Fine penetration tests for fine websites

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

Pentest-Report 1Password B5 Web & API 10.-11.2021

Cure53, Dr.-Ing. M. Heiderich, MSc. S. Moritz, MSc. N. Krein

Index

Introduction

“1Password is the easiest way to store and use strong passwords. Log in to sites and fill forms securely with a single click.”

From https://1password.com/

This report describes the results of a security assessment of the 1Password B5 web application complex. Carried out by Cure53 in October 2021, the project included a penetration test and a dedicated audit of the source code.

Registered as *1PW-17,* the project was requested by 1Password in April 2021 and then scheduled as part of the more structured series of security-centered assessments conducted by Cure53 for the 1Password team. Examinations of the 1Password B5 application have taken place with certain regularity in 2020 and 2021.

As for the precise timeline and specific resources allocated to *1PW-17*, Cure53 completed the examination in mid-to-late October 2021, namely in CW40 and CW41. A total of fourteen days were invested to reach the coverage expected for this assignment, whereas a team of three senior testers has been composed and tasked with this

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

project's preparation, execution and finalization. All testers in this group were already familiar with the 1Password B5 web app compound via previous project work.

For optimal structuring and tracking of tasks, the work was split into two separate work packages (WPs):

- **WP1**: 1Password B5 web application frontend, written predominantly in TS
- **WP2**: 1Password B5 web application backend, written predominantly in GoLang

Cure53 was given access to a dedicated 1Password B5 test environment as well as furnished with all relevant sources. Additionally, test-supporting documentation, lots of details pertinent to focus areas and related documentation were provided to make sure the project can be executed in line with the agreed-upon framework. Following best practices established during past cooperation, the methodology utilized in *1PW-17* was accordingly white-box.

The project progressed effectively on the whole. All preparations were done in CW39 to foster a smooth transition into the testing phase. Over the course of the engagement, the communications were done using a private, dedicated and shared Slack channel which was spun up by 1Password. All partaking team members could join test communications there. The discussions throughout the test were very good and productive and not many questions had to be asked. The scope was well-prepared and clear, with no noteworthy roadblocks encountered during the test.

Cure53 offered frequent status updates about the test and the emerging findings. Live-reporting was performed as usual and the Cure53 team managed to acquire good coverage over the test targets delineated for WP1 and WP2 of this *1PW-17* project.

Among six security-relevant discoveries, four were classified to be security vulnerabilities and two to be general weaknesses with lower exploitation potential. It needs to be noted that this is a good result for 1Password, largely asserting that frequent engagement with external tests and audits positively contributes to the security posture of the complex.

None of the findings were marked as *Critical* and only one discovery represented a borderline *High* risk. This problem would let an attacker cause a DoS of the 1Password account under certain circumstances related to invalid vault data submission. Other flaws related to DoS and possible reuse of leaked information in browser storage containers. Both the severities and the total number of issues can be seen as manageable.

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

In the following sections, the report will first shed light on the scope and key test parameters, as well as the structure and content of the WPs. Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following a chronological order in each group. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions about this autumn 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team's observations and collected evidence. Tailored hardening recommendations for the 1Password B5 web application complex are also incorporated into the final section.

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

Scope

- **Penetration-Tests & Code Audits against 1Password B5 Web Application**
  - **WP1**: 1Password B5 Web Application Frontend, written predominantly in TS
    - Test environment:
      - https://1pw17testcompany.b5test.com/
    - All relevant sources were shared with Cure53
  - **WP2**: 1Password B5 Web Application Backend, written predominantly in GoLang
    - Key focus areas:
      - New 1Password vault invite flow
      - Service account authorization feature
      - Item sharing feature
      - Duo v4 MFA integration
    - All relevant sources were shared with Cure53
- **Test user-accounts:**
  - https://1pw17testcompany.b5test.com/
    - U: seba@cure53.de
    - U: niko@cure53.de
  - https://1pw17second.b5test.com/
    - U: seba@cure53.de
  - https://1pw17third.b5test.com
    - U: seba@cure53.de
  - https://1pw17fourth.b5test.com
    - U: seba@cure53.de
  - Duo MFA:
    - https://admin-da89dc78.duosecurity.com/login
    - U: seba@cure53.de
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

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

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *1PW-17-001*) for the purpose of facilitating any future follow-up correspondence.

1PW-17-001 WP1: Reusable authorization token in *localStorage* *(Medium)*

***Note**: The 1Password team has accepted this finding as a low severity issue. They cannot follow the provided recommendation for usability reasons, but are looking to mitigate the risk of accidentally long-lived authentication tokens by introducing an opt-out for storing this in a future version of item sharing.*

During the assessment of the newly introduced item sharing feature, the discovery was made that the currently implemented authentication mechanism is prone to several local attack scenarios. The feature furnishes the option to define a set of accessors via email addresses. These determine where the item can be shared.

After the email gets confirmed via an OTP token, an additional authorization token is returned and is stored in the *localStorage*. Due to the fact that items in the *localStorage* remain after the browser is closed and the token is reusable, a local acting adversary is able to use this token to access all valid items shared with the email address.

Moreover, the expiration date of the authorization token is set to one month, which greatly increases the likelihood of being exploited. Please keep in mind that URLs are stored together with the *location.hash* value in the browser's history, which also make it attractive as a route to easily access unexpired shared items.

Affected content of *localStorage*:

```
console.log(localStorage.getItem('token_<email>'))
```

Resulting content:

```
{"token":"lMVEQPMSTxKdHtJjMp7RFROLqwSIZa8h","email":"seba+inviter@cure53.de","ex
piresAt":"2021-11-05T09:56:27.717Z"}
```

It is recommended to cease storing sensitive data such as tokens in the *localStorage*[1]. Instead, it is advisable to use the *HTML5 sessionStorage* which stores data for the current browser session only. Thus, the data is deleted once the browser has been

[1] https://dev.to/rdegges/please-stop-using-local-storage-1i04

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

closed and local attackers have no access to previously-stored data in a new browser session. In addition, in order to further harden the current implementation, consideration should be given to limiting the token's expiry and performing email verification for each item shared to the address.

1PW-17-003 WP2: Missing check allows spoofing of *Activity Logs* *(Low)*

***Note**: The 1Password team has reviewed the issue and confirms it is present, but was not able to accept this issue. This is a known and accepted consideration in how item sharing works: 1Password cannot access your secrets, and therefore cannot see what secrets users share. As a result, 1Password can only verify if a user shares a secret they have access to, but not which of those secrets that is.*

It was found that the *Activity Log* feature available for higher-privileged 1Password users suffers from several spoofing attacks related to the shared items. The current implemented design randomly generates *secret bytes* for each new share. A unique *UUID*, a *token* and a *rawKey* are derived from these. The *rawKey* is used to encrypt the content, which is sent along with the *token*, the *UUID* of the shared item, the *UUID* of the vault and the *UUID* of the item to the server.

Leveraging the current design, an adversary is able to send other items, vault *UUIDs* or other encrypted content that do not belong to the shared item. This allows spoofing entries to create fake logs of shared items, whereby other content was made publicly available. The impact can be considered limited due to the fact that this might be used in scenarios where admins safely rely on the values behind shared items.

Affected request (decrypted):

```
POST /api/v3/itemshare HTTP/2
Host: 1pw17testcompany.b5test.com
X-Agilebits-Mac: v1|19|oLlvjks7SgGGwzYe
X-Agilebits-Session-Id: ZIEBQG5J4NE7FMNFXV6JFUYS5Q
[...]

{
  "uuid" : "ngcj47nuwsaa24dr2pc6i5zqau",
  "token" : "5nzdbyuza7w7ep75i7sathqs5e",
  "vaultUuid" : "k2l2l5ushqwardrwzu5r5y2wxu",
  "itemUuid" : "dzidcgo2ki6poscx25cxar35ru",
  "itemVersion" : 1,
  "encOverview" : {
[...]
```

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

Steps to reproduce:

1. Open the item selected for sharing with others.
2. Open *b5-*min.js* in the DevTools and place a breakpoint on line:
   `e.post('/api/v3/itemshare', t).then(Object(o.unsafeDecodeAs) (f))]`
3. Click on *"Share…"* and *"Get Link to Share"*.
4. Open the console in DevTools and add the *UUID* of an item that is supposed to be shown in the logs: `t.itemUuid="<itemUUID>"`
5. Continue the script and open the resulting *share* link.

As a result, the initial item is shown on the shared page, whereby the spoofed item is displayed in the logs.

Since the server has no knowledge about decrypted contents and the issue does not seem to be solvable on the client-side, the risk is somewhat diminished. Still, it is recommended to implement additional verifications in order to raise the bar for further exploitations.

1PW-17-004 WP2: DoS against other users via blocklist *(Medium)*

***Note**: The 1Password team confirms this issue is present and has accepted it as a best practice issue. 1Password has no plans to address this in the short term but are looking at alternatives for their request filter mechanisms that would be less susceptible to this issue.*

It was found that the 1Password web application is still prone to a Denial-of-Service attack. While the issue has already been reported by Cure53 in January 2019 (see *1PW-01-005*), it was found that it has still not been addressed accordingly. The implemented defense mechanism adds an IP to the blocklist if some of the defined strings are added to request parts, for instance to the URL.

After the IP is added, the client is no longer able to reach the application and the status *403 Forbidden* is returned. This effectively prevents the users from a prolonged engagement with the product if a prepared URL gets sent from the client. Please note that those URLs can also be opened without user-action from other origins, for example via simple *fetch* requests.

Affected files:
server/src/main/middleware/filter.go

Affected code:

```
func filter(hosts []string, handler http.Handler) http.Handler {
```

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

```
      requestAnalyzer := activedefence.NewRequestAnalyzer(hosts,
pathsRequiringSession())

   return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
      ip := r.Header.Get(remote.XIPHeader)

      if blocklist.Includes(ip) && !allowlist.Includes(ip) {
         renderForbidden(w, r, "Blocklist is returning blocklisted response")
            return
      }
[...]
banList.AddFilter(`alert(`)
[...]
```

PoC URL:
https://1pw17testcompany.b5test.com/alert(

The following PoC shows how such a request can be fired from other origins. Please note that accessing the response via JavaScript is not needed for this type of attack.

PoC HTML (via fetch):

```
<script>fetch("https://1pw17testcompany.b5test.com/alert(")</script>
```

Nevertheless, it is recommended to switch to a more WAF-based approach, which would only block specific requests instead of both requests and IPs. This would prevent exploiting the behavior as a stepping stone for further Denial-of-Service attacks against 1Password users.

1PW-17-006 WP1: Client-side DoS via invalid encrypted data in vaults *(High)*

***Note**: The 1Password team confirms this issue was present and has addressed this in currently released versions of their product.*

During the assessment of the client-side parts of the B5 application, the discovery was made that the application suffers from a Denial-of-Service attack. During vault creation, an *access* object containing encrypted keys is sent to the server. In case the object contains invalid encrypted data, the client-side parts will throw the error of *"Key IDs do not match"* during decryption. As a result, all functionalities that require accessing decrypted vault content are affected, including *Dashboard*, *People*, *Vaults* or *Integrations*.

Due to the fact that the behavior is also exploitable for lower-privileged users and it affects the whole 1Password account, the issue was rated as *High*.

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

Affected file:
client/web-api/src/model/symmetric_key.ts

Affected code:
```
public decrypt = async (json: JweB): Promise<Uint8Array> => {
   if (CONTENT_TYPE !== json.cty) {
      throw new Error("Unexpected content type");
   }
   if (json.kid !== this.uuid) {
      throw new Error("Key IDs do not match");
   }
[...]
```

The following PoC shows how a faulty vault entry can be created.

PoC request example:
```
{
  "uuid" : "fbgkclhge7vbxthhlfsrk6dhwf",
  "type" : "U",
  "attrVersion" : 1,
  "contentVersion" : 1,
  "encAttrs" : {
    "kid" : "2skapppepcf4krmir32zpt2qx4",
    "enc" : "A256GCM",
    "cty" : "b5+jwk+json",
    "iv" : "BvAy24vWyl5-yQcs",
    "data" : "-
AFRgqx3XellWUHQ_87crCw8TX3GFVAyUBKl1v5T5Yw6M9xL3JZ83db7_eAqTvjjSZiR9sg5p__xDsiA9
vn-
z1y4cUHVhLBvoTDD5LyE0B8n2IhcmNBc5pUAu_9ZoT8vm0H5jqZEpyn7ENVFibGJewtp4JE_aNt2PZVn
AclYK7G7vmvuZXpqvr3szm6DcWtYoTbgfOee84LD1GCdCzpoL9PYAapSCul_wKMJc76_"
  },
  "access" : [ {
    "vaultUuid" : "fbgkclhge7vbxthhlfsrk6dhwf",
    "accessorType" : "user",
    "accessorUuid" : "QMV4E5KYXJGLDFEVJRPYSR72YQ",
    "acl" : 2147483646,
    "leaseTimeout" : 0,
    "vaultKeySN" : 1,
    "encryptedBy" : "3ohtmkjj4emonbbnemjgpzho3y",
    "encVaultKey" : {
      "kid" : "3ohtmkjj4emonbbnemjgpzho3y",
      "enc" : "RSA-OAEP",
      "cty" : "b5+jwk+json",
      "data" : "INVALID"
    }
  }, {
    "vaultUuid" : "fbgkclhge7vbxthhlfsrk6dhwf",
```

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


```
    "accessorType" : "group",
    "accessorUuid" : "qz3gw6t42vwdlgswpivbxeblhq",
    "acl" : 1,
    "leaseTimeout" : 0,
    "vaultKeySN" : 1,
    "encryptedBy" : "xujdyj4y2o3tdvdaeze2zob4ai",
    "encVaultKey" : {
      "kid" : "xujdyj4y2o3tdvdaeze2zob4ai",
      "enc" : "RSA-OAEP",
      "cty" : "b5+jwk+json",
     "data" : "INVALID"
    }
  }, {
    "vaultUuid" : "fbgkclhge7vbxthhlfsrk6dhwf",
    "accessorType" : "group",
    "accessorUuid" : "abszsozul3zrtf2csq4dmqx22e",
    "acl" : 2,
    "leaseTimeout" : 0,
    "vaultKeySN" : 1,
    "encryptedBy" : "xk2tbxw6fyfhrp7rkrufoahzfi",
    "encVaultKey" : {
      "kid" : "xk2tbxw6fyfhrp7rkrufoahzfi",
      "enc" : "RSA-OAEP",
      "cty" : "b5+jwk+json",
     "data" : "INVALID"
    }
  }, {
    "vaultUuid" : "fbgkclhge7vbxthhlfsrk6dhwf",
    "accessorType" : "group",
    "accessorUuid" : "kc5gnate6iqlfdrfvlc3cjghxu",
    "acl" : 2,
    "leaseTimeout" : 0,
    "vaultKeySN" : 1,
    "encryptedBy" : "ilaprfxvecjz36itrky5zajaoi",
    "encVaultKey" : {
      "kid" : "ilaprfxvecjz36itrky5zajaoi",
      "enc" : "RSA-OAEP",
      "cty" : "b5+jwk+json",
     "data" : "INVALID"
    }
  } ]
}
```


Steps to reproduce:

1. Create a new vault and intercept the request with the provided Burp plugin.
2. Replace the content in the data field of an accessor object with invalid data (see PoC above).
3. Send the edited request to the server.

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

4. Click on *Dashboard* or *Vaults* and observe an error being thrown. The account is no longer usable.

It is recommended to introduce proper exception handling on the affected client-side parts of the application. By doing so, an error would not interrupt a continuous engagement with the product. The web interface should still be displayed even when undecryptable data is passed.

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

1PW-17-002 WP2: Inconsistent check leads to faulty *Activity Logs* *(Low)*

***Note**: The 1Password team has accepted this finding as a best practice issue and will address it in an upcoming release.*

A further investigation of *1PW-17-003* revealed that the application lacks some proper checks when a shared item entry is being created. The affected code part does not prevent creating new audit event records with UUIDs that already exist in the database. Due to the fact that an entry is firstly created in the *Activity Log* component, inconsistent data can be created in the logs of events pointing to other shared items, even if the final creation fails.

Affected file:
server/src/db/audit/item_share.go

Affected code:

```
// CreateItemShare creates a new audit event record when sharing an item
externally
func CreateItemShare(tx *datastore.Tx, userSession *model.UserSession, vaultItem
*model.VaultItem, vault *model.Vault, shareUUID string) error {
	event := userSession.NewAuditEvent(ActionShare, ObjectVaultItem,
vaultItem.ID, vaultItem.UUID, &model.AuxInfo{ID: vault.ID, UUID: vault.UUID,
Info: shareUUID})
	if err := tx.AddAuditEvent(event); err != nil {
		return errors.Wrapf(err, "CreateItemShare audit failed for user %d
%s, item %v, vault %v, share %v", userSession.UserID, userSession.UserUUID,
vaultItem, vault, shareUUID)
	}
```

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

Affected request (decrypted):

```
POST /api/v3/itemshare HTTP/2
Host: 1pw17testcompany.b5test.com
X-Agilebits-Mac: v1|19|oLlvjks7SgGGwzYe
X-Agilebits-Session-Id: ZIEBQG5J4NE7FMNFXV6JFUYS5Q
[...]

{
  "uuid" : "ngcj47nuwsaa24dr2pc6i5zqau",
  "token" : "5nzdbyuza7w7ep75i7sathqs5e",
  "vaultUuid" : "k2l2l5ushqwardrwzu5r5y2wxu",
  "itemUuid" : "dzidcgo2ki6poscx25cxar35ru",
  "itemVersion" : 1,
  "encOverview" : {
[...]
```

It is recommended to first check if the received UUID of a shared item already exists in the *Activity Log* component. By doing so, the operation should be aborted, which would prevent creating entries having the same UUID.

1PW-17-005 WP2: Information disclosure via WebSocket channel *(Low)*

***Note**: The 1Password team has accepted this finding as a low severity issue and is working on a different way to handle the described notification messages that is not susceptible to this.*

It was found that the B5 web application discloses information via the notification feature using WebSockets. Notifications are used to send confirmations back to the client in case data was successfully updated on the server.

Due to a missing check, a valid WebSocket connection can be established without knowing a corresponding UUID of a connected device. This lets adversaries obtain valid session IDs or device UUIDs from other users who have updated an entry. The behavior can also be used to track some activities about a user.

Affected file:
cmd/b5notifier/main.go

Affected code:

```
path := strings.TrimPrefix(r.URL.Path, "/")
params := strings.Split(path, "/")
if len(params) != 3 {
   l.Warning(r, "["+ip+"] Invalid number of parameters:
"+strconv.Itoa(len(params)))
   http.Error(w, "Method not allowed", 405)
```

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

```
   return
}

accountUUID := params[0]
userUUID := params[1]
deviceUUID := params[2]
[...]
```

Affected request:

```
GET
/WVQEGA6WHBGVXCNC4SWFYFRGSU/AR2PKGCWBJAYPKJ2TF7COILNPM/vboioxt4rawx6iy66dg5s7dui
a HTTP/1.1
Host: b5n.b5test.com
Connection: Upgrade
Origin: https://1pw17second.b5test.com
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: G1w+GCmY6ppqtm+Rw55jqQ==
[...]
```

For a successful attack, only a valid account UUID and a user UUID are required, which are both obtainable from the API via a 1Password account. The following PoC shows how such a connection can be established.

PoC:

```
<script>var s = new
WebSocket("wss://b5n.1password.com/<accountUUID>/<userUUID>/");s.onmessage =
function (event) {document.write(event.data)}</script>
```

However, the obtainable data could not be exploited in order to impersonate other users or to access juicy data, thus explaining the *Miscellaneous* rating. This is mainly due to the fact that for a working *X-Agilebits-Mac* header, a valid session key is required and only known by the client.

Nevertheless, it is recommended that neither session IDs nor device UUIDs get broadcasted via WebSockets to the connected clients. Moreover, it is advised to implement an additional check which makes sure that a valid device UUID is required for a valid WebSocket connection. It actually needs to belong to the user UUID.

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

Conclusions

In this *1PW-17* project, which was a next iteration of the ongoing assessments pertaining to the client- and server-side parts of the 1Password B5 web application, Cure53 examined various components through a selection of tailored approaches. To reiterate, three members of the Cure53 team completed the project over the course of fourteen days in October 2021.

While six issues were found during the audit, four of them were exploitable and two were just hardening recommendations or best practices. It must be underlined that the project concludes with a fine result for 1Password. The narrow span and low volume of issues is a clear sign that an external attacker without in-depth insight into the software implementation, codebase and architecture would have had a hard time when wanting to cause noteworthy damage.

The basic idea behind the Cure53 investigation of the web applications was to find out whether the existing functionality of the connected endpoints and its environment can be deemed healthy enough to withstand attacks by malicious users. With the focus on typical modern application problems, the issues connected with various types of injection attacks and misconfigurations, which could compromise the server part of the application, were investigated without significant success.

While most of the tests were performed on the created test-environments, all found issues are also exploitable on the corresponding production environments. This is mainly due to the fact that the offered sources differ only in the configuration settings and basically show no differences in the respective implementations. Still, the security posture of the applications running on production machines was also examined, especially for the newly introduced services used by the item sharing feature. Thus, the applications were checked for exposed sensitive services, insecure configurations, as well for hidden files or routes.

Given long-term cooperation, a lot of attention was given to the newly introduced item, which was here the sharing feature. As a result, three issues connected to *Low* and *Medium* severities could be found in this area during this autumn 2021 review. This corresponds to a solid result for the first review of a newly implemented feature.

To give more details, the first found issue - 1PW-17-001 - affects the client-side parts of the item sharing feature. Items only made accessible to specific email addresses are prone to some local attack scenarios due to the usage of the browser's insecure *localStorage*. In combination with the expiration duration of about one month, the likelihood of it being exploited by local adversaries is heavily increased. It is

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

recommended to follow the proposed mitigations in order to prevent credentials from being compromised.

Moreover, two additional issues connected to the item sharing feature were found. They lie in the *Activity log* component. Due to the implemented design, the server has no data on decrypted data, which is encrypted on the client-side with a key derived from randomly created secret bytes. This allows adversaries to send other UUIDs or encrypted data to the server as it is shown in the logs (see 1PW-17-003). However, since it only allows some spoofing, the issue was rated as *Low*. Nevertheless, it should be taken into account to raise the bar of exploitation, making these attacks less attractive.

A similar issue could be found regarding the creation of duplicate log entries of shared items due to a missing check observed within the sharing flow. However, due to the fact that it only fosters some inconsistencies, no further advantage could be found, hence the *Miscellaneous* rating (see 1PW-17-002).

The server-side implementation of the newly designed invite flow has been analyzed as well. This feature is now directly available in vault settings to give new users access to specific vaults. In this testing process new groups of users have been created. They only have access to one specific vault and exclusively have the permission to invite new users into said vault. The frontend implementation of the UI does not show this, but the logic behind the invite flow iterates over the request body and allows to specify multiple vaults and user IDs to be added. It was made sure that the loops exit correctly whenever vaults are specified but the necessary permissions are missing.

Even though the server sometimes accepts requests to non-accessible vaults, it was determined that this only happens when an invite to an accessible vault is already present; the logic then simply skips all proceeding checks. This does not create any form of vulnerability. Output-validation of the generated emails has been checked as well. Some sinks like the created button URLs for accepting invites exist because of dangerously looking *sprintf* calls. However, no issues transpired.

Another important refreshed feature that was covered in this pentest spanned the DUOv4 implementation that is heavily based on redirects. Since the new flow leaves the application context, it is stored in an encrypted session storage, whereas the key is stored in the *state* parameter. Cure53 made sure that the session storage was correctly cleared on each flow, even when the MFA flow failed in the end. Verification of configurable DUO settings, to - for instance - prevent SSRF attacks via the DUO API hostname, is also being done correctly through extensive use of validators to check for

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

correct domain suffixes. The DUO code verification is also appropriate, with pinned DUO certificates for the final verification code within the server-side MFA API.

A dedicated and requested reanalysis of permission checks for users with access to creating service accounts was performed. Despite problems in the past (i.e. viewing the secret automation as a way to forge JWTs and get access to restricted vaults), Cure53 was unable to reproduce past attack vectors or identify new ones. Similarly, it was not possible to find any further ACL flows through the service account integration over a newly configured connect server. Cure53 found that the permission matrix for vaults works as intended. The testing team checked if previously found issues connected to several DoS attacks were addressed accordingly. While the issues are properly fixed, a new issue was found in this area (see 1PW-17-006). This time, invalid sent data during vault creation allows adversaries to perform a permanent Denial-of-Service attack affecting the *entire* 1Password account. Absence of a proper exception handling of invalid keys means that the web interface cannot be displayed. This leads to a locked account, so a clear recommendation is to first focus on fixing this issue to prevent DoS attacks against 1Password accounts.

Another discovery affected the current notification feature, which was missing some additional checks during the WebSocket creation. This meant adversaries could obtain session IDs and device UUIDs of currently authorized users (see 1PW-17-005). However, Cure53 could not see a way to turn this into a real vulnerability, hence the *Miscellaneous* score. Nevertheless, it is recommended to further follow the principle of minimalism and only send back data that is really required by the connected clients.

All in all, the examined 1Password B5 web application and the related specific areas leave a very good and solid impression. Despite one *High* finding, the average impact of discoveries was limited, thereby indicating quite stable protection against attacks that could imaginably target the examined applications and services. However, the audit also shows that there is still room for improvement, especially in the client-side parts. The tested items somewhat suffered from insufficiently implemented exception handling and the usage of the proven-unsuitable browser features linked to storing sensitive data. Once the relevant issues are fixed, Cure53 will have greater confidence in the 1Password B5 web application being more properly secured for a continuative production use. It is hoped that the listed flaws can increase the compound’s capacity to deliver a secure foundation, at a level expected of password managers.

Cure53 would like to thank Stephen Haywood, Brett Bollman, Jasper Patterson, Whymarrh Whitby, Rob Yoder, Rick van Galen and Artem Kresling from the 1Password team for their excellent project coordination, support and assistance, both before and during this assignment.