Bitwarden Client Applications Security Report

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

Table of Contents

Summary

In November 2024, Bitwarden engaged with cybersecurity firm IOActive to perform penetration testing and a dedicated audit of the Bitwarden client applications and SDKs. A team of testers from IOActive were tasked with preparing and executing the audit over four weeks to reach total coverage of the system under review.

Ten issues were discovered during the audit. Five issues were resolved post-assessment. Three issues were determined not feasible to address. Two issues are under planning and research. One positive finding is excluded from this report.

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 IOActive has also been attached to this report.

Issues

BITSR24-06 - Open Redirect (Critical)

Status: Resolved post-assessment.

Pull requests:

- https://github.com/bitwarden/clients/pull/12149

Hostname validation was enhanced to capture a wider range of Duo URLs.

BITSR24-11 - Master Password Input Not Cleared (Medium)

Status: Accepted.

A poor user experience would result if the master password input were cleared on every error and the existence of the incorrect master password aids the user in eventual login success.

BITSR24-10 - Account Lockout (Medium)

Status: Resolved post-assessment with additional improvements underway.

A myriad of user experience improvements have been made to the passkey login experience.

BITSR24-02 - Denial of Service via Local File Permissions (Medium)

Status: Resolved post-assessment.

Pull requests:

- https://github.com/bitwarden/clients/pull/12048

If the SSH agent spawned thread encounters errors there is no way for the TypeScript implementation to see this error, nor is there a way to see if the agent is running. This can lead to the application crashing because it does not know that the agent has not properly started. Error handling was added to protect against this.

BITSR24-01 - Denial of Service via Untrusted Search Path (Low)

Status: Resolved post-assessment.

Pull requests:

- https://github.com/bitwarden/sdk-sm/pull/1346

Reliance on an external library was removed in favor of a standard PHP library call.

BITSR24-07 - Denial of Service via CSPRNG Code Flaws (Low)

Status: Accepted, but under planning and research.

Random number generator functions have been previously audited and are considered to be implemented securely, but the Bitwarden applications will migrate the TypeScript implementation used today to a stronger Rust-based implementation in the SDK.

BITSR24-08 - SSO Organization Enumeration via Error Messages (Low)

Status: Resolved post-assessment.

Pull requests:

- https://github.com/bitwarden/server/pull/6345

Single sign-on errors have been made consistent so that their differences cannot imply any information exposure.

BITSR24-09 - Vault Timeout Options Modifiable via Local Storage (Low)

Status: Under planning and research.

User-based signatures will be generated and the user's settings signed upon write, then verified upon usage, therefore preventing tampering.

BITSR24-04 - No Warning for Auto-fill from Different Site (Info)

Status: Accepted.

A setting exists to warn the user for certain autofill scenarios but it is disabled by default to support a less interruptive user experience.

BITSR24-05 - Multi-Factor Authentication Not Enforced by Default (Info)

Status: Accepted.

Various use cases exist to not immediately require or force multifactor authentication setup. Enhancements have been made to require email verification and additional multifactor offerings continue to be introduced to the platform.

IOActive
Research-fueled Security Services

Security Assessment 2024

Bitwarden, Inc.

IOActive, Inc.
1426 Elliott Ave W
Seattle, WA 98119

Toll free: (866) 760-0222
Office: (206) 784-4313
Fax: (206) 784-4367

Technical Summary

Project Approach

The consultants used a white-box methodology, meaning they had access to source code and internal documentation, as well as internal experts.

IOActive's approach to testing adhered to industry-standard frameworks, including OWASP Application Security Verification Standard (ASVS), ensuring thorough and systematic evaluation.

The methodology comprised the following steps:

- Reconnaissance and Threat Modeling
  - Identified the architecture, components, and security-critical workflows within the Bitwarden ecosystem
  - Developed a threat model to focus testing efforts on high-risk areas
- Static Application Security Testing (SAST)
  - Reviewed source code for vulnerabilities in key areas, including cryptographic functions, input handling, and inter-process communication
- Dynamic Application Security Testing (DAST)
  - Conducted runtime analysis of the applications to identify vulnerabilities during real-world usage scenarios
  - Tested functionality, such as login workflows, autofill features, and browser extensions, under various attack conditions
- Cryptographic Analysis
  - Assessed cryptographic primitives, protocols, and key management for adherence to best practices
  - Evaluated the security of encryption mechanisms used in storing and transmitting sensitive data
- Exploit Simulation
  - Simulated common attack scenarios such as cross-site scripting (XSS), cross-site request forgery (CSRF), privilege escalation, and enumeration attacks to assess resilience
  - Performed penetration testing to identify potential misuse of exposed APIs, cookies, and local storage

Detailed Findings

#BITSR24-06 - Open Redirect

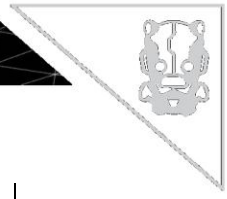
Host(s) / File(s)	https://github.com/bitwarden/clients/
Category	CWE-601: URL Redirection to Untrusted Site ('Open Redirect') CWE-807: Reliance on Untrusted Inputs in a Security Decision CWE-923: Improper Restriction of Communication Channel to Intended Endpoints
Testing Method	White Box
Tools Used	Vim
Likelihood	High (4)
Impact	Critical (5)
Total Risk Rating	Critical (20)
Effort to Fix	Low
CVSS	9.6 (Critical) - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H

Threat and Impact

Unintentional redirects and forwards are possible when a web application accepts untrusted input that directs the web application to a URL given in that untrusted input. By modifying the input to contain a malicious address, an attacker may steal data or impersonate a victim user.

As part of the Duo authentication flow, the Bitwarden client exposes an endpoint that redirects to a client-supplied URL. A malicious actor can use this endpoint and its legitimate affiliation with the Bitwarden brand as part of a social engineering attack to harvest Duo credentials of victims or take over their accounts.

Consider the following scenario: Threat actor Mal registers a domain called `myduosecurity.com`. Mal creates web content on that domain that bears the Duo brand. On a sub-page of that web content, Mal creates a form that informs the user their password may have been compromised as part of a purported Duo attack and that they need to reactivate their account by submitting and authenticating with their Duo credentials. The victim, out of concern for their account, submits those credentials in the form. At this point, Mal has harvested the victim's credentials for later password stuffing attacks, sale on the dark web, or proxies them to the `Bitwarden.com` domain for immediate use as account takeover. In the latter case, the attacker has initiated the login sequence by proxy and authenticated with one factor. Bitwarden sends the Duo 2FA challenge to the victim, which only reinforces the



authenticity of the transaction in the mind of the victim, which the victim then accepts. At this point Mal has a Bitwarden session identifier and has control of the account.

For this scenario to work, Mal has to convince the victim (Alice) that she needs to visit the malicious `myduosecurity.com` domain. This is where the open-redirect flaw comes into play. Mal creates an email with the urgent message that their Duo account at Bitwarden has been compromised and they need to reset it. The reset URL is given as `https://vault.bitwarden.com/duo-redirect-connector.html?duoFramelessUrl=https://myduosecurity.com/bitwarden-reset`.

This looks and appears to be a completely legitimate email address. First, it is running code from the `bitwarden.com` domain. Second, it is redirecting to a site that bears the Duo brand. This would fool casual users and many professionals.

This proof of concept works because, within the Duo 2FA flow, the Bitwarden web client performs an incomplete validation on the redirect domain:
`https://github.com/bitwarden/clients/blob/234a832fc445e43d799989a93bbd78428542f06e/apps/web/src/connectors/duo-redirect.ts#L58-L59`

```
if (
    validateUrl.protocol !== "https:" ||
    !(
      validateUrl.hostname.endsWith("duosecurity.com") ||
      validateUrl.hostname.endsWith("duofederal.com")
    )
  ) {
    throw new Error("Invalid redirect URL");
  }
```

Instead of matching the supplied URI against the valid domain name "`.duosecurity.com`" (note the leading dot), the code validates against any URI ending in "`duosecurity.com`" (note the missing dot). This allows an unintentional redirect to a non-Duo site, such as the malicious `myduosecurity.com` site in this proof of concept.

Recommendations

At administration time, ensure URI are checked against valid domain names, not parts of domain names. This appears to function securely [1]: when trying to enter an illegitimate Duo domain (e.g. `myduosecurity.com`) on the Settings > Security > Two-step login page, an error appears stating "Host is invalid."

At redirect time, ensure the URI is checked against valid domain names, not parts of domain names. Also, ensure the list of valid domain names here matches the list of valid domain names in the administrative area.

For this specific code, update the existing checks to look specifically for the domains "`duosecurity.com`" and "`duofederal.com`" at a minimum.

```
validateUrl.hostname.endsWith(".duosecurity.com") ||
      validateUrl.hostname.endsWith(".duofederal.com")
```

Consider also checking that the hostname begins with "`api-`", as is done in the code path at [1].

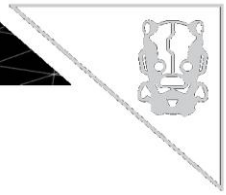
Finally, consider issuing a CVE to alert the community to patch.

Additional Information

[1] https://github.com/bitwarden/server/blob/eb20adb53eb703feacb36d38c6451f91c4e89c4c/src/Core/Auth/Identity/TokenProviders/DuoUniversalTokenService.cs#L151-L152

During the testing window, the BitWarden team was able to remediate this finding by changing the code to:

```
/**
* validate the Duo AuthUrl and redirect to it. *
@param redirectUrl the duo auth url
*/
function redirectToDuoFrameless(redirectUrl: string) { const
      validateUrl = new URL(redirectUrl); const validDuoUrl =
      validateUrl.protocol === "https:" &&
      validateUrl.hostname.startsWith("api-") && (
      validateUrl.hostname.endsWith(".duosecurity.com") ||
      validateUrl.hostname.endsWith(".duofederal.com") );
      if (!validDuoUrl) { throw new Error("Invalid redirect URL"); }
window.location.href = decodeURIComponent(redirectUrl);
}
```



#BITSR24-11 - Master Password Input Not Cleared

Host(s) / File(s)	https://github.com/bitwarden/clients
Category	CWE-312: Cleartext Storage of Sensitive Information
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Low (2)
Impact	Critical (5)
Total Risk Rating	Medium (10)
Effort to Fix	Low
CVSS	6.3 (Medium) - CVSS:3.1/AV:P/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H

Threat and Impact

During authentication, the user enters their correct master password; however, owing to a problem on the server hosting the Bitwarden vault, the authentication takes a long time to respond. The user, assuming it will happen soon, gets up to take a break. Meanwhile, the request times out after one minute and the login fails, leaving the master password input field containing the correct master password for this user. A threat actor in the same physical space as this user then exfiltrates that master password and uses it for an account takeover.

When the error is "user mistyped their master password," the threat of retaining the input password is limited; however, when the error is a timeout, there is a likelihood the password is correct and the threat of retaining it is substantially higher.

To reproduce this finding:

1. Visit `https://vault.bitwarden.com/`.
2. Arrange for a request to the Bitwarden endpoint /identity/connect/token to return with a timeout error (either induce it with excessive login attempts, insert a proxy, or wait for a maintenance window).
3. Enter the email address and master password.
4. Click "Login with master password."
5. Wait 60 seconds for the request to timeout.
6. Click the eye icon to expose the correct password.

Recommendations
Always clear the master password input field after every error, regardless of what the error is.

#BITSR24-10 - Account Lockout

Host(s) / File(s)	https://github.com/bitwarden/clients
Category	CWE-837: Improper Enforcement of a Single, Unique Action CWE-1288: Improper Validation of Consistency within Input
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Low (2)
Impact	Critical (5)
Total Risk Rating	Medium (10)
Effort to Fix	Medium
CVSS	6.3 (Medium) - CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:N/I:N/A:H

Threat and Impact

The Bitwarden UI allows passkeys to participate in log-in flows at two locations:

1. As a primary login method (aka "first factor"), configured under Settings > Security > Master Password > Login with Passkey
2. As a second factor, configured under Settings > Security > Two-step Login > Passkey > Manage

The Bitwarden implementation of passkeys in these two locations allows for the same device to be registered as a passkey; however, because passkeys operate by closing a hash over the domain, this leads to the passkey added last overwriting the passkey added first. With the passkey now overwritten, it cannot be used for its intended factor: this can lead to account lockout in the case of second factor overwrite or no longer having the convenience of passkeys as a first factor.

A threat actor could convince a victim who has a second factor passkey configured to also setup a first factor passkey on the same device, then have the victim remove that first factor, claiming they already had it as a second factor, leading to total account lockout.

In the following example, account access is denied by overwriting the second factor passkey with a first factor passkey, then removing the first factor. This attack is either a social engineering attack, where a threat actor convinces a victim to lock themselves out, or self-harm, where a naive victim accidentally configures their account and locks themself out.

Warning: ensure you have recorded the tested account's two-step account recovery code, as the following steps will cause an account lockout.

1. Enable second factor passkey in Settings > Security > Two-step Login > Passkey > Manage.

2. Choose a passkey device, like a mobile phone, to act as the authenticator.
3. In an incognito window, authenticate using email address & master password, with second factor.
    a) Expected: to be logged in
    b) Actual: as expected
4. Close that incognito window.
5. Enable first factor passkey in Settings > Security > Master Password > Login with Passkey.
6. Choose the same device as chosen in step 2.
7. In an incognito window, authenticate using email address and master password, with the second factor.
    a) Expect: to be logged in
    b) Actual: invalid passkey
8. In the same incognito window, authenticate using the passkey device.
    a) Expect: to be logged in
    b) Actual: as expected
9. Close that incognito window.
10. Remove the first factor passkey.
11. In an incognito window, authenticate using email address & master password, with the second factor.
    a) Expect: to be logged in
    b) Actual: invalid passkey
12. In the same incognito window, authenticate using the passkey device.
    a) Expect: to be logged in
    b) Actual: invalid passkey

The account is now locked out. A threat actor could encourage a victim, who is known to have passkeys as a second factor, to add a passkey first factor using the same device, then encourage them to remove the first factor since they already have it as a second factor.

Recommendations

Prevent the addition of a passkey that has already been recorded for that device and domain. This protection is already present on the Login with Passkey administration page, where, when trying to add the same device as another passkey, an error is presented ("Error creating passkey").

Additional Information

https://www.w3.org/TR/webauthn/#relying-party-identifier

#BITSR24-02 - Denial of Service via Local File Permissions

Host(s) / File(s)	https://github.com/bitwarden/clients/
Category	CWE-280: Improper Handling of Insufficient Permissions or Privileges
Testing Method	Code Review
Tools Used	Vim
Likelihood	Medium (3)
Impact	Low (2)
Total Risk Rating	Medium (6)
Effort to Fix	Low
CVSS	4.4 (Medium) - CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:N/I:N/A:H

Threat and Impact

Unix domain sockets leverage the operating system's filesystem permissions to control access to the socket file. Careful consideration must be given to the location and accessibility of a socket file before the socket is used. Storage in a common directory or overly permissive access could allow a local attacker to deny access to the socket or to intercept communication over the socket.

In `https://github.com/bitwarden/clients/tree/master/apps/desktop/desktop_native/core/src/ssh_agent/unix.rs`, if the user does not have a home directory, the application falls back to storing the Unix domain socket file at `/tmp/.bitwarden-ssh-agent.sock`. While the application attempts to remove any prior existing socket file at this path to ensure that only the application user has access to the socket file, the application does not check the return code of the file removal, which means the socket file in use may not be fully under the control of the application user.

To invoke a denial of service, an attacker needs to create a socket only they can read in the fallback location, then induce the application user

```
# (pre-condition)
# through misconfiguration or malice, victim user (alice) does
not have a home directory as seen by homedir::my_home
# this might occur on Windows machines per the documentation:
#   https://docs.rs/homedir/latest/homedir/#for-windows-users

# (pre-condition)
# local attacker user (mal) has created a bitwarden ssh auth
socket using the hard-coded fallback path from the code:
#
https://github.com/bitwarden/clients/blob/48294aac864082b22a31
```

```
6ccbb1edfb07b8220e4e/apps/desktop/desktop_native/core/src/ssh_
agent/unix.rs#L36
# and set the permissions so that the victim user (alice)
desktop client cannot remove it, as intended:
#
https://github.com/bitwarden/clients/blob/48294aac864082b22a31
6ccbb1edfb07b8220e4e/apps/desktop/desktop_native/core/src/ssh_
agent/unix.rs#L51
$ python -c "import socket as s; sock = s.socket(s.AF_UNIX);
sock.bind('/tmp/.bitwarden-ssh-agent.sock')"
$ chmod 600 /tmp/.bitwarden-ssh-agent.sock
$ ls -l /tmp/.bitwarden-ssh-agent.sock
-rw-------  1 mal   0 Nov 14 17:07 /tmp/.bitwarden-ssh-
agent.sock

# victim user launches the desktop app, with the necessary
settings to start the ssh agent along this code path
# and the application crashes with an error "[SSH Agent Native
Module] Error while starting agent server"
# because the application cannot bind to a socket that is
unwritable by the application process' user
```

A variation on this attack sees the local attacker opening the permissions of the socket to allow for any user to read and write the socket, but then uses a socket tee to record the conversation

```
# (pre-condition)
# local attacker has created a socket and made it writeable by
the victim user
$ chmod 666 /tmp/.bitwarden-ssh-agent.sock

# (pre-condition)
# local attacker has setup a socket tee, like:
#   https://unix.stackexchange.com/a/471369/50240
$ sudo ./sockdump.py /tmp/.bitwarden-ssh-agent.sock | tee -a
/home/mal/socket-capture
```

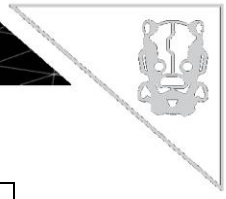
In these proofs of concept, the local attacker may not be a local user but rather an Advanced Persistent Threat (APT) infection.

Recommendations

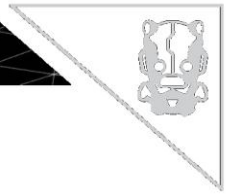
At a minimum, check the return code of the `std::fs::remove_file(sockname)` call to ensure that the socket file is fully removed. However, note this minimum fix still has a race condition, where a malicious attacker could recreate the fallback socket file in `/tmp` between the `remove_file` and `bind` calls. The best solution is to fallback into a directory that has had strict permissions set to only the app user and, if possible, to use a safe temporary file creation mechanism like `mktemp{3)`.

Additional Information

In general, when dealing with Unix domain sockets, consider these tactics in the design and implementation:



- File system permissions:
    - Create the socket file with appropriate user and group ownership, restricting access to only the intended processes.
    - Set restrictive permissions on the socket file path, ensuring only authorized processes can read and write to the socket.
- Socket path selection:
    - Avoid placing the socket file in a world-writable directory like `/tmp`.
    - Use a dedicated directory with strict permissions for your application's sockets.
- Error handling:
    - Gracefully handle potential errors during socket creation, binding, connecting, and data exchange to prevent unexpected behavior.
- Authorization:
    - Use the `getpeercred` system call (where available) to retrieve the user and group ID of the connecting process, allowing identity verification.



#BITSR24-01 - Denial of Service via Untrusted Search Path

Host(s) / File(s)	https://github.com/bitwarden/sdk
Category	CWE-426: Untrusted Search Path
Testing Method	Code Review
Tools Used	Vim
Likelihood	Low (2)
Impact	Low (2)
Total Risk Rating	Low (4)
Effort to Fix	Low
CVSS	2.2 (Low) - CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:N/I:N/A:L

Threat and Impact

The application executes an external operating system resource using an externally-supplied search path that can point to resources that are not under the application's direct control. This might allow attackers to execute their own programs, access unauthorized data files, or modify configuration in unexpected ways.

In the code at `https://github.com/bitwarden/sdk/blob/main/languages/php/src/BitwardenLib.php#L36-L36`, `uname -m` is invoked with `exec()` without sanitizing the environment or restricting the path.

```
# (pre-condition) attacker installs a malicious uname
implementation
printf 'echo "haha";' > /home/mal/uname
chmod 755 /home/mal/uname

# (pre-condition) attacker gets their path added to the
victim's environment
PATH=/home/mal:"${PATH}"

# victim invokes their program that includes the vulnerable
PHP SDK
php myprog.php

# the vulnerable PHP SDK now uses either the cross-compiled
debug version of libbitwarden_c.dylib (if present) or throw an
exception if not available.
```

```
# in either case, the outcome is not desirable for the victim
user and the reason isn't obvious as to why
```

Recommendations

When invoking other programs, specify those programs using fully-qualified pathnames, or using tools that limit the PATH to known system paths.

In this case, replace

```
$architecture = trim(exec('uname -m'));
```

with

```
$architecture = trim(exec('/usr/bin/env -i /usr/bin/uname -
m'));
```

This clears out the environment variables that might affect current or future versions of `uname` and explicitly uses the trusted system version of `uname` to identify the architecture.

#BITSR24-07 - Denial of Service via CSPRNG Code Flaws

Host(s) / File(s)	https://github.com/bitwarden/clients
Category	CWE-190: Integer Overflow or Wraparound CWE-674: Uncontrolled Recursion
Testing Method	White Box
Tools Used	Vim
Likelihood	Low (2)
Impact	Low (2)
Total Risk Rating	Low (4)
Effort to Fix	Low
CVSS	3.9 (Low) - CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:N/I:L/A:L

Threat and Impact

Edge conditions may not occur frequently within an application, but when they do, they can produce invalid, incorrect, or potentially weak results that have less security than intended or render the result unusable.

The key management service contains a random number generator:
`https://github.com/bitwarden/clients/blob/34e20b7ae86b0ac3737fd5e096303077dfa91a19/libs/key-management/src/key.service.ts#L564`

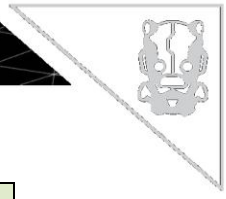
While this random number generator uses a cryptographically secure source of entropy and correctly handles bias when mapping the output range via a rejection sampling strategy, the algorithm suffers from several edge-case implementation flaws that could lead to denial of service.

First, this code does not handle integer overflow, leading to return values that are out of the desired range [1]. This was fixed upstream in a PR that was never merged [2].

```
rval = rval & mask
```

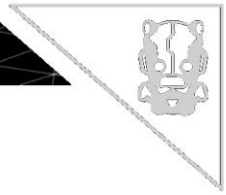
Second, this code does not bound the recursive depth, potentially leading to a stack overflow in scenarios where the entropy source produces many generations of values that are out of the desired range:

```
return this.randomNumber(min, max);
```



Recommendations
Apply the upstream commit that was never merged and switch the recursion to iteration. Or, if possible, switch the implementation to use an implementation provided by the language, such as node.js `randomInt()`

Additional Information
[1] https://jsfiddle.net/oyd9ptnv/ [2] https://github.com/EFForg/OpenWireless/pull/306/commits/d7987e0c4f6ed3121d40c6890fca79bb0efac209 Reference: https://nodejs.org/api/crypto.html#cryptorandomintmin-max-callback



#BITSR24-08 - SSO Organization Enumeration via Error Messages

Host(s) / File(s)	https://vault.bitwarden.com/#/sso?
Category	CWE-204: Observable Response Discrepancy
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Low (2)
Impact	Low (2)
Total Risk Rating	Low (4)
Effort to Fix	Low
CVSS	5.3 (Medium) - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

Threat and Impact

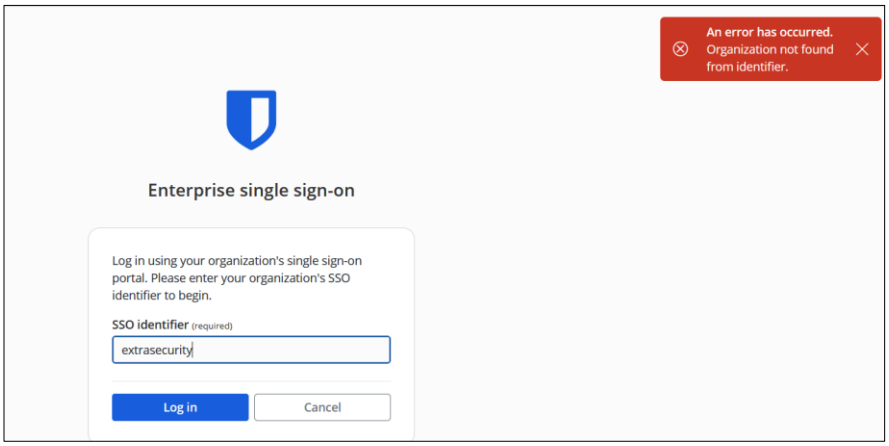
An attacker can use differing error messages from the SSO login page to enumerate valid organization IDs. Once valid organization IDs are identified, the attacker could:

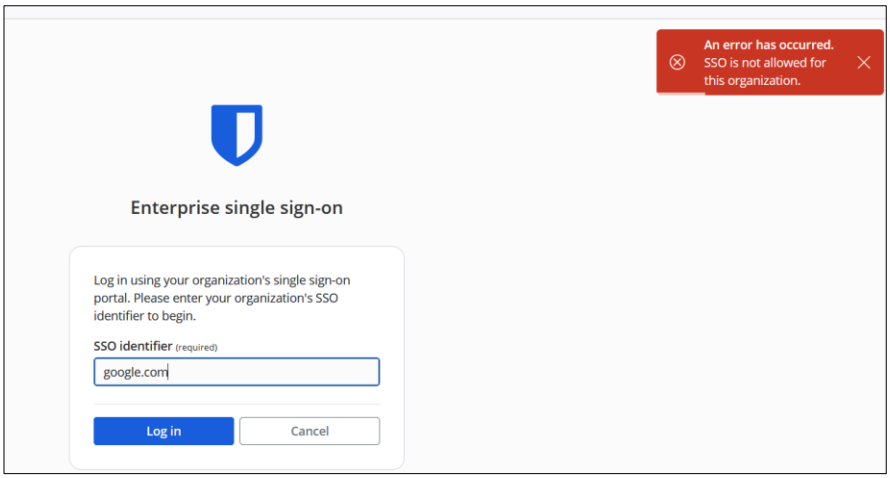
- Attempt brute-force attacks on user credentials within the organization.
- Exploit organization-specific configurations or vulnerabilities.
- Conduct phishing attacks against members of known organizations by spoofing the SSO interface.

The issue was identified by submitting random and known organization IDs to the SSO login form and observing differences in error messages and behavior. Navigate to `https://vault.bitwarden.com/#/login` and follow the standard Enterprise Login procedure.

When prompted for an organization identifier, enter different values to observe different error messages.

The below screenshots show the different error messages :



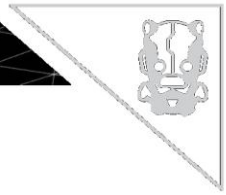


Recommendations

Standardize error messages:

- Return a generic error message regardless of the validity of the organization ID (e.g. "Invalid organization or credentials").
- Avoid revealing whether the organization ID is valid or invalid.

Implement rate limiting or CAPTCHA to deter automated enumeration attempts.



#BITSR24-09 - Vault Timeout Options Modifiable via Local Storage

Host(s) / File(s)	Local Storage Data
Category	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Informational (1)
Impact	Medium (3)
Total Risk Rating	Low (3)
Effort to Fix	Low
CVSS	5.1 (Medium) - CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

Threat and Impact

Vault timeout settings dictate how and when a user's session times out and requires re-authentication.

If these settings are stored in modifiable locations:

- Attackers with access to the local environment can modify timeout settings to extend sessions indefinitely, bypassing timeout policies.
- This increases the risk of sensitive data being exposed if the vault remains accessible without proper re-authentication.

In compromised environments, unauthorized users could exploit this to maintain persistent access to the vault.

Examine the local storage while logged into the vault. Change the `....vaultTimeoutSettings_vaultTimeout` to an arbitrary number. The current session reflects that change.

Recommendations

Use encrypted values to prevent tampering by unauthorized users.

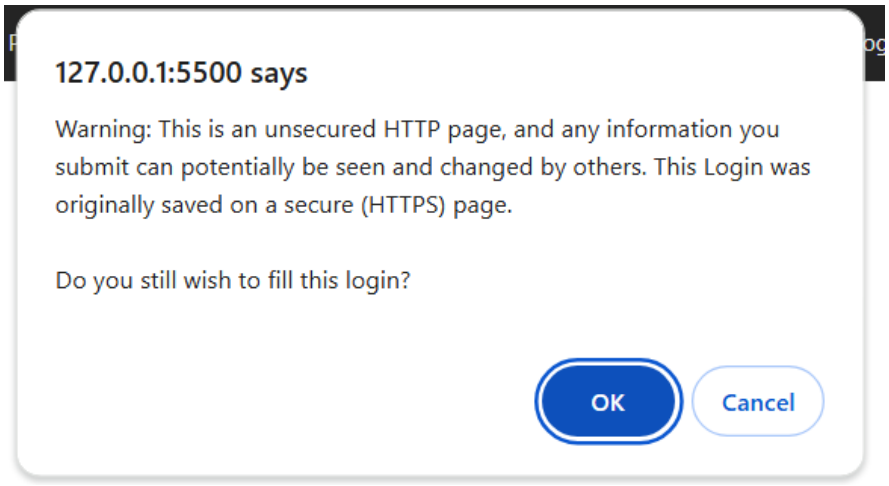
Additional Information
During testing, it was discovered that the Organization rules were validated on each login. This means that a bad actor would need to change that value during a current session. As a result, the severity of the issue was reduced to low.

#BITSR24-03 - Warning on HTTPS downgrade (Positive-Finding)

Host(s) / File(s)	Browser Extension
Category	CWE-319: Cleartext Transmission of Sensitive Information
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Informational (1)
Impact	Informational (1)
Total Risk Rating	Informational (1)
Effort to Fix	Low
CVSS	0.0 (Informational) - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

Threat and Impact
The browser extension appropriately detects when a site is downgraded from HTTPS to HTTP and provides a warning to the user. This behavior helps protect users from inadvertently transmitting sensitive data over an insecure connection. The lack of such a warning would present a risk where users may unknowingly submit sensitive data over an insecure channel (HTTP), leading to potential exposure of data during transit (e.g., passwords, tokens, sensitive forms). This behavior mitigates the risk of man-in-the-middle (MiTM) attacks, where an attacker intercepts data transmitted over an unencrypted connection.

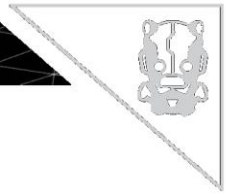
A test site was created that served a login page through HTTPS. Login, and save the credentials in the browser extension, then reload the test site over HTTP. The browser extension will show a pop-up that warns the users of the possible fake site.



Recommendations

Continue to provide warnings to users when a site downgrades from HTTPS to HTTP, emphasizing the potential risks of data exposure.

Consider adding user-configurable options that allow the user/organization to define how the extension handles mixed content scenarios, such as blocking auto-fill on insecure pages or providing additional details about the risks.

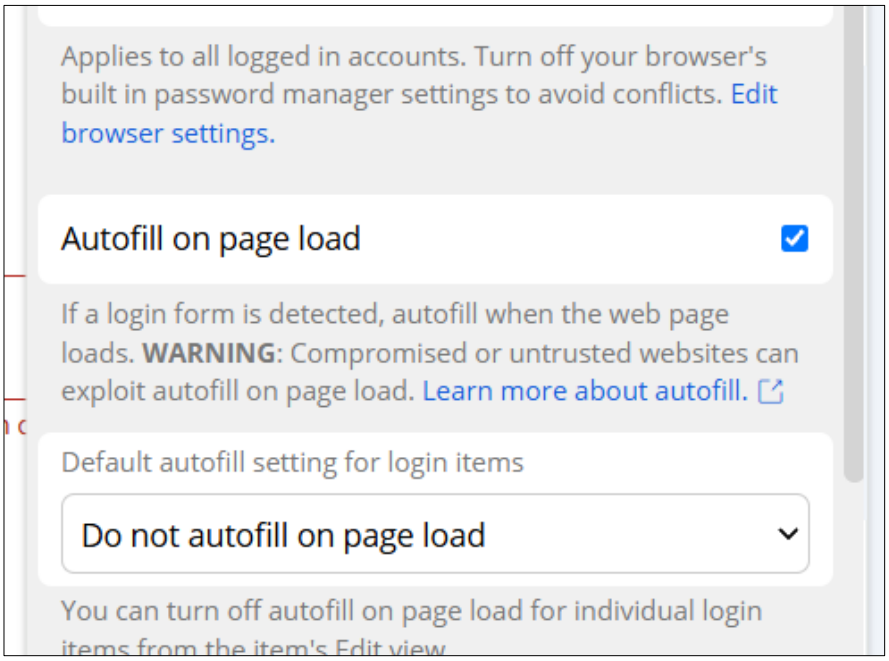


#BITSR24-04 - No Warning for Auto-fill from Different Site

Host(s) / File(s)	Bitwarden Browser Extension
Category	CWE-601: URL Redirection to Untrusted Site ('Open Redirect')
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Informational (1)
Impact	Informational (1)
Total Risk Rating	Informational (1)
Effort to Fix	Low
CVSS	0.0 (Informational) - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

Threat and Impact
The browser extension does not display a warning or prompt to the user when auto-filling credentials on a site that differs from the one for which the credentials were originally saved. This behavior could lead to unintended auto-filling on phishing or malicious websites that mimic the appearance of trusted sites, potentially exposing sensitive information such as usernames and passwords. This behavior introduces a potential risk that credentials could be leaked to malicious sites attempting to impersonate trusted domains. Attackers can exploit this to harvest credentials by creating lookalike sites that trigger auto-fill behavior, increasing the risk of user data theft and compromising account security.

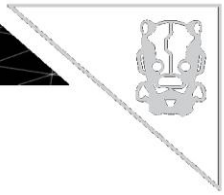
Note that a warning is given to the user when they enable auto-fill on page-load:



To reproduce this finding, save credentials for a test website, such as `https://example.com`. Visit a visually similar but different domain, such as `https://example-login.com`, and use the extension's auto-fill feature. The extension will fill in the saved credentials without providing a warning or prompt to the user.

Recommendations

While it is impossible to defeat user error, a user warning prompt could be used to help prevent users from being phished.



#BITSR24-05 - Multi-Factor Authentication Not Enforced by Default

Host(s) / File(s)	Bitwarden
Category	CWE-306: Missing Authentication for Critical Function
Testing Method	White Box
Tools Used	Web Browser
Likelihood	Informational (1)
Impact	Informational (1)
Total Risk Rating	Informational (1)
Effort to Fix	Low
CVSS	0.0 (Informational) - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

Threat and Impact
Multi-Factor Authentication (MFA) is an essential security control designed to enhance the protection of user accounts by requiring additional verification beyond the primary password. Currently, the browser extension allows users to enable or disable MFA at their discretion, and it is not enforced by default. This could leave users' accounts more susceptible to unauthorized access if they choose not to enable MFA, particularly if their primary credentials are compromised. Without enforced MFA, user accounts are more vulnerable to attacks such as password breaches, phishing, or credential stuffing. If a user's primary credentials are compromised, an attacker could gain full access to their account and any associated sensitive data. When creating an account, a user is just required to create a single master password.
Recommendations
Enforce MFA for all users during the account setup process, or strongly encourage it with repeated prompts until enabled.
Additional Information
Implementing MFA in a way that aligns with a "secure by design" philosophy ensures that security is built into the core of the user experience from the start. By enforcing or strongly encouraging MFA during initial account setup, you reduce the risk of unauthorized access, making security an inherent part of account usage rather than an optional feature. Secure by design principles prioritize proactive

measures, minimizing the attack surface and offering users the highest level of protection possible from the moment they interact with the extension.

To further support this design philosophy, consider making MFA opt-out only under specific circumstances, offering clear prompts about security risks when disabled, and providing robust options for MFA methods to meet diverse user needs. This approach aligns with security best practices and establishes a baseline level of trust and resilience for all users.

Appendix A: Overview of Detailed Findings

Host(s) / File(s)

This section includes a list of the assets affected by the finding.

Category

IOActive uses Common Weakness Enumeration (CWE™)[1] identifiers to categorize each finding. CWE is a community-developed list of software and hardware weakness types that have security ramifications. This software assurance strategic initiative is sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security and published by The MITRE Corporation.

Testing Method

The testing method captures the approach that the consultants used to discover the finding.

Table 2. Examples of testing methods

Method	Description
Black Box	The consultants had no internal knowledge of the target and were not provided with any information that was not publicly available.
Grey Box	The consultants had access and knowledge levels comparable to a user, potentially with elevated privileges. The consultants may also have been provided documentation, accounts, or other information.
White Box	The consultants had full access to the target's source code, documentation, etc.

Tools Used

The section lists the specific tools the consultants used to discover the finding.

[1] https://cwe.mitre.org/

Likelihood and Impact

IOActive assigns two ratings for each finding: one for likelihood and another for impact. Each rating corresponds to a numeric score ranging from 5 (critical) to 1 (informational).

Table 3. Description of likelihood and impact

Rating (Score)	Likelihood	Impact
Critical (5)	The finding is almost certain to be exploited; knowledge of the issue and how to exploit it are in the public domain	Extreme impact to the entire organization if exploited
High (4)	The finding is relatively easy to detect and exploit by an attacker with low skills	Major impact to the entire organization or a single line of business if exploited
Medium (3)	A knowledgeable insider or expert attacker could exploit the finding without much difficulty	Noticeable impact to a line of business if exploited
Low (2)	Exploiting the finding would require considerable expertise and resources	Minor damage if exploited or could be exploited in conjunction with other vulnerabilities as part of a more serious attack
Informational (1)	The finding is not likely to be exploited on its own but may be used to gain information for launching another attack	Does not represent an immediate threat but may have security implications if combined with other vulnerabilities

Total Risk Rating

IOActive then calculates a total risk score by multiplying likelihood and impact.

Table 4. Total risk rating and corresponding aggregate risk scores

Total Risk Rating	Total Risk Score Range (Likelihood × Impact)
Critical	20–25
High	12–19
Medium	6–11
Low	2–5
Informational	1

Effort to Fix

IOActive estimates the effort it will take to fix the finding based on our consultants' experience. An organization's actual effort may vary based on factors such as skill sets, process efficiency, and available resources.

CVSS (Optional)

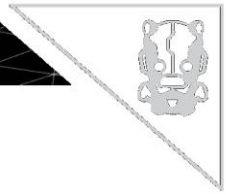
IOActive may also use the Common Vulnerability Scoring System (CVSS)[2] to capture the principal characteristics of a finding and produce a numerical score reflecting its severity. CVSS is used by organizations worldwide to supply a qualitative measure of severity; however, CVSS is not a measure of risk.

IOActive assigns a value to each metric of the scoring system.

Table 5. CVSS metrics and selectable values

Metric	List of Values
Attack Vector (AV)	Network (N) Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC)	Low (L) High (H)
Privileges Required (PR)	None (N) Low (L) High (H)
User Interaction (UI)	None (N) Required (R)
Scope (S)	Unchanged (U) Changed (C)
Confidentiality (C)	None (N) Low (L) High (H)
Integrity (I)	None (N) Low (L) High (H)
Availability (A)	None (N) Low (L) High (H)

[2] https://www.first.org/cvss/



These values translate to a base score[3] and severity rating.

Table 6. CVSS 3.1 base score and associated rating

Severity Rating	Base Score Range
Informational	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

[3] https://www.first.org/cvss/calculator/3.1