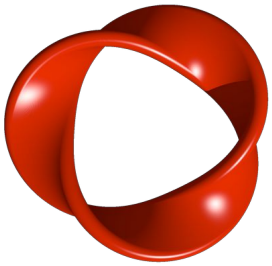

Secfault Security

1Password in the Browser

Security Assessment

Report
FINAL

for

Agilebits Inc dba 1Password

4711 Yonge St., 10th Floor
Toronto,ON M2N 6K8 AgileBits

- hereafter called "Agilebits" -

Document History

Version	Author	Date	Comment
0.1	Leonard König	2022-06-24	First Draft
0.2	Gregor Kopf	2022-06-28	Internal Review
0.3	Leonard König	2022-07-27	Customer Comments
0.4	Dirk Breiden	2022-07-29	Last Customer Comment
1.0	Dirk Breiden	2022-07-29	Create Final Version
1.1	Gregor Kopf	2022-11-03	Included Additional Feedback

Table of Contents

1 Executive Summary

Secfault Security was tasked by Agilebits with a security review of selected parts of the 1Password ecosystem, namely "1Password in the browser" (b5x) along with its accompanying components. The review has been performed in the time frame from 2022-06-07 to 2022-06-24. This document describes the results of the project.

1Password in the browser aims to improve the integration between 1Password and web browsers, so that for instance passwords and other sensitive information can be directly filled from within the browser session. Furthermore, the "Save in 1Password" API enables selected web sites to allow users to directly store confidential information into a 1Password vault.

During the review two issues, which are described in detail in section 4 of this document, have been identified. Section 5 of this document provides additional observations and recommendations for components that were not in scope of the review.

Overall, the reviewed codebase left a positive impression. The code is well-structured and readable and a large number of common possible security issues have been avoided. This indicates that the code has been implemented with security in mind.

After having received a draft version of this document, Agilebits provided feedback on the identified issues, which can be found in section 6 of this document.

2 Overview

1Password is a password manager product developed and maintained by AgileBits Inc. The solution provides a secure place for customers to store various passwords, software licenses, and other sensitive information in virtual vaults.

Agilebits tasked Secfault Security with a review of the "1Password in the browser" feature as part of their product suite, with its optional feature to share its lock state with the native app, removing the need for duplicate password entry to unlock different components of 1Password.

Further, the "Save in 1Password" API to communicate with the aforementioned 1Password in the browser extension, alongside with an NPM package helping to implement a button consuming this API was part of the project.

In section 2.1.1 of this document, a description of the project's scope is provided. Section 2.1.2 provides details on the test procedures.

2.1.1 Target Scope

The following source code archives have been provided by Agilebits for review:

- `core` at revision release-stable-2.3.5

Furthermore, Agilebits provided the respective installation packages for 1Password in the browser at the above mentioned revision, as well as links for downloads to the official 1Password desktop client and the NPM package for the "Save in 1Password" API.

2.1.2 Test Procedures

The overall project followed a white-box approach, which means that Agilebits provided the source code, the compiled binaries and technical documentation for the solution. Therefore, the solution has been analyzed by performing a source code review, as well as targeted dynamic testing.

The source code review has been performed in a manual fashion, i.e., without relying on automated vulnerability scanners or similar tools. Besides identifying possible classical implementation weaknesses, one main focus of the review was the identification of potential logic problems. This requires an in-depth understanding of the solution's inner workings, which is best achieved by a manual process.

The dynamic tests have been performed in a targeted fashion. On the one hand, this served the purpose of validating issues identified during the source code review. On the other hand, dynamic tests were also performed to obtain a better understanding of the overall solution and the interplay of its individual components.

2.1.2.1 Native Messaging Host

The NMH implementation has been reviewed, focusing on various types of implementation and/or logical issues. On the one hand, the overall communication scheme between the NMH and the 1Password application has been subject to review. On the other hand, the communication between the browser and the NMH has been analyzed.

This led to the identification of the issue described in section 4.1 of this document.

Furthermore, checks have been performed on the implementations of the communication protocols (e.g., the JSON parsing for the NMH communication), as well as on the general application logic. However, no issues have been identified in this area.

2.1.2.2 1Password in the Browser

1Password in the browser has been reviewed, focusing on different types of possible vulnerabilities. These included the following:

- XSS or similar issues
- Missing or bad verification of the website URIs (so that malicious websites could obtain passwords they are not entitled to)
- Missing or bad isolation of the extension background scripts and content scripts from websites
- Usage of untrusted information from possibly malicious websites' DOMs
- Autofill logic issues

2.1.2.3 Save in 1Password API

The provided code has been subject to review, focusing on classical implementation issues and possible logic problems, including:

- XSS or similar issues
- Vulnerabilities in the encryption scheme used for confidential information
- General information-leakage issues
- General logic issues when adding information to the user's vaults

2.2 Project Execution

The project has been executed in the time frame from 2022-06-07 to 2022-06-24 in seven person days.

The consultants assigned to this projects were:

- Maik Münch
- Leonard König

- Gregor Kopf

3 Result Overview

An overview of the project results is provided in the following table.

Description	Chapter	Type	Severity
Browser Verification Bypass (Linux)	4.1	Design	Medium
Compromised Shared Lock State Leads to Full Data Access	4.2	Design	Medium

Each identified issue is briefly described by its title, its type, its exploitability and by the impact of a successful exploitation. Technical details for the individual issues are provided in the respective sections of chapter 4 of this document. Details regarding the vulnerability rating scheme used in this document are provided in section 7.

4 Results

The issues identified during the project are described in detail in the following sections. For each finding, there is a technical description, recommended actions and - if necessary and possible - reproduction steps. For details regarding the used vulnerability rating scheme, please refer to section 7 of this document.

4.1 Browser Verification Bypass (Linux)

Summary

Type	Location	Severity
Design	Browser Verification	Medium

Technical Description

1Password in the browser can optionally communicate with the 1Password desktop application through a Native Messaging Host (NMH). This is used to launch the 1Password desktop window when creating a new password from the browser instead of from within the extension, if the feature is activated. Furthermore, it allows 1Password in the browser to be automatically logged in with the user's credentials when the 1Password desktop application is logged in, removing the need for additional input of the password.

To securely implement this functionality, the 1Password desktop application, which provides both the actual application logic as well as the NMH, must verify that the application requesting the login credentials is indeed 1Password in the browser.

In order to do so, the 1Password desktop app verifies that the process it is communicating with via a UNIX domain socket, is indeed the NMH or at least carrying the privileges of the `onepassword` group; and the NMH binary verifies that the immediate parent process is installed by the root user and from a list of trusted applications (since `/bin/sh` is installed by root but can be used to launch arbitrary code).

However, an attacker may launch a "trusted" application such as Chrome in a modified environment, using e.g., an attached debugger or `LD_PRELOAD` to modify its execution environment. This way, the trusted application is compromised and can be used to launch the NMH while being under an attacker's control. One way to exploit this is to send a `NmRequestAccounts` request to 1Password Desktop which will then print out the secrets needed to login to 1Password without needing to enter a password, as 1Password in the browser does.

This will allow an attacker to make use of the "Shared Lock State" and leads to a full compromise

of the data held in this state. In order to investigate and demonstrate this attack, a PoC was developed, based on this finding.

Recommended Action

Trusting a (parent) process based on its properties rather than cryptographic attestation is not compatible with the classic POSIX/UNIX process model. While client processes are under some control by their parent (execution environment, privileges), parent processes are generally difficult to inspect from the child's perspective. A framework to provide such trust would live outside of the UNIX process model, and may consist of, e.g., additional system calls for attestation, lockdown and anti-tamper, using the OS kernel to verify the user space processes.

That being said, one option for mitigating the risk might be to educate users about the fact that in presence of local attackers, the confidentiality of their vault information cannot be fully guaranteed. Furthermore, please also refer to the finding described in section 4.2, which discusses additional measures that could help reduce the emerging risks.

Reproduction Steps

The provided `exec-hook.so` file can be preloaded into a trusted application's process image; it will override the `exec(2)` family of functions and use the `dlfcn.h` API to pass their arguments to the original C library instances of these functions, logging the calls in-between. Further, it will kill its "host" trusted application's process, and launch the NMH from within its context. Please note that the launched NMH executable is unchanged, and its `setgid` bit allows the hooked process to use the privileges of the group of its owner.

```
$ LD_PRELOAD=$PWD/exec-hook.so /usr/lib/chromium/chromium
$ ps aux | grep op-browser-support
ljrk       59350  0.0  0.0 624284  8684 ?        Sl   17:47   0:00
/usr/local/bin/op-browser-support
chrome-extension://hjlinigoblmkhjejkmbegnoaljkphmgo/
ljrk       59364  0.0  0.0   7040  2804 pts/5    S+   17:47   0:00 grep --
color=auto op-browser-support
```

Using the process ID of the NMH, one can look up the process ID of the parent and use that to retrieve the parent processes' executable in `procfs`:

```
$ cut -d' ' -f4 /proc/59350/stat
59348
$ ls -l /proc/59348/exe
lrwxrwxrwx 1 ljrk ljrk 0  9. Jun 17:47 /proc/59348/exe ->
/usr/lib/chromium/chromium
```

This illustrates that the parent executable of the NMH is indeed Chromium.

Finally, one can check the logs to see whether the NMH indeed verifies its parent process to be a valid browser:


```
$ cat $XDG_CONFIG_HOME/1Password/BrowserSupport/1Password_rCURRENT.log
INFO  2022-06-10T09:45:04.864 main(ThreadId(1)) [1P:native-messaging/op-
browser-support/src/main.rs:163] Starting 1Password-BrowserSupport 8.7.0-
28.BETA debug build no. 999999999.
INFO  2022-06-10T09:45:04.866 main(ThreadId(1)) [1P:native-messaging/op-
browser-support/src/browser_verification/linux.rs:40] Verifying browser
"/usr/lib/chromium/chromium"
INFO  2022-06-10T09:45:04.867 main(ThreadId(1)) [1P:native-messaging/op-
browser-support/src/browser_verification/linux.rs:54] Browser
"/usr/lib/chromium/chromium" verified successfully
INFO  2022-06-10T09:45:04.867 main(ThreadId(1)) [1P:native-messaging/op-
browser-support-lib/src/communication_logic.rs:122] Starting SLS communication
(attempting connection to desktop app)
INFO  2022-06-10T09:45:04.867 main(ThreadId(1)) [1P:native-messaging/op-
browser-support-lib/src/communication_logic.rs:194] Desktop app not running, no
connection established
```

In order to illustrate the messages exchanged between the NMH and the desktop application, a small proxy has been developed. This proxy acts as a NMH towards the browser, and uses the actual NMH binary to communicate with the desktop app. The proxy source code has been delivered to Agilebits separately from this report.

In order to use this proxy, the 1Password app must be launched first, as it will install/override the NMH manifest. This manifest, which resides at e.g., `~/.config/google-chrome/NativeMessagingHosts/com.1password.1password.json` is then modified to point to the path of the `nmh_proxy` application attached.

When the NMH is launched by the browser, the proxy will launch the original NMH as its own child and intercepts the communication between the browser / 1Password in the browser and the 1Password application through the original NMH, logging data sent from the former to `/tmp/stdin.log`, and data sent from the latter to `/tmp/stdout.log`.

Each request/response sent by these means is prefixed with a 32 bit length, followed by a JSON object of that size in bytes. This may contain e.g., the following:

```
{
 "type": "Success",
 "content": {
   "callbackId": 3963783914,
   "response": {
     "type": "NmRequestAccounts",
     "content": {
       "accounts": [
         {
           "type": "Unlocked",
           "content": {
             "details": {
```

```
              "accountUuid": "KNVF3T2FEJBMVGZ2DEL4AMBKHE",
              "accountName": "1Password / SecFault",
              "userVersion": 3,
              "email": "leo@secfault-security.com",
              "server": "1password.com",
              "signInAddress": "https://1passwordsecfault.1password.com/",
              "userUuid": "JLTVO6K7ZFFFHAVXXV2W2T7ILA"
            },
            "secretKey": "RE-DACTED-REDACT-EDRED-ACTED-REDAC-TEDRE",
            "muk": {
              "kty": "oct",
              "kid": "mp",
              "alg": "A256GCM",
              "k": "r3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr3d",
              "ext": true,
              "key_ops": [
                "encrypt",
                "decrypt"
              ]
            },
            "srpX":
"r3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr3d4ct3dr"
          }
        }
      ],
      "authorizable": true
    }
  }
 }
}
```


Please observe that this JSON object contains the user login data and credentials such as `secretKey`, `muk.k` and `srpX`.

4.2 Compromised Shared Lock State Leads to Full Data Access

Summary

Type	Location	Severity
Design	Shared Lock State	Medium

Technical Description

As described in the finding 4.1, an attacker may circumvent the browser verification in order to impersonate the legitimate Native Messaging Host (NMH) to the 1Password Desktop application.

As outlined in that finding, this will lead to being able to read the full Shared Lock State, containing all required keys to decrypt and access the passwords stored in the vault. This behavior is by design, and can likely not easily be changed. However, apart from the verification of the browser by the NMH binary, there currently is no second line of defense. It might therefore be worthwhile to consider introducing a stricter scheme for accessing passwords via the NMH. This issue serves the purpose of raising awareness for such considerations.

Recommended Action

One option for mitigating the risk of exposing confidential information via the NMH could be to obtain passwords directly from the desktop app. 1Password in the browser could be required to query the desktop app for each password individually; subsequently, the desktop app could confirm with the user, whether accessing the respective password is intended.

Such a confirmation could also be made optional, in case the user decides to accept the emerging risks.

Reproduction Steps

Please refer to finding 4.1 for a PoC MITM attack dumping all the secrets necessary to share the lock state with the desktop application.

5 Additional Observations

Secfault Security would like to point out a number of general observations and recommendations regarding the analyzed system in the following subsections.

5.1 Unhandled Error Returns in Domain Punycode Conversion

The file `op-domain/src/lib.rs` provides functions such as `naked_domain_for_host` which uses the `idna` crate to convert punycode encoded domains to unicode, checking for erroneously encoded punycode domains:

```
let (unicode_naked_domain, res) = idna::domain_to_unicode(ascii_naked_domain);
res.ok()?;
```

However, the code in e.g., `op-ui/src/view_item/website.rs` does not check the error result of the function in the same way:

```
let unicode_domain = idna::domain_to_unicode(host).0;
```

Since `domain_to_unicode` is guaranteed to always return a valid string, e.g., `host = "xn--foo-a.de"` would result in `unicode_domain` being `foo.de`.

A potential check for equality of those two domains would thus evaluate to true, and may cause issues when matching against domain names.

6 Customer Feedback

After receiving a draft version of this document, Agilebits reviewed the identified issues and provided feedback, describing their assessment. In order to provide full transparency, this feedback is included in the below sections.

6.1 Browser Verification Bypass (Linux) (Finding 4.1)

We've reviewed this issue and reviewed the provided recommended action options. Of the options presented, we currently mitigate the risk through user education provided in the 1Password Security Design Whitepaper. The technical option within the recommended actions is not viable due to the potential for application failure in many valid Linux setups. Refer to sections "Malicious client" and "Malicious processes on your devices" within the 1Password Security Design Whitepaper for more information, potential risks, and threats.

6.2 Compromised Shared Lock State Leads to Full Data Access (Finding 4.2)

We've reviewed this issue and it requires the finding in 4.1 to be exploited prior to being able to access/compromise the data. This is addressed in the 1Password Security Design Whitepaper.

6.3 Unhandled Error Returns in Domain Punycode Conversion (Finding 5.1)

We've accepted this finding as a best practice issue. In the future, 1Password will investigate any necessary changes to the implementation.

7 Vulnerability Rating

This section provides a description of the vulnerability rating scheme used in this document. Each finding is rated by its type and its severity. The meaning of the individual ratings are provided in the following sub-sections.

7.1 Vulnerability Types

Vulnerabilities are rated by the types described in the following table.

Type	Description
Configuration	The finding is a configuration issue
Design	The finding is the result of a design decision
Code	The finding is caused by a coding mistake
Observation	The finding is an observation, which does not necessarily have a direct impact

7.2 Severity

The severity of a vulnerability describes a combination of the likelihood of attackers exploiting the vulnerability, and the impact of a successful exploitation.

Severity Rating	Description
Not Exploitable	This finding can most likely not be exploited.
Low	The vulnerability is either hard to exploit (e.g., because a successful exploitation requires significant prerequisites) or its consequences can be considered benign.
Medium	The vulnerability can be exploited (possibly under certain preconditions) and a successful exploit can be used to at least partially bypass the security guarantees of the solution.
High	The vulnerability can be exploited easily and a successful exploit bypasses one of the core security properties of the solution.
Critical	The vulnerability can be exploited easily and a successful exploit can be used to compromise systems beyond the scope of the analysis.

8 Glossary

Term	Definition
API	Application Programming Interface
ID	Identification
JSON	JavaScript Object Notation
MITM	Man-In-The-Middle
NMH	Native Messaging Host
OS	Operating System
PoC	Proof-of-Concept
XSS	Cross Site Scripting