Bitwarden Web App Report

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

Table of Contents

Summary

In August 2023, Bitwarden engaged with cybersecurity firm Cure53 to perform penetration testing and a dedicated audit of the Bitwarden web application. A team of two senior testers from Cure53 were tasked with preparing and executing the audit over two days to reach total coverage of the system under review.

Two issues were discovered during the audit. One issue was resolved post-assessment. One issue was determined not feasible to address.

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 report delivered by Cure53 has also been attached to this report.

Issues

BWN-08-001 WP4: Storage-enabled unlocking of client-side premium features (Low)

Status: Accepted as a low business risk.

As an open-source application, Bitwarden accepts that certain bad actors may locally bypass feature checks, enabling access to premium features for free accounts. This is ultimately a low business risk rather than a security risk as these actors are not gaining access to any sensitive data.

BWN-08-002 WP4: No password complexity checks on vault-export (Medium)

Status: Issue was fixed post-assessment.

Pull requests:

- https://github.com/bitwarden/clients/pull/6936

The application now has a password complexity meter on the vault export experience to better educate the user around brute force attacks. This should drive the user to create strong export passwords, similarly to other password-creation experiences in the app.

Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, each ticket has been given a unique identifier (e.g., *BWN-08-001*) to facilitate any future follow-up correspondence.

BWN-08-001 WP4: Storage-enabled unlocking of client-side premium features *(Low)*

The Bitwarden web application stores the user and its properties to both the local storage and the session storage of the browser. The web application creates a new entry in both storage items with the user's ID as key containing *data*, *keys*, *profile*, *settings* and *tokens* properties. The *profile* property in turn contains an object that has the *hasPremiumFromOrganization* and *hasPremiumPersonally* properties. Cure53 noted that the web application used these properties directly to determine if the user had a premium account, doing so without validating them on the backend for certain features.

Hence, an attacker could unlock some premium features without paying for them. Particularly, the attacker can use the *Exposed passwords* and *Weak passwords* features of the web application for this purpose.

Steps to reproduce:

1. Log in to the Bitwarden web application with a user that is neither associated with a premium organization nor has a premium account.
2. Navigate to the *Reports* section and open the developer tools of the web browser.
3. Execute the command shown below in the developer console of the browser to get all data associated with the authenticated account. The user's ID used for this PoC corresponds to *064c89f9-717d-42f9-94d7-b05700bfeed4*.

   Command:
   ```
   sessionStorage.getItem('064c89f9-717d-42f9-94d7-b05700bfeed4')
   ```

   Result:
   ```
   '{[...],"profile":{"userId":"064c89f9-717d-42f9-94d7-
   b05700bfeed4","name":"<b>AP</
   b>","email":"apirker+bwn1@cure53.de","hasPremiumPersonally":false,"kdfIte
   rations":2,"kdfMemory":16,"kdfParallelism":1,"kdfType":1,"keyHash":"EGt9l
   Yr160lHCcp5naIPhiXLCQz4DZzJgaAx8yeIo4U=","emailVerified":false,"hasPremiu
   mFromOrganization":false,"usesKeyConnector":false,"convertAccountToKeyCon
   nector":null},[...]}'
   ```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

4. The result demonstrates that the user does not have a premium account, neither personally nor organizationally.

5. Flip both *hasPremiumPersonally* and *hasPremiumFromOrganization* properties from *false* to *true*. This updates the entry of the current user in the session storage of the browser.

   Command:

```
data = '{[...],"profile":{"userId":"064c89f9-717d-42f9-94d7-
b05700bfeed4","name":"<b>AP</
b>","email":"apirker+bwn1@cure53.de","hasPremiumPersonally":true,"kdfIter
ations":2,"kdfMemory":16,"kdfParallelism":1,"kdfType":1,"keyHash":"EGt9lY
r160lHCcp5naIPhiXLCQz4DZzJgaAx8yeIo4U=","emailVerified":false,"hasPremium
FromOrganization":true,"usesKeyConnector":false,"convertAccountToKeyConne
ctor":null},[...]}''
sessionStorage.setItem('064c89f9-717d-42f9-94d7-b05700bfeed4', data)
```

6. Click on the *Weak passwords* premium content of the web application. The premium content is opened, as demonstrated by the figure below.

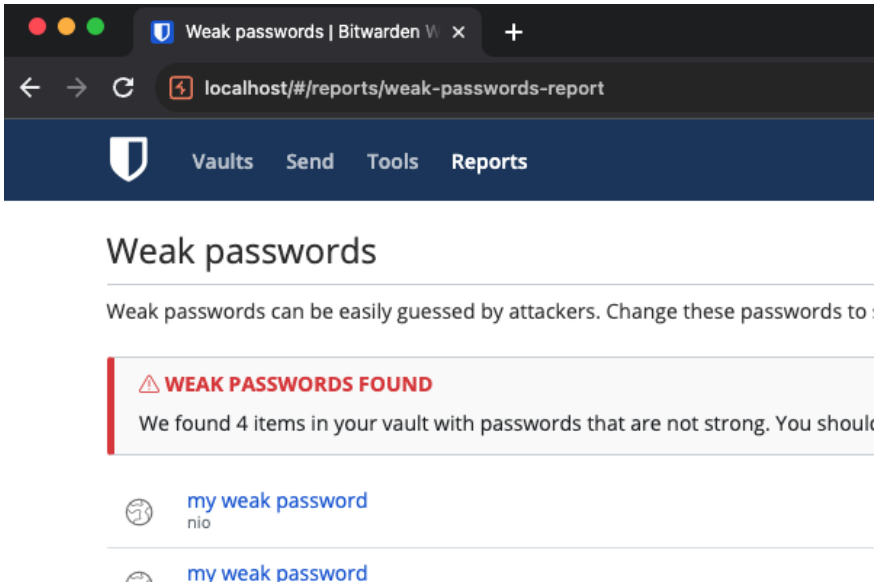

Fig.: Access to premium client-side features without a premium account.

Cure53 advises cross-checking the premium flags with the backend service on every attempt towards usage of premium features. Alternatively, the client should not persist any premium flags and solely query the backend on every access to the premium features.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, whilst a vulnerability is present, an exploit may not always be possible.

BWN-08-002 WP4: No password complexity checks on vault-export *(Medium)*

Dynamic testing of the Bitwarden web application revealed exposing exports of vault entries into *csv*, *json* and *encrypted json* format. The exports into *csv* and *json* are done in plaintext, whereas the export into *encrypted json* format involves the encryption of the vault using AES-CBC.

The web application prompts the user for a password and uses this password as input to a key derivation function, which is used to generate a symmetric key for vault encryption. The web application does not perform any complexity checks or similar operations on the provided passwords.

Even though the web application applies a key derivation function to the password, the lack of password complexity checks could tremendously simplify brute-force attacks. This is due to potentially weak passwords chosen by the user. An attacker who manages to acquire an encrypted JSON vault could attempt to brute-force the password used for the encryption of the vault.

Affected file:
clients-master/libs/exporter/src/vault-export/services/vault-export.service.ts

Affected code:

```
async getPasswordProtectedExport(password: string, organizationId?: string):
Promise<string> {
  [...]
  const key = await this.cryptoService.makePinKey(password, salt, kdfType,
       kdfConfig);
  const encKeyValidation = await this.cryptoService.encrypt(Utils.newGuid(),
       key);
  const encText = await this.cryptoService.encrypt(clearText, key);
  [...]
}
```

It is recommended to perform suitable complexity checks on the password the user provides when exporting a vault to an encrypted *JSON* file[1].

[1] https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#impl[...]-controls