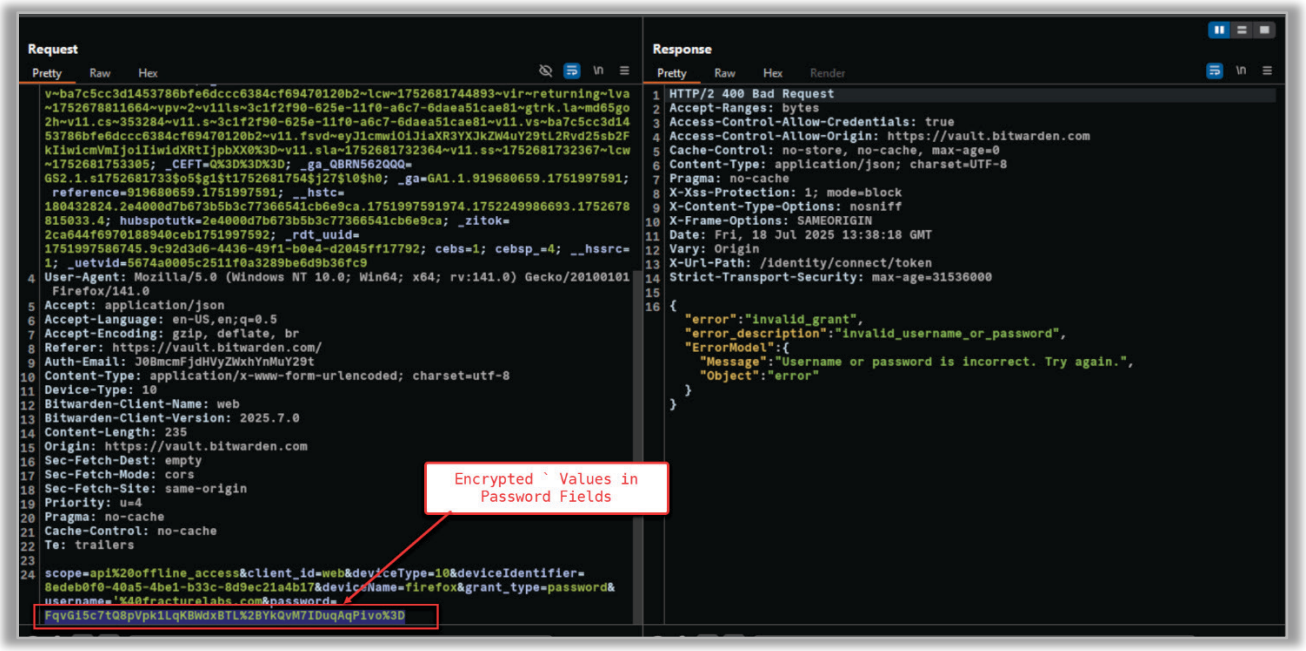
By contrast, the assessment team observed that browser-based session handling behaved as expected. Fracture Labs confirmed that the web interface implemented proper session handling controls. The team noted that upon log out, simply using the browser's back button correctly redirected you to the login page. In addition, direct navigation to authenticated URLs without a valid session was also blocked, proving effective session validation for browser-based access.



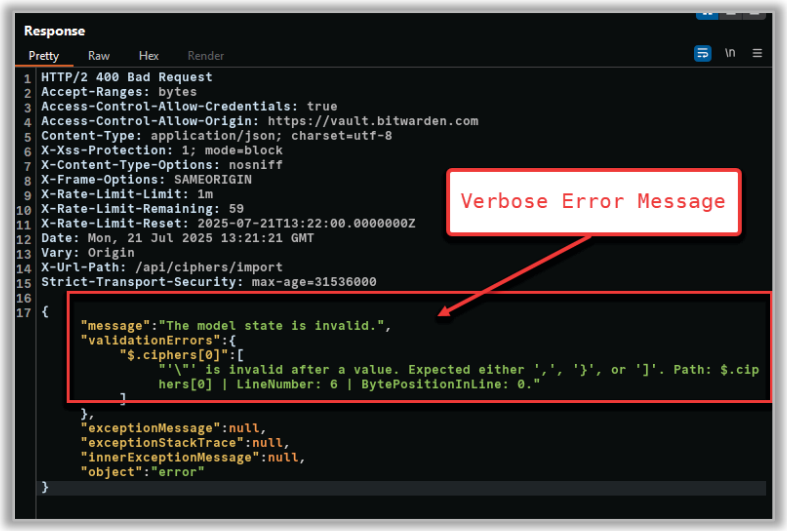
Following the session management evaluation, the assessment team continued to assess injection risks across the application. Fracture Labs conducted comprehensive injection testing across all identified input vectors, including form fields, URL parameters, HTTP headers, and JSON request bodies. As part of the Cross-Site Scripting (XSS) assessment, the team focused testing efforts on the vault item creation functionality and various user input fields. Multiple payloads were submitted to test client-side rendering and sanitization behaviors. All attempts to execute JavaScript were unsuccessful, with the application consistently rendering special characters as plain text rather than executing embedded scripts.



SQL injection testing was also performed to evaluate potential backend query manipulation risks. However, Fracture Labs found that the application's architecture largely eliminated traditional SQL injection vectors. Specifically, search and filter operations were executed entirely on the client side against locally cached vault data that was decrypted in browser memory. As a result, no direct interaction with a server-side database occurred for these features. Additionally, Fracture Labs confirmed that authentication mechanisms were protected through client-side encryption prior to transmission, effectively preventing injection into login-related queries.

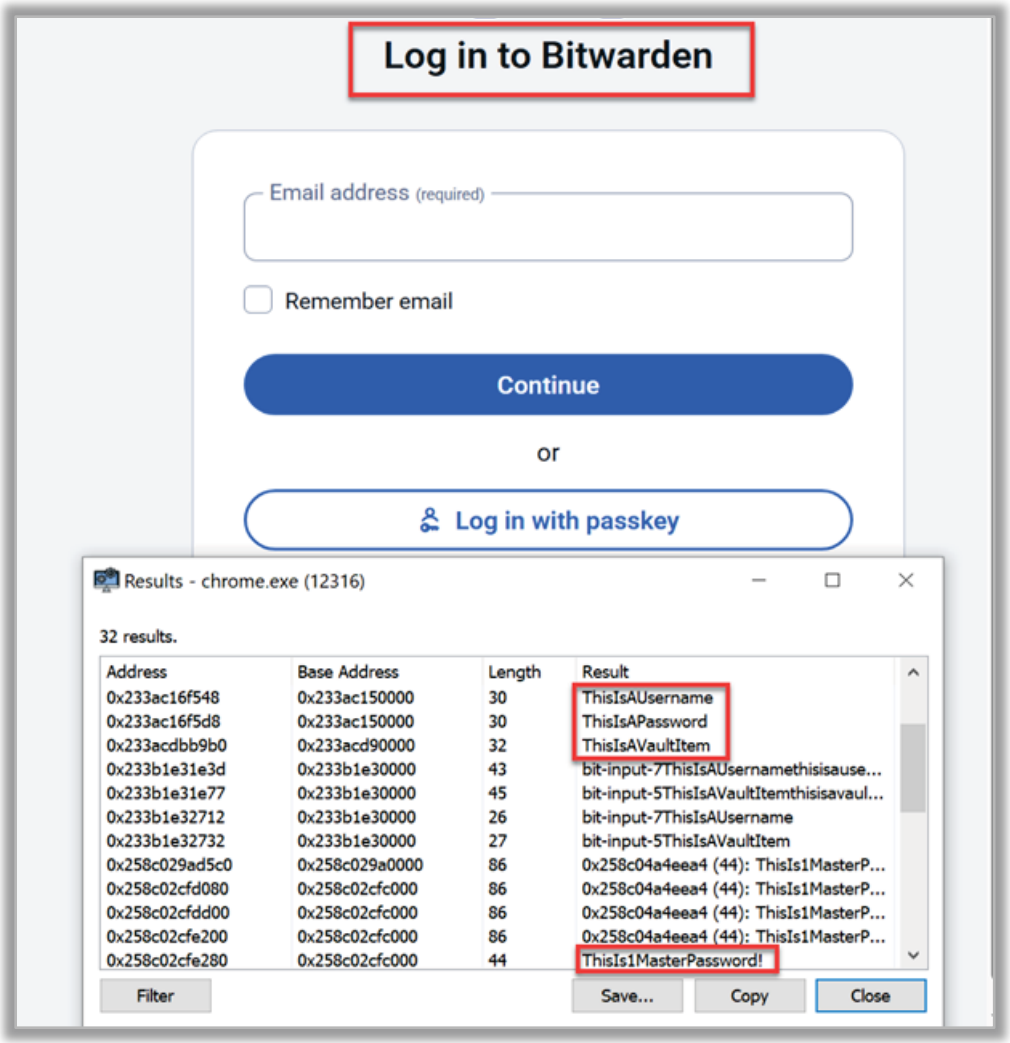


Command injection testing targeted areas such as `cipher import` functionality, where a variety of payloads were submitted to evaluate input handling at the system level. These included basic execution attempts such as `$(id)` and `whoami`, time-based attacks like `sleep(5)`, and template injection strings such as `{{7*7}}` and `${7*7}`. In all cases, the application responded with *HTTP 406 Not Acceptable* or *400 Bad Request* errors, suggesting that invalid or malicious input was properly rejected. However, Fracture Labs identified that malformed JSON submissions to the same endpoint returned verbose error messages that disclosed internal parsing logic and technical architecture details (see 2025.WEB.BPI.02 – Disable Verbose Error Messages). While these messages did not directly expose exploitable behavior, they could aid threat actors in constructing targeted payloads.



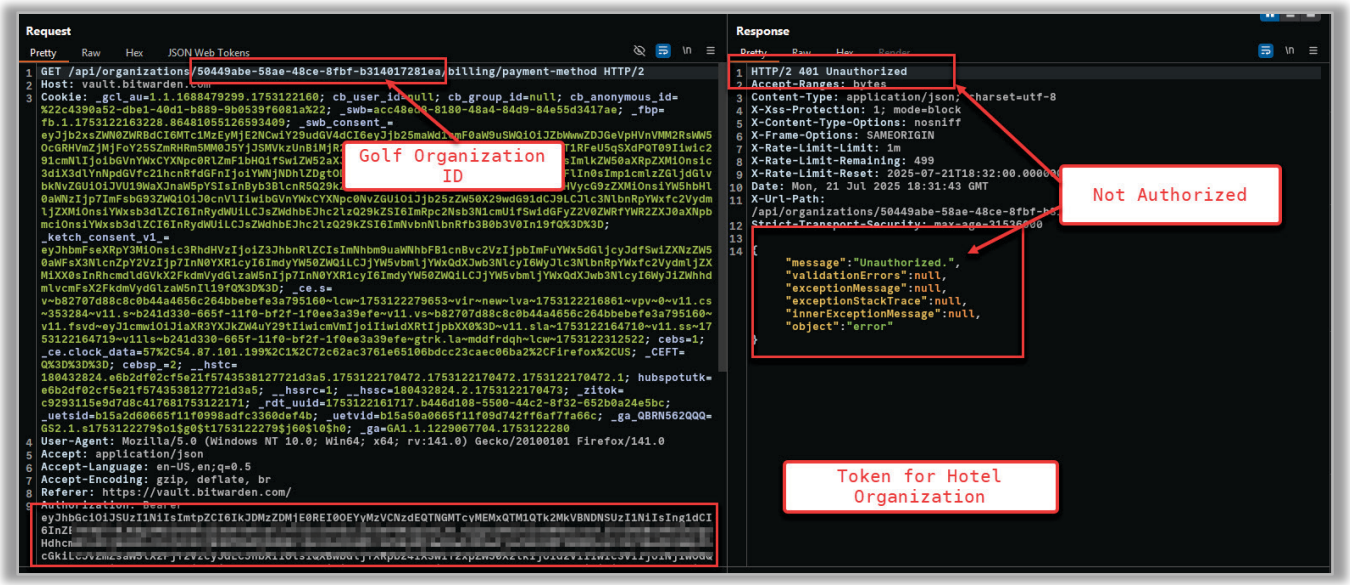
Fracture Labs also assessed the application's resilience to protocol-level injection by trying host header manipulation. These attempts were unsuccessful and consistently returned *HTTP 421 Misdirected Request* responses, indicating proper validation of host headers and resistance to potential cache poisoning or routing attacks.

To evaluate the application's memory handling behavior, Fracture Labs conducted a security analysis of how sensitive data was managed within browser memory during and after vault interactions. The team authenticated into the application, accessed vaults, and manually locked them using the "Lock now" feature before analyzing browser memory contents. Fracture Labs discovered that even after locking or logging out, decrypted vault contents and master passwords remained present in plaintext within browser process memory (see 2025.WEB.M.01 – Cleartext Storage of Sensitive Information in Memory).



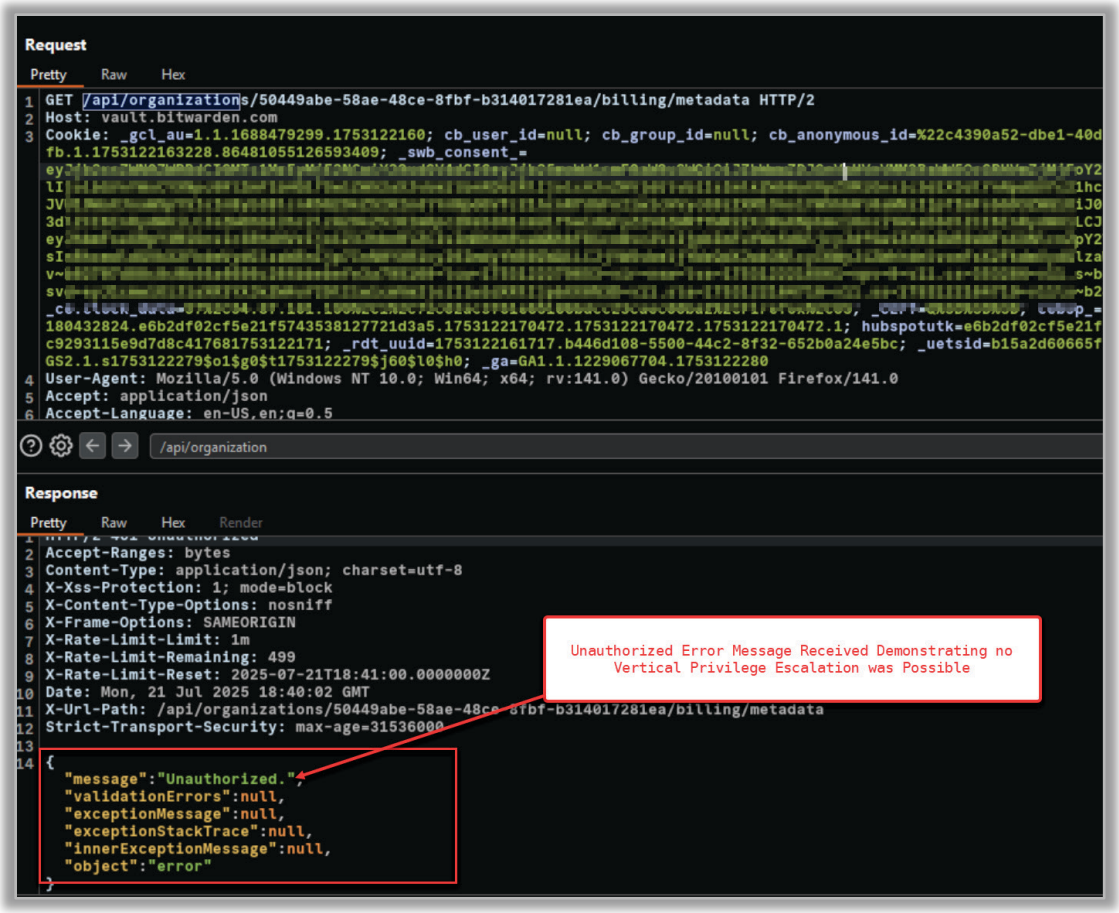
Threat actors with access to the underlying host system could extract sensitive data from memory, effectively bypassing the vault locking mechanism that users may believe provides immediate protection. This finding was carried over from the previous year's assessment as Bitwarden's fixes did not remediate the issue.

Following this, Fracture Labs conducted access control testing using accounts with varying privilege levels across different organizations. Fracture Labs focused on uncovering both horizontal and vertical privilege escalation opportunities through API request tampering and parameter manipulation. Fracture Labs tried to access organizational billing information by modifying the organization ID in an authenticated API request. Fracture Labs used a valid session token associated with the `Hotel` organization to craft a request targeting the billing information of the `Golf` organization, as indicated by the modified organization ID in the request URL.

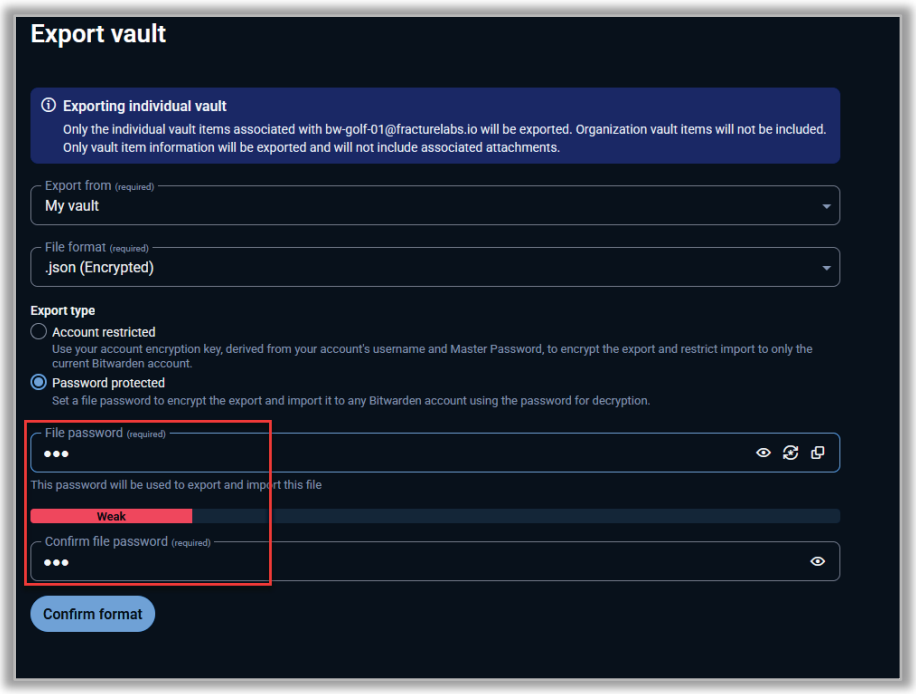


The server responded with a `401 Unauthorized` status code, indicating that access to the targeted resource was correctly denied. This behavior confirmed that the application enforced proper authorization checks at the object level, effectively preventing horizontal privilege escalation and mitigating risks associated with insecure direct object references (IDOR).

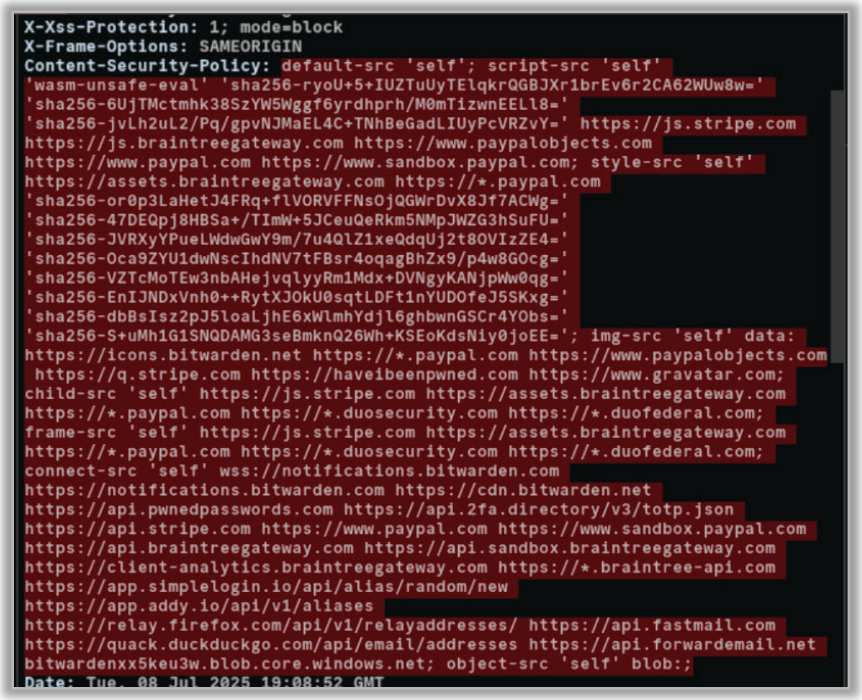
Fracture Labs also conducted Role-Based Access Control (RBAC) verification and found that regular user accounts were properly restricted from accessing administrative functions including group management, member administration, and organizational settings. When Fracture Labs tried to access restricted administrative endpoints, the requests returned proper "*Unauthorized*" or "*Resource Not Found*" responses, with no successful privilege escalation paths found through role manipulation or request parameter modification.



As previously identified, Fracture Labs observed that the encrypted vault export feature continued to allow the use of extremely weak passwords, including single-character entries. While the application was updated to include a password strength indicator since the previous assessment, there were still no enforced minimum complexity requirements. As a result, users may inadvertently choose insecure passwords for exported vault data, which could reduce the overall protection of sensitive information in certain scenarios.

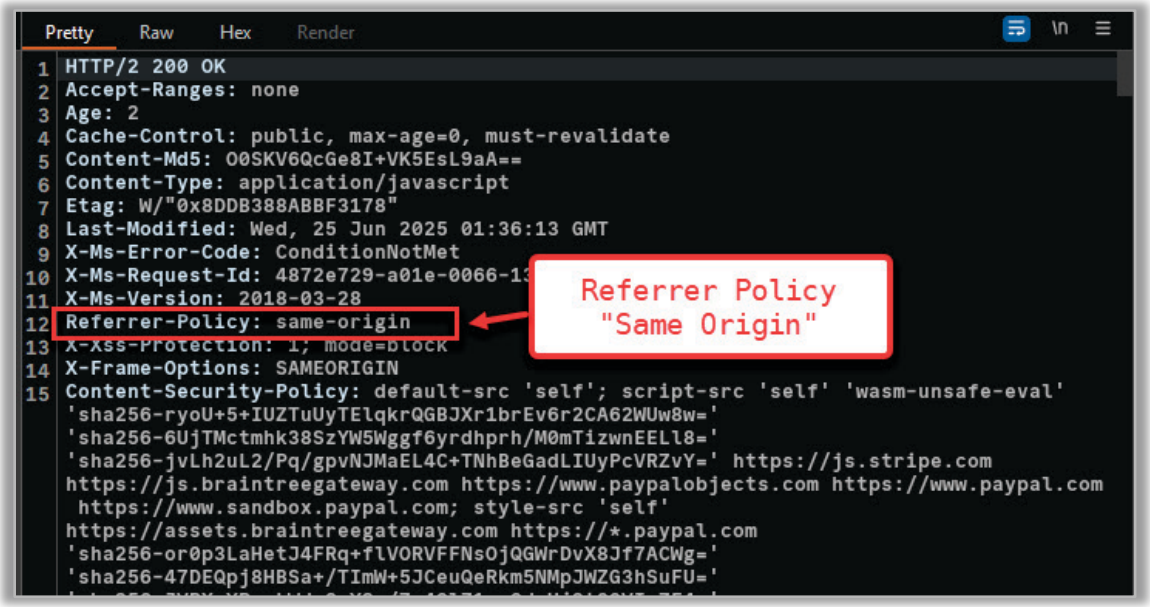


To evaluate client-side protection, Fracture Labs performed security header analysis to assess browser-side protections against common client-side attack vectors. Fracture Labs confirmed that a strict Content Security Policy was in place, limiting executable scripts to trusted sources.

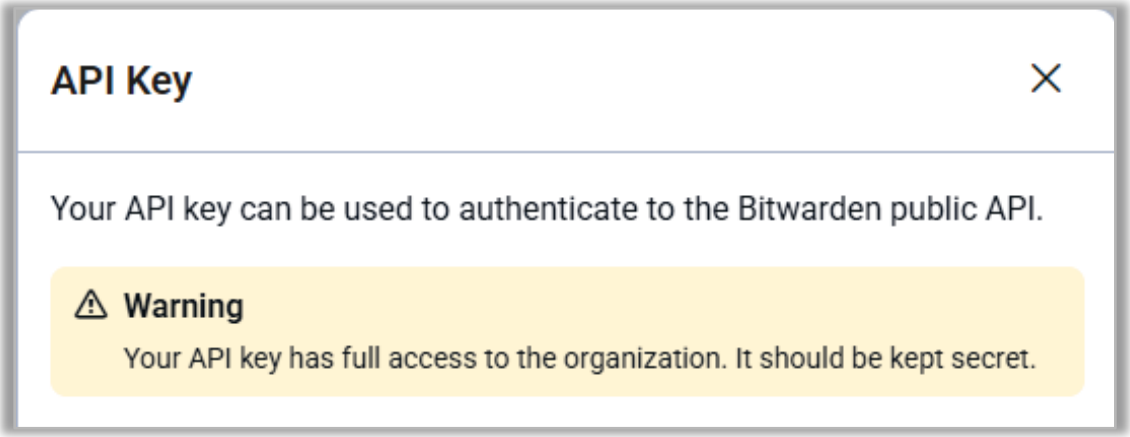


`X-Frame-Options` headers were consistently applied, mitigating clickjacking risks, while the presence of `X-XSS-Protection` and `Strict-Transport-Security` headers further reinforced secure browser behaviors.

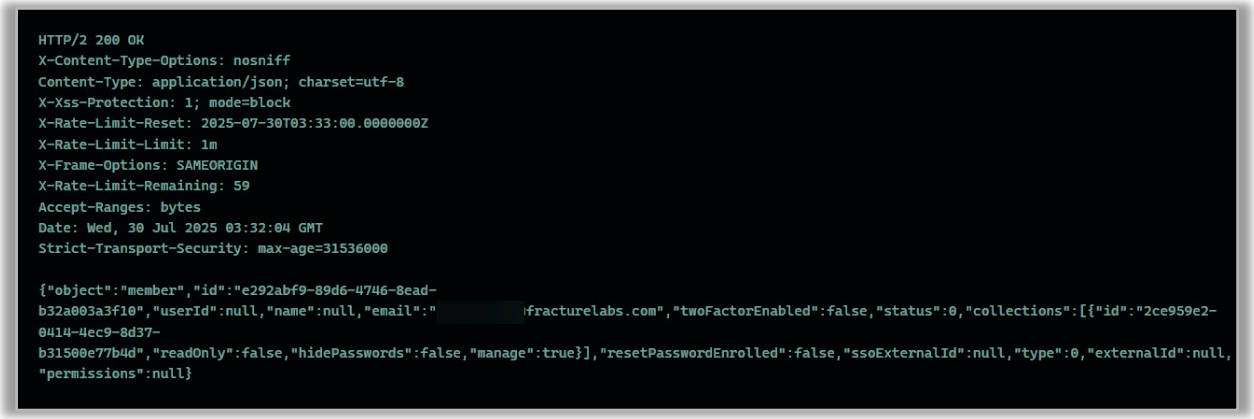
The application also implemented the `Referrer-Policy` header with a value of "`same-origin`," which provided excellent protection of referral data.



Fracture Labs also evaluated the security of Bitwarden's integration and API architecture, particularly the management and scoping of API keys. Fracture Labs discovered that organizations were issued a single API key with broad access privileges instead of separate keys tied to specific functions or access scopes (see 2025.WEB.BPI.03 – Implement Fine-Grained API Scope Assignments).



This design choice increases the risk associated with credential leakage, as a compromised API key could provide far more access than a narrowly scoped integration key would permit. For example, the API key was used to create a user even though the key was intended for other purposes.



Lastly, Fracture Labs tested file upload capabilities across the platform and confirmed that the application enforced strict controls. Uploaded files were validated for appropriate extensions, and unauthorized formats such as potentially dangerous SVG files were correctly rejected.

HIGH-RISK FINDINGS

No High-Risk findings were identified.

MEDIUM-RISK FINDINGS

2025.WEB.M.01 – CLEARTEXT STORAGE OF SENSITIVE INFORMATION IN MEMORY

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

RISK SUMMARY

Fracture Labs retested a previously identified vulnerability where vault passwords and the master password were stored in cleartext within browser memory and persisted even after users logged out of the application. The original finding demonstrated that threat actors with access to the underlying host system could extract sensitive vault credentials from memory dumps, effectively bypassing logout protections across all tested browsers.

Furthermore, the master password and vault items remained in browser memory in Chrome, Edge, and Firefox even after closing the window that once held the Bitwarden Vault tab as long as at least one other window remained open.

Although Bitwarden implemented changes to how the master password was stored in memory, Fracture Labs observed no improvements and was able to recreate the previous results.

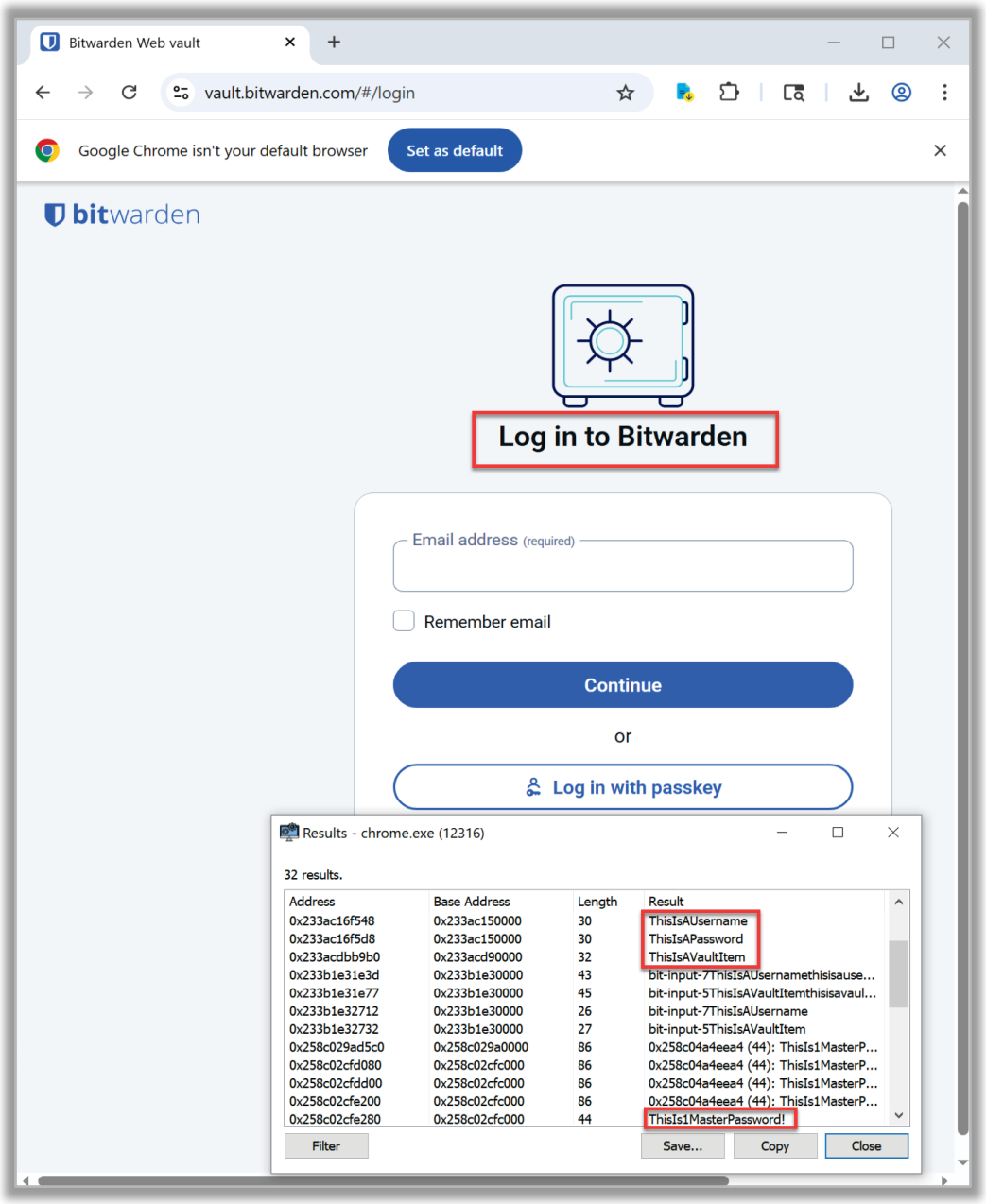
NOTE: While Bitwarden may be able to reduce the likelihood of this through application changes and driving user behavior with updated documentation, the core issue remains with how browsers manage memory. Fracture Labs manually verified this same weakness existed in other password vault providers.

AFFECTED ASSETS

- https://vault.bitwarden.com

TECHNICAL DETAILS

Fracture Labs logged into the password vault using a valid master password and then manually locked it using the application's “Lock now” feature. Next, Fracture Labs searched the browser process memory for the master password and vault items, and found everything remained in cleartext in memory. The same held true even after completing logging out of the vault.



REMEDIATION RECOMMENDATIONS

Because modern browsers reuse long-lived renderer/content processes and typical memory allocators don't wipe buffers on free, sensitive values (e.g., a vault master password) can persist in the same process after "lock" or "logout." A simple page refresh or window reload may not be sufficient, so the only reliable boundary is process termination.

Therefore, the application and documentation should make this limitation clear to users, noting that all browser-based password managers have inherent memory persistence risks that desktop applications can better control. The logout/lock flow should instruct users to fully exit the browser, so the renderer process is torn down and its address space is re-initialized for greater assurance of data confidentiality.

Additionally, Bitwarden could reduce in-process exposure by adopting the Web Cryptography API to reduce the likelihood of a successful attack:

- Avoid immutable JavaScript strings for secrets.
- Read sensitive data once, convert to a Uint8Array, and immediately derive a non-extractable `CryptoKey` with `crypto.subtle` (e.g., PBKDF2→AES-GCM) inside a dedicated Web Worker.
- Transfer the byte buffer to the worker using the transfer list (to avoid cloning), then zeroize it and remove the input element from the DOM.
- Keep keys non-extractable, perform decrypt-on-demand only at reveal time, render plaintext to the UI for the shortest possible interval, and then remove the node and zeroize any working buffers.

ADDITIONAL RESOURCES

CWE-316: Cleartext Storage of Sensitive Information in Memory
https://cwe.mitre.org/data/definitions/316.html

Web Crypto API
https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API

Bitwarden: Is My Bitwarden Master Password Stored Locally
https://bitwarden.com/help/security-faqs/#q-is-my-bitwarden-master-password-stored-locally

2025.WEB.M.02 – AUDIT/EVENT LOG TAMPERING			
	Low	Medium	High
Damage Potential	Trivial Info	***Sensitive Info, Defacement***	Admin, Full Trust
Reproducibility	Difficult to Reproduce	Needs Special Circumstances	***Easily Reproduced***
Exploitability	Expert, Advanced Skills	***Skilled***	Novice
Affected Users	***Few Users***	Some Users	All Users
Discoverability	Obscure	***Limited***	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/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. For example, the following message was dropped intentionally by the proxy so the audit message did not appear in the event log.

Request

Pretty Raw Hex

```
POST /events/collect HTTP/2
Host: vault.bitwarden.com
Cookie: _gcl_au=1.1.1688479299.1753122160; cb_user_id=null; cb_group_id=null;
cb_anonymous_id=%22c4390a52-dbe1-40d1-b889-9b0539f6081a%22; _swb=
acc48ed8-8180-48a4-84d9-84e55d3417ae; _fbp=fb.1.1753122163228.86481055126593409;
_ketch_consent_v1_=
eyJhbmFseXRpY3MiOnsic3RhdHVzIjoiZ3JhbnRlZCIsImNhbm9uaWNhbFB1cnBvc2VzIjpbImFuYWx5dGljcyJdfSwi
ZXNzZW50aWFsX3NlcnZpY2VzIjp7InN0YXR1cyI6ImdyYW50ZWQiLCJjYW5vbmljYWxQdXJwb3NlcyI6WyJlc3NlbnRp
YWxfc2VydmljZXMiXX0sInRhcmdldGVkX2FkdmVydGlzaW5nIjp7InN0YXR1cyI6ImdyYW50ZWQiLCJjYW5vbmljYWxQ
dXJwb3NlcyI6WyJiZWhhdmlvcmFsX2FkdmVydGlzaW5nIl19fQ%3D%3D; _ce.s=
v~b82707d88c8c0b44a4656c264bbebefe3a795160~lcw~1753122279653~vir~new~lva~1753122216861~vpv~0
~v11.cs~353284~v11.s~b241d330-665f-11f0-bf2f-1f0ee3a39efe~v11.vs~b82707d88c8c0b44a4656c264bb
ebefe3a795160~v11.fsvd~eyJ1cmwiOiJiaXR3YXJkZW4uY29tIiwicmVmIjoiIiwidXRtIjpbXX0%3D~v11.sla~17
53122164710~v11.ss~1753122164719~v11ls~b241d330-665f-11f0-bf2f-1f0ee3a39efe~gtrk.la~mddfrdqh
~lcw~1753122312522; _CEFT=Q%3D%3D%3D; __hstc=
180432824.e6b2df02cf5e21f5743538127721d3a5.1753122170472.1753122170472.1753122170472.1;
hubspotutk=e6b2df02cf5e21f5743538127721d3a5; _zitok=c9293115e9d7d8c41768175312217l;
_rdt_uuid=1753122161717.b446d108-5500-44c2-8f32-652b0a24e5bc; _uetvid=
b15a50a0665f11f09d742ff6af7fa66c; _ga_QBRN562QQQ=
GS2.1.s1753122279$o1$g0$t1753122279$j60$l0$h0; _ga=GA1.1.1229067704.1753122280
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:142.0) Gecko/20100101 Firefox/142.0
Accept: */*

[
    {
        "type":1107,
        "cipherId":"5df06be9-af35-4e15-bce0-b3410123490c",
        "date":"2025-07-22T11:41:34.821Z",
        "organizationId":"50449abe-58ae-48ce-8fbf-b314017281ea"
    }
]
```

Search 0 highlights

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

From 07/14/2025, 12:00 AM – To 07/23/2025, 11:59 PM Update Export

Timestamp	Client	Member	Event
Jul 22, 2025, 7:41:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 22, 2025, 7:41:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 22, 2025, 7:41:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 22, 2025, 7:41:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 22, 2025, 7:41:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 22, 2025, 7:41:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 22, 2025, 7:40:34 AM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Viewed item 5df06be9.
Jul 21, 2025, 3:58:55 PM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Logged in
Jul 21, 2025, 3:58:22 PM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Login attempt failed with incorrect password.
Jul 21, 2025, 3:50:27 PM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Invited user d9aacc0a.
Jul 21, 2025, 2:50:11 PM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Edited group d46251ee.
Jul 21, 2025, 2:48:52 PM	Web vault - Firefox	bw-golf-01@fracturelabs.io	Created group d46251ee.

REMEDIATION RECOMMENDATIONS

Bitwarden updated the documentation but did not address the unreliability of the event log data. Bitwarden should consider advising users that the event report relies upon user-reported data and may not be suitable for security, 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

2025.WEB.L.01 – NON-TIME CONSTANT COMPARISON OF SECURITY TOKEN [REMEDIATED]

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

RISK SUMMARY

Fracture Labs discovered that security-sensitive operations used non-time constant comparisons for token validation, creating potential timing attack vulnerabilities. Source code analysis revealed that token verification functions used standard string comparison methods instead of cryptographically secure constant-time comparison functions.

Fracture Labs conducted timing analysis of email verification codes but found that statistical averaging across multiple attempts showed no significant exploitable patterns.

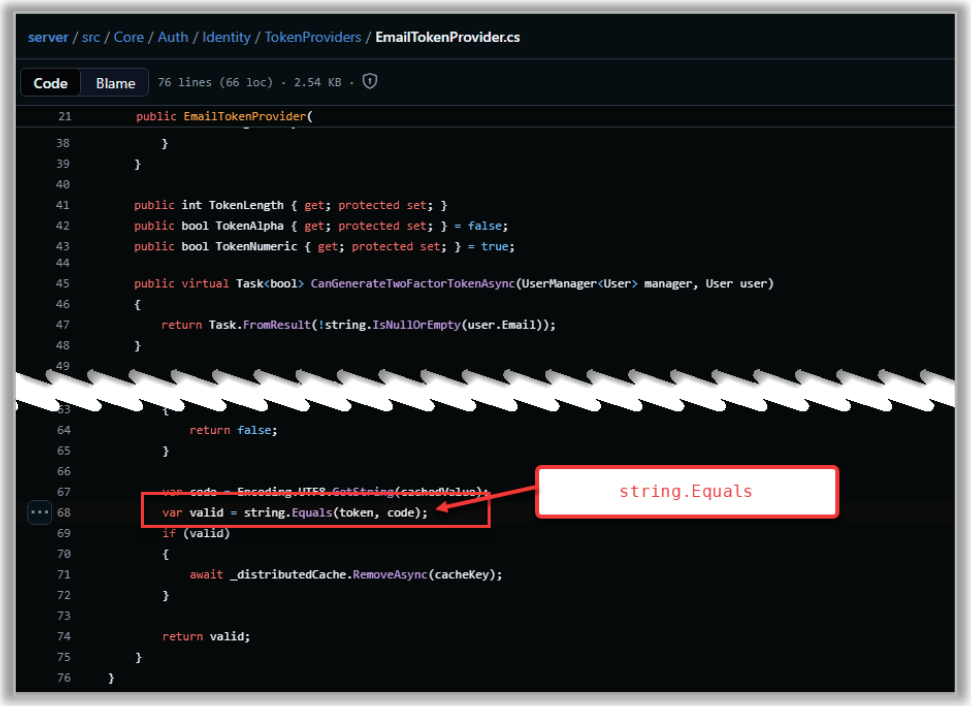
While IP rate limiting protections and inconsistent timing signals limited the practical exploitability of this theoretical attack, the underlying implementation represented a cryptographic best practice deviation that should be addressed to prevent future abuse should controls or timing conditions change.

AFFECTED ASSETS

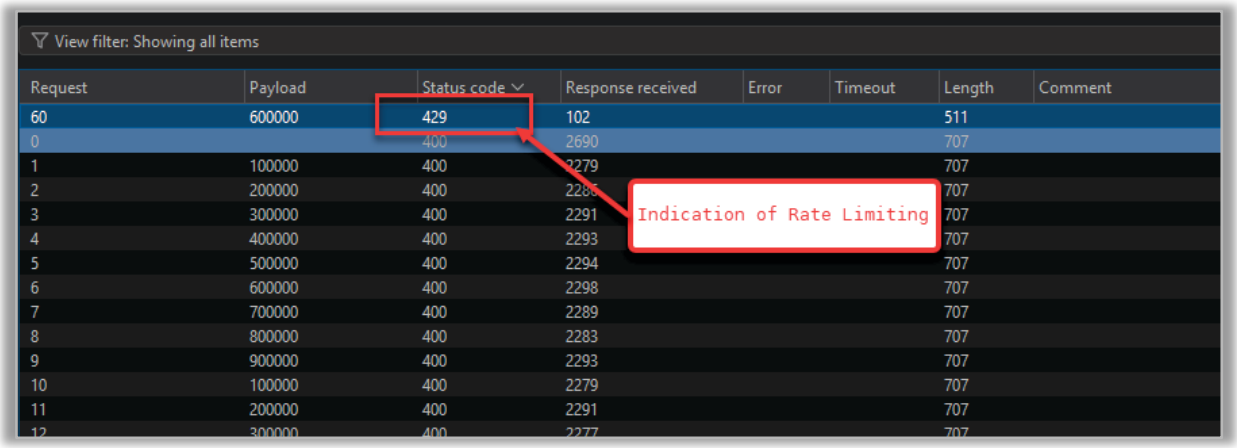
- https://github.com/bitwarden/server/blob/30300bc59beae15d615dd12e3f4e6d3d3e1514bb/src/Core/Auth/Identity/TokenProviders/EmailTokenProvider.cs#L68
- https://github.com/bitwarden/server/blob/30300bc59beae15d615dd12e3f4e6d3d3e1514bb/src/Core/Auth/Identity/TokenProviders/OtpTokenProvider/OtpTokenProvider.cs#L67

TECHNICAL DETAILS

Fracture Labs performed source code analysis and discovered that token validation functions used standard string comparison methods that could theoretically create timing variation issues. In particular, the `EmailTokenProvider` and `OtpTokenProvider` classes used `string.Equals()` for token comparison, which performs character-by-character comparison and could potentially create timing variations based on the position of the first mismatched character differences.



Fracture Labs conducted statistical timing analysis against the email verification system; however, the results showed no significant exploitable timing patterns.



The timing analysis demonstrated that while theoretical timing vulnerabilities existed in the code, practical exploitation was prevented by network latency, server-side processing variations, and rate limiting controls.

REMEDIATION RECOMMENDATIONS

Bitwarden should consider implementing constant-time comparison functions for all security-sensitive token validation operations as a cryptographic best practice. The `CoreHelpers` class already contains a `FixedTimeEquals` function that should be used instead of `string.Equals()` for token comparison.

This change would eliminate any theoretical timing attack vectors and align with cryptographic security best practices.

REMEDIATION VERIFICATION RESULTS

Fracture Labs reviewed pull request #6279 on September 3, 2025 and verified the code change would resolve the issue.

ADDITIONAL RESOURCES

Observable Time Discrepancy

https://cwe.mitre.org/data/definitions/208.html

2025.WEB.L.02 – EXISTING SESSION NOT INVALIDATED AFTER MANUAL LOGOUT

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

RISK SUMMARY

Fracture Labs identified that the application did not properly invalidate user sessions when users performed manual logout operations. The session tokens remained valid and functional even after users clicked the logout button and were redirected to the login page.

Threat actors who obtain access to session tokens through methods such as cross-site scripting, session hijacking, or physical device access could continue to authenticate and access user accounts despite the legitimate user's logout attempt.

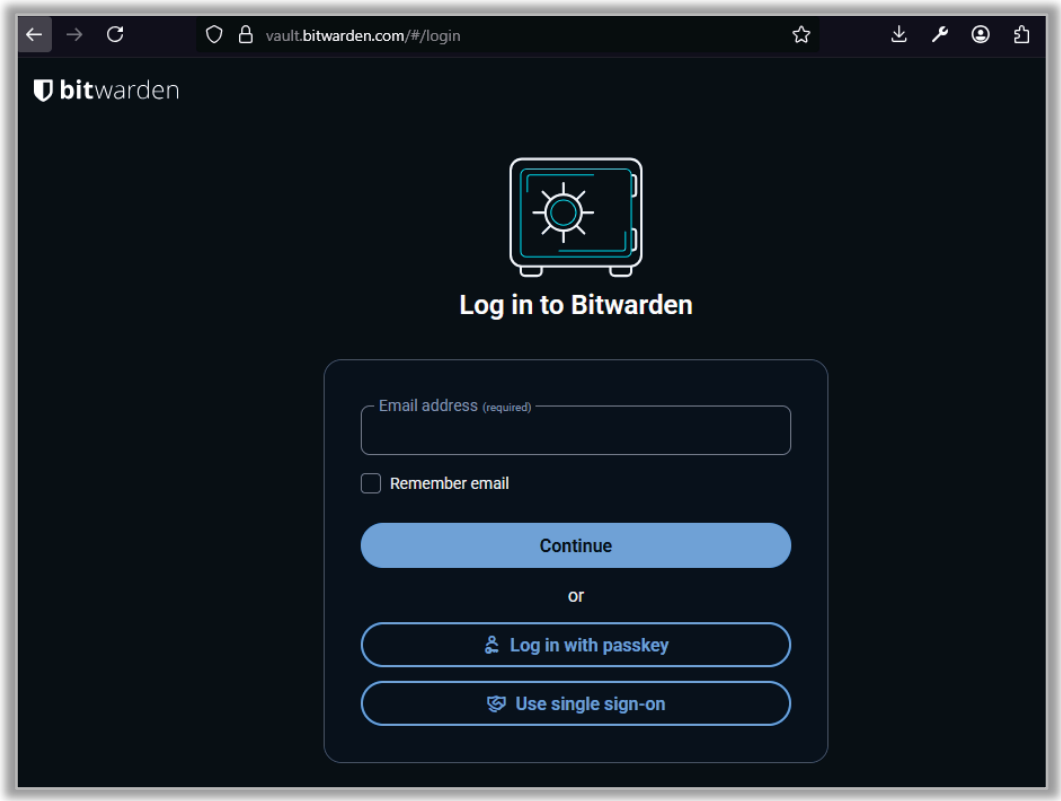
AFFECTED ASSETS

- https://vault.bitwarden.com

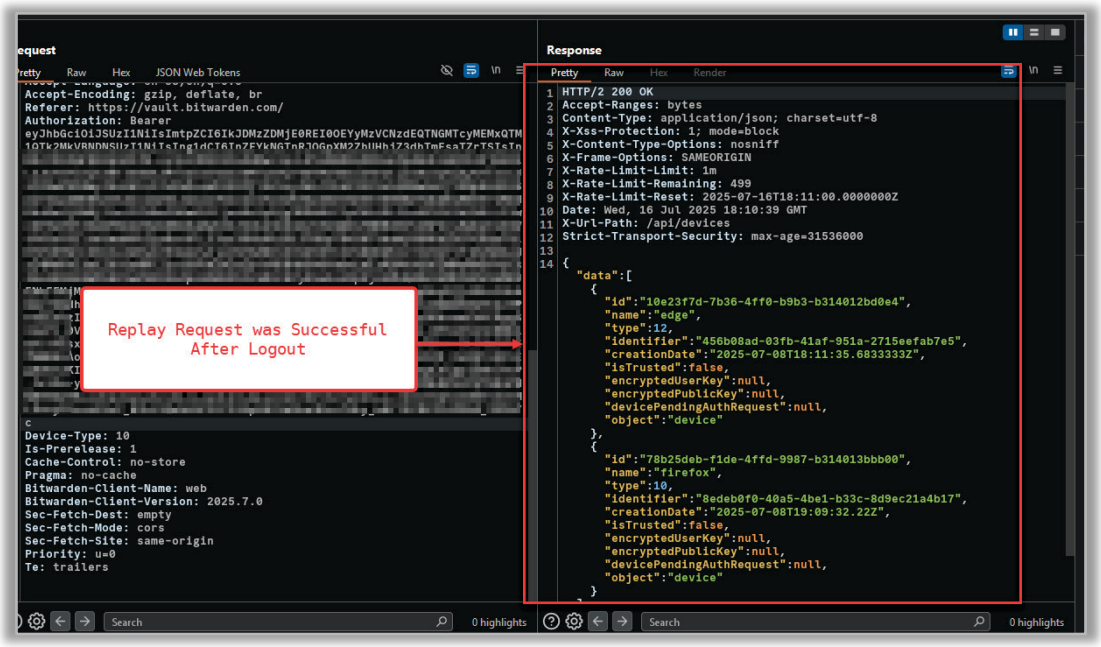
TECHNICAL DETAILS

Fracture Labs authenticated to the application and captured an API request to `/api/devices` that included a valid bearer token in the Authorization header.

Next, Fracture Labs initiated a logout action by clicking the “Log Out” button in the menu which redirected the browser to the login page.



However, when the same `/api/devices` request was replayed using the original bearer token, the server returned a `200 OK` response which included the vault data and information.



REMEDIATION RECOMMENDATIONS

Bitwarden should consider implementing server-side token revocation during the logout process. The logout endpoint should maintain a short-lived token revocation list keyed off the JWT's `jti` value until expiration that API endpoints check before accepting tokens. While one-hour lifespans are already relatively short, Bitwarden should consider implementing even shorter token lifespans (5-10 minutes) with automatic refresh mechanisms for active sessions. For enhanced security, Bitwarden could consider tracking active tokens per user and providing users visibility into their active sessions with the ability to revoke specific tokens.

ADDITIONAL RESOURCES

OWASP - JSON Web Token for Java: No Built-In Token Revocation by the User

https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html#no-built-in-token-revocation-by-the-user

BEST-PRACTICE IMPROVEMENT OPPORTUNITIES

2025.WEB.BPI.01 – DISABLE SOFTWARE NAME AND VERSION REPORTING

RISK SUMMARY

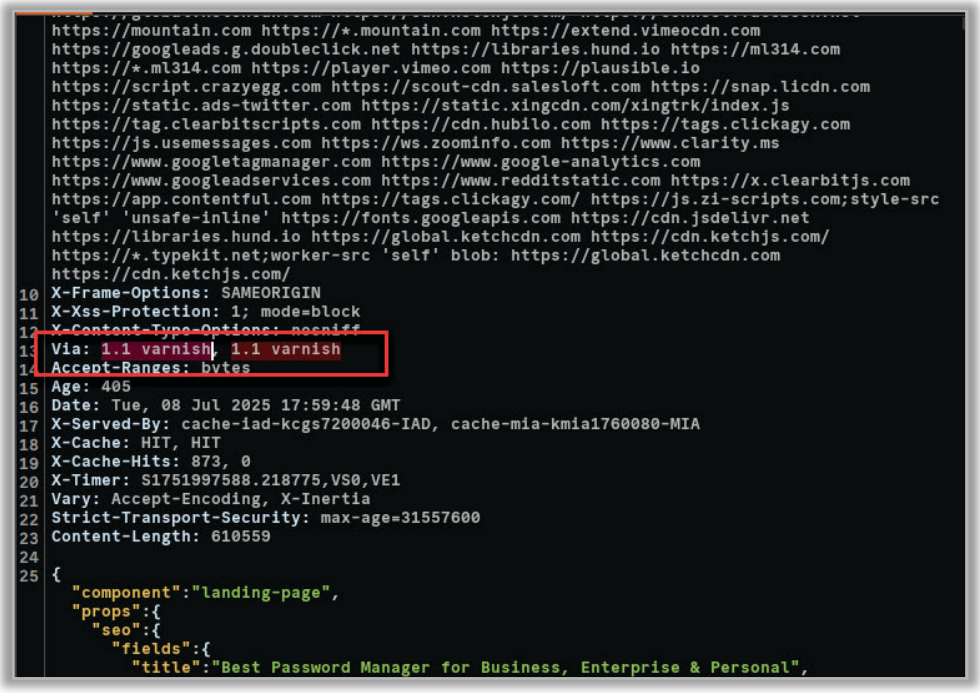
Fracture Labs observed an instance of information disclosures via HTTP headers that provided insight into software names and versions. A threat actor could leverage this information disclosure vulnerability to launch targeted attacks against the application. The header values could also be logged in 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

- The list of affected assets has been provided to Bitwarden separately

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.



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

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

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

2025.WEB.BPI.02 – DISABLE VERBOSE ERROR MESSAGES

RISK SUMMARY

Fracture Labs found an information disclosure vulnerability in Bitwarden's `cipher import` endpoint that exposed internal JSON parser details and validation logic through verbose error messages.

Information disclosure vulnerabilities occur when applications reveal internal system details, error handling mechanisms, or processing logic that could aid threat actors in reconnaissance activities. While not directly exploitable for system compromise, these vulnerabilities provide valuable intelligence that can inform more sophisticated attack strategies.

Fracture Labs discovered that the `cipher import` endpoint processed user-submitted password data during vault imports from other password managers. When malformed JSON data was sent, the endpoint returned detailed error messages containing internal parser state information, including specific line numbers, byte positions, and validation logic details. This information disclosure could help threat actors understand the backend architecture and identify potential attack vectors.

AFFECTED ASSETS

- https://vault.bitwarden.com/api/ciphers/import

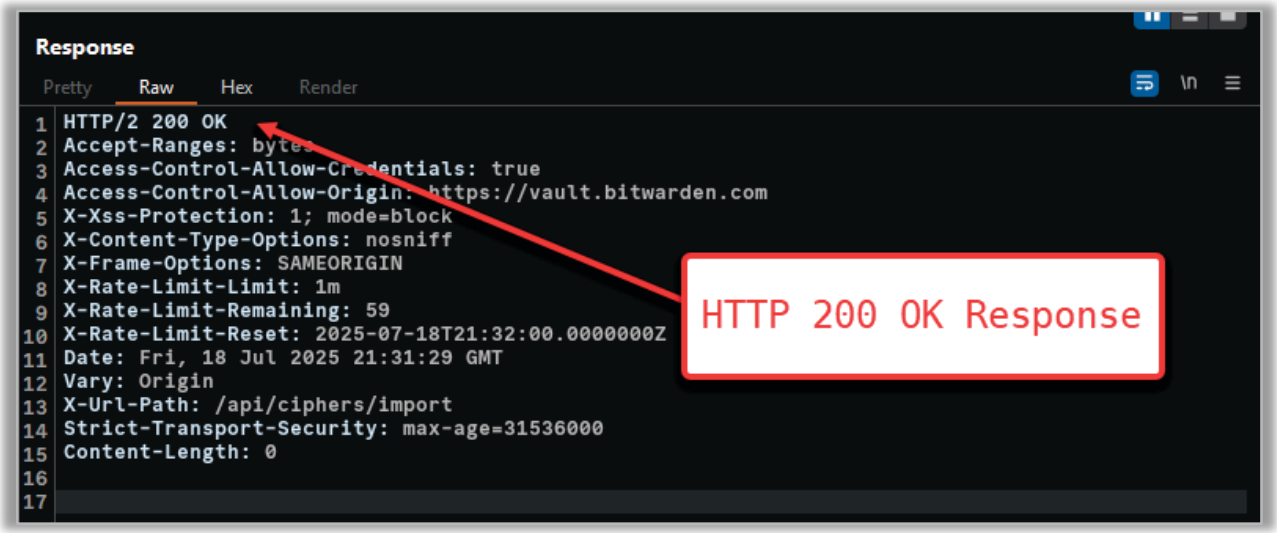
TECHNICAL DETAILS

The `cipher import` endpoint allowed users to import password data that was exported from other password managers into their Bitwarden vault. Fracture Labs analyzed the available Bitwarden API endpoints for injection weaknesses by submitting malformed payloads to the `cipher import` endpoint and examining error responses for verbose debugging details. Fracture Labs then analyzed the collected error responses for injection success indicators and identified verbose error messages that revealed internal parser states and system architecture details.

To establish baseline behavior, Fracture Labs sent the following standard `cipher import` request:

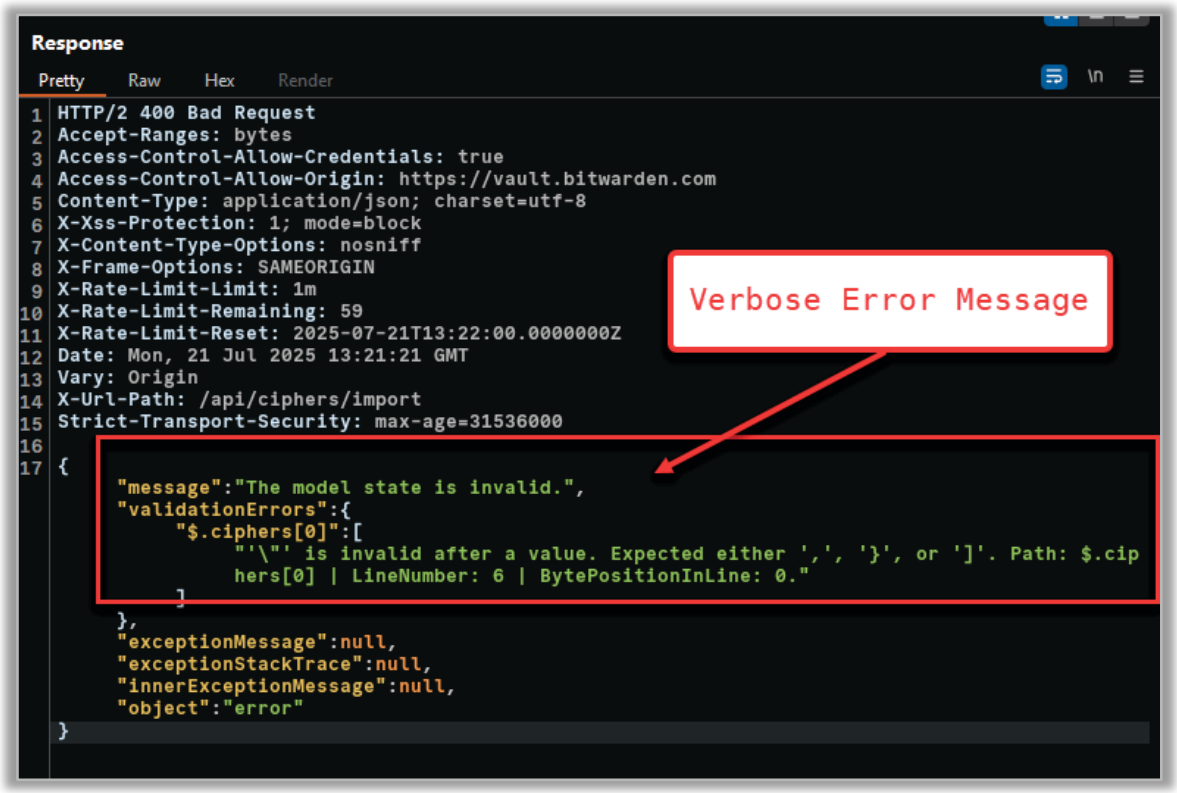
```
POST /api/ciphers/import HTTP/2
  Host: vault.bitwarden.com
  Authorization: Bearer [token]
  Content-Type: application/json; charset=utf-8

  "ciphers":[{"type":1,"encryptedFor":"c7b552de-7eda-47f0-a21f-
b314012bcecc","folderId":null,"organizationId":null,"name":"2.ypj6HSQJN2K2YBIgomQhcQ==|NrS3qgcnwt9zENLBSLxGrVIXSJbfNf6YkV/IcaZWIwo=|rAcgjFOrmilzrqN
8sEVN3L9eHiP4C72/8Pzq76Yx4XY=","notes":null,"favorite":false,"lastKnownRevisionDate":null,"reprompt":0,"login":{"response":null,"uris":
[{"response":null,"match":null,"uri":"2.WVVpdb7nsbubYDdWQOFzvg==|DeGNKFQMXNZpkRgyDfetXqcWCF3KTPQ3uGNAjhpVc+w=|lb4Hw7tBamGfP5kLIZgguY+y1wOBKhaYixS3u
rn7jWM=","uriChecksum":"2.sIJKJ91V2hw3Tui46e3Y+A==|sk9wUiZmkfa5u4UVMxIhei32UFR93xBCbVuZHMJZREO+mu+/xEEO7nd/wZJrTpJe|7EC3882/0o9omuvMXb1XdvDsrGO+LDL
eXsMMRHFtiTA=="}],"username":"2.ZcGwQu3c+Nv8V6FFXC5fVQ==|DEzV6N+QvBaCsGvNDHZClb66AH7aTyUzfEoG+RkZgKw=|tSllBuaFL6f2IOZnboDXv0uaV/irbxLXXyP72YW2zrw=",
"password":"2.T6OD3+TT7sS0aZ+J02NqpA==|NMOlrVdql8KjqU7ahBgpTQ==|fks7myY88cxsnhK0QD+rMoCIXI1iHMoZuK0eUHNwAg8=","passwordRevisionDate":null,"totp":nu
ll,"autofillOnPageLoad":null}},{"type":1,"encryptedFor":"c7b552de-7eda-47f0-a21f-
b314012bcecc","folderId":null,"organizationId":null,"name":"2.Va92lkY9JQOcqVCv0xPnPQ==|Rh2D+pIWqN8rGfGnl+m3sRV+AsQ9wOEu/oOyMw7QmOM=|G9MJOXAFqtoU+tp
ox4YYCB2aqXROyiWd6Ounwy7UNNw=","notes":"2.SCg81Y07NMNPvc8otPEwsg==|vC9iU88Cv4xHcxEe6wQ82w==|2ni0zuLRBKY2cA6LKYjxUiFLwZgfg3HcHAOC6dU1xfs=","favorite
":false,"lastKnownRevisionDate":null,"reprompt":0,"login":{"response":null,"uris":
[{"response":null,"match":null,"uri":"2.177MP3V1XZ/NEvTQRypkcQ==|yaCqOOthTc9Jp7iduImhHfG0iRdwgqZ0f9H+A+9GazA=|f5QTo4UH/Dzs5GXpazAUDF/pkWfDBsYmEVbUC
0+6ITs=","uriChecksum":"2.S0uzzRbd0iYyoylqkTH0Cg==|9ELnFTy1G8UPfpTldaBn4OdDIGslLlG8TQ4isccEQGSFYM8K39/hbi7XA3cRMlqV|cPQJWGnNRm3yLx5KSt0ov874ozooVeF
RiuJlpiCs0Aw=="}],"username":"2.Skg2HXPiRttuKu2nDrNTtQ==|hAQlMvW/isSEdLfpWNo1KYQXCWnHf0HRJgA3/AsE89E=|toUT86gB4afALPmz07b12USvzAknoF7f7XmyHfUZYWI=",
"password":"2.3it36a8DlakbIMJJi8GohQ==|8pJpeO6L3ikytiBuMW/Ntw==|9T5ZupOOiKjiuqs/pAHmOxeVDOH5fKM5LCEthBi3wJA=","passwordRevisionDate":null,"totp":nu
ll,"autofillOnPageLoad":null}}],"folders":
[{"name":"2.jyFQwVoTdvkwmQtEos7G4Q==|uhRIwxAJauHF75fjNbHqUg==|qPgLdX1ci1hCEn7sdPFH92+UUyCssY1R85vfz9CiHhE=","id":null}],"folderRelationships":
[{"key":1,"value":0}]}
```


The request returned `200 OK`, establishing the expected baseline response.



Fracture Labs then sent the following modified request with command injection payloads in the cipher name field:

The server returned a `400 Bad Request` response with detailed parsing information.



REMEDIATION RECOMMENDATIONS

Bitwarden should consider implementing generic error messages for validation failures and removing internal parser details (`LineNumber`, `BytePositionInLine`) from client-facing responses.

Bitwarden should also consider standardizing error responses across all endpoints to prevent information leakage and logging detailed errors server-side while returning sanitized messages to clients.

Lastly, Fracture Labs recommends reviewing all API endpoints for similar verbose error message patterns.

ADDITIONAL RESOURCES

OWASP Information Exposure Through Error Messages

https://owasp.org/www-community/Improper_Error_Handling

CWE-209: Generation of Error Message Containing Sensitive Information

https://cwe.mitre.org/data/definitions/209.html

2025.WEB.BPI.03 – IMPLEMENT FINE-GRAINED API SCOPE ASSIGNMENTS

RISK SUMMARY

Fracture Labs discovered that Bitwarden's OAuth API implementation used a single, broadly scoped API key with extensive administrative permissions for all integration use cases. The API client credentials possessed the `api.organization` scope, granting full control over critical organizational functions including member management, group administration, policy configuration, and collection management.

This over-scoped approach could create unnecessary security risks when API keys are used for limited purposes such as event collection or SIEM integration, as documented in Bitwarden's public integration guides. If these API credentials are compromised through integration systems, log files, or third-party services, threat actors could gain complete administrative control over the organization's vault.

AFFECTED ASSETS

- https://vault.bitwarden.com

TECHNICAL DETAILS

Fracture Labs analyzed the Bitwarden Organization API structure and discovered that a single OAuth client ID/secret pair provides access to extensive administrative functionality beyond what is typically needed for integration use cases.

Fracture Labs used the API keys to create members outside the organization's domain restrictions and automatically trigger invitation emails. While users must accept invitations to gain access, this demonstrated the excessive permissions granted to integration-focused API keys.

REMEDIATION RECOMMENDATIONS

Bitwarden should implement fine-grained API scoping to allow creation of limited-privilege API keys appropriate for specific integration use cases. Integration-specific scopes should be created such as `api.events` for event collection, `api.reports` for reporting functions, and `api.members.read` for user enumeration, while reserving full `api.organization` scope for legitimate administrative automation needs.

ADDITIONAL RESOURCES

SCWE-016: Insufficient Authorization Checks

https://scs.owasp.org/SCWE/SCSVS-AUTH/SCWE-016/