1Password CLI Service Accounts Security Assessment

Project No. 378.2202
Report
FINAL

for

Agilebits Inc dba 1Password
4711 Yonge St., 10th Floor
Toronto, ON M2N 6K8
Canada

Document Versions and Changes

Version	Author	Date	Comment
0.1	Micaela Ranea-Sánchez	2022-07-14	Initial draft
0.2	Carsten Wentzlow	2022-07-20	Additions
0.3	John Lißke	2022-07-20	Technical review
0.4	Micaela Ranea-Sánchez	2022-07-20	Clarifications
0.5	Nico Lindner	2022-07-21	Editorial review
0.6	Micaela Ranea-Sánchez	2022-07-21	Clarifications
0.7	Nico Lindner	2022-07-21	Review
0.8	Micaela Ranea-Sánchez	2022-07-27	Incorporating feedback[1]
0.9	Nico Lindner	2022-07-27	Review
0.10	Micaela Ranea-Sánchez	2022-07-27	Incorporating feedback[2]
1.0	Nico Lindner	2022-07-29	Final version based on v0.10[3]
1.1	Nico Lindner	2022-10-14	Incorporating feedback[4]
1.2	Nico Lindner	2022-10-14	Final version based on v1.1

1 solely wording-related changes
2 solely wording-related changes
3 solely finalizing- and format-related changes, no changes in technical content
4 Minor changes to wording in non-technical chapters. Removal of the Appendix section (example of JSON payload) and references to it, for non-disclosure of sensitive data.

Table of Contents

Terms and Definitions

Term	Definition
API	Application Programming Interface
CLI	Command-line Interface
DoS	Denial-of-Service
HTTP	Hyper Text Transfer Protocol
ID	Identification

1 Executive Summary

Recurity Labs was tasked by AgileBits to perform a time-boxed security assessment of the newly implemented *Service Accounts* feature, added to the *1Password* password manager.

This assessment was performed in the context of the regular review policy followed by AgileBits. In conformance with this policy, the application has been audited by multiple other security consultancy teams in the course of the last years, something clearly noticeable when assessing the components of the solution. In a previous testing cycle, Recurity Labs was tasked with assessing the *1Password Server API* in itself[5]. Please also refer to chapter 2.1. In the present assessment, the scope was placed on the *Service Accounts* and their integration both in the *1Password Server API*, *1Password Web application* (publicly available at the domain `1password.com`) and *1Password CLI*, the `Go` -based command-line client of the password manager[6].

Summarizing the assessment, the *Service Accounts* feature was found to be in good condition, from a security standpoint. No critical issues could be identified, and only one issue with unknown security impact (rated *Informational*) has been identified. Specifically, during the assessment, the observation was made that the server issues `HTTP Error 500` responses for some invalid API calls. The root cause of these errors is unknown, also, since the conducted source code analysis did not reveal any missing error handling. Further technical information can be found in section 3.1, including reproduction steps and a recommendation for further analysis.

Additional notes on the areas covered and their high-level results can be obtained from section 2.4.

In conclusion, Recurity Labs wants to mention that the overall code quality was found to be remarkably good, and structurally sound.

Remark

In this document, "Service Accounts" is used when referring to the feature, and "service accounts" as the actual account(s) of type service.

1.1 Table of Findings

The following table summarizes the findings Recurity Labs made during the assessment. Each finding is briefly described by its title, its type as well as the effort and impact of a successful exploitation. Technical details for the individual findings are provided in the respective sections of chapter 3 of this document. Details regarding Recurity Labs' rating scheme are provided in section 4.

ID	Description	Chapter	Effort	Impact
378.2202.1	Improper Error Handling	3.1	N/A	Informational

5 https://bucket.agilebits.com/security/378.2101-Recurity_Labs-Report-B5-v1.0.pdf
6 https://1password.com/downloads/command-line/

2 Project Background

Recurity Labs was tasked by AgileBits to perform a time-boxed security assessment of the newly implemented *Service Accounts* feature, added to the *1Password* password manager. Service accounts are intended to help developers build tools to automate secrets management within applications and infrastructure.

2.1 Project History

The current iteration is based on the following previously executed assessments:

Project ID	Title	Date	Report
378.2101	B5 Security Assessment	March 2022	378.2101-Recurity_Labs-Report-B5-v1.0.pdf

2.2 Team

The assessment was conducted between July 04th and 15th in 2022 by Carsten Wentzlow and Micaela Ranea-Sánchez of Recurity Labs. Support was provided by the responsible personnel at AgileBits throughout the project, and Florian Grunert of Recurity Labs as the responsible project manager.

2.3 Analyzed System

The assessment was performed against a test environment available at the domain `b5test.com`.

Multiple user accounts were created by the team, using the following email addresses (provided here to ease the clean-up process at AgileBits):

- `carsten1@recurity-labs.com`
- `mica1@recurity-labs.com` to `mica4@recurity-labs.com`

A shared vault was created by AgileBits, available at `1passwordrecuritylabs.1password.com`. The following documents were provided:

Name	Content	SHA256
`b5-main.tar.gz`	*1Password Server API* and Web application source code	`f6ef03723c53c0c29d6ecdb8e52669cf83ad8a666db5557ff078165c7a4bef25`
`op-binaries.zip`	*1Password CLI* binaries for different architectures	`19420e60f58eaa61d31e3e93e1a05b2b4c7b3b19e0a00ed82d5a82ec1693ea37`
`op.zip`	*1Password CLI* source code	`26dbb8f2edebf979a9b933820b2932732c9b1c035d34792f804f7c397e440f33`
`[RFD 00012.1] B5 Service Accounts Revised.pdf`	Service Accounts RFD	`246a30474147898d05dfed93919b4f2a062cb2e59d5567139b3d37de99600373`
`Service_Account_Security_Documentation_.pdf`	Service Accounts security documentation	`fcfb36d895ed18f1cb79ddd9fdc10cc41e50982284ea44fd771596811af5b05d`

The *1Password Server API* and Web application source code utilized for this assessment was extracted from the `b5-main.tar.gz` file, and, as stated by AgileBits in the file notes, corresponds to the commit hash `c3d1f8cf24602a4f50ab6dff230516a405291947`.

The following output from the tool `scc`[7] provides a high-level overview of said source code utilized for the audit:

Language	Files	Lines	Blanks	Comments	Code
Go	18108	5950910	546914	816867	4587129
TypeScript	2537	374440	37805	16502	320133
Go Template	1057	102473	8771	941	92761
Markdown	872	94477	25781	0	68696
License	496	34243	6011	0	28232
SVG	492	789	29	1	759
Sass	441	39932	6784	440	32708
JSON	408	89865	133	0	89732
YAML	311	37558	2177	1288	34093
SQL	287	13031	1875	761	10395
Assembly	232	32728	7151	0	25577
gitignore	227	2849	502	360	1987
Shell	199	17685	1974	1609	14102
Makefile	158	8298	1645	464	6189
JavaScript	113	8515	640	1386	6489
Plain Text	101	43262	3100	0	40162
BASH	34	1639	206	174	1259
CSV	23	482	12	0	470
Dockerfile	19	427	84	33	310
Protocol Buffers	19	2775	476	288	2011
TypeScript Typings	13	666	87	57	522
XML	11	14391	59	227	14105
C	10	448	94	86	268
Systemd	10	229	20	0	209
TOML	10	403	59	56	288
Gherkin Specificati…	9	451	92	2	357
HTML	7	454	22	8	424
Gemfile	6	18	6	0	12
Python	5	895	128	69	698
CSS	4	29	4	2	23
Docker ignore	4	22	0	0	22
Bazel	3	129	10	0	119
C Header	3	977	79	670	228
Powershell	3	33	8	18	7
AWK	2	32	8	6	18
Happy	2	4350	295	0	4055
Vim Script	2	2	0	0	2
Objective C	1	67	13	7	47
Total	26239	6879974	653054	842322	5384598

A Slack channel was provided by AgileBits to ensure an efficient communication between the consultants and the team.

In addition to the documents mentioned above, the public documentation available at `1password.com` was utilized.

Furthermore, Recurity Labs utilized a number of documents from the previous testing cycle (see section 2.1) to aid in the present assessment:

- a *1Password Server API* overview, based on the OpenAPI[8] standard
- a short tutorial video on how to manipulate requests
- a previous build of the *1Password CLI* tool, which allowed to send custom requests to the server

7 https://github.com/boyter/scc
8 https://www.openapis.org

2.4 Procedures

The assessment was performed between July 04th and 15th in 2022, in a total of 20 person-days.

A mixed approach was followed, where both a full source code review and dynamic testing were performed, with the main objective to uncover weaknesses and vulnerabilities within the integration of *Service Accounts* into the previously existing *1Password Server API*, *1Password Web application* and *CLI* solutions. As typical for such Web-based assessments, OWASP's Top 10 Web Application Security Risk[9] document was utilized as a general guideline to identify common vulnerabilities. In the context of *Service Accounts*, focus was placed upon, but not limited to, the following categories:

- Access Control
- General injection
- Hardcoded credentials
- Insecure configurations
- Insecure random number generation
- Logical flaws
- Sensitive data exposure
- Session management

Furthermore, AgileBits provided a document stating the main *Areas of Concern*, as follows (formatted by Recurity Labs for clarity):

```
- Can users successfully create and use tokens that go beyond their own privileges?
```

Via source code and dynamic testing no further access to other vaults was possible. In the present assessment, no weaknesses were identified which could be utilized to forge tokens.

```
- Can users continue to use revoke tokens?
```

Via dynamic testing, it was verified that existing tokens for service accounts could not be utilized after revokation.

```
- Can non admin/owner users, gain access to managing service accounts? Only admin and
owner users should be able to generate a service account
```

Via dynamic testing, only admin or owner users were found to be able to manage service accounts.

```
- Service accounts cannot access vaults that have disabled service accounts.
```

Dynamic testing showed that service accounts cannot access vaults with disabled service accounts.

```
- Can non admins and owners use the CLI to circumvent permissions on 1Password in order
to manage service account (create, read, delete)?
```

During the assessment, it was not possible to circumvent permissions on *1Password*.

9 https://owasp.org/Top10/

3 Findings in Detail

This section provides technical details on the findings Recurity Labs made during the security assessment. Each finding is described by its title, its type, effort and impact of exploitation. For details regarding Recurity Labs' rating scheme, please refer to section 4 of this document.

3.1 Improper Error Handling

Overview

ID	378.2202.1	
Type	Observation	
Effort/Impact	N/A	Informational
Location	Service Accounts API	

Details

During the assessment, it was identified that the server issues `HTTP Error 500` responses for some invalid API calls. These errors result from conditions, such as:

- when a user attempts to create a service account with an already existing `UUID` and `email` created by another user
- when a user attempts to create a service account with a keyset and preflight token generated by another user

Please note that more instances may exist, where these type of improperly handled errors occur in the API. The cases mentioned above shall merely represent two examples identified during the dynamic portion of the assessment.

As no (direct) security consequences have been identified in the context of the present assessment, this finding has been marked as having an *Informational* impact.

Reproduction Steps

In order to reproduce the present finding, a *1Password CLI* binary supporting the `request` command and two admin accounts in the same team were utilized. In order to use the accounts with *1Password CLI*, these must be added to it first. For detailed instructions to add accounts, the help option can be utilized as follows:

```
./op-internal-linux account add -h
```

The following steps can be considered to reproduce the present finding:

- Log-in to an existing *1Password* account via a modern Web browser, such as Mozilla Firefox, for example
- Navigate to *Service Accounts* on the menu in the right side of the screen, then *Directory* on the top, and finally click on *Create Service Account* to start the process for creating a service account
- In order to view the content of the request sent to the *Service Account* API, it is necessary to overwrite the encryption function utilized by the client before sending the request. For this purpose, open the browser's developer console, and paste the following `JavaScript` code:

```
window.crypto.subtle.encryptOrig = window.crypto.subtle.encrypt;
window.crypto.subtle.encrypt = function(alg, key, data) {
    const req = new TextDecoder().decode(data);
    console.log(req);
    return window.crypto.subtle.encryptOrig(alg, key, data);
}
```

- Now, follow the steps described by the user interface to create the service account
- In the developer console, an entry such as `POST /api/v3/serviceaccounts` should appear, and above it, a `JSON` object
- Copy the `JSON` object
- In a shell, log-in to a different account in the same team using *1Password CLI*, as follows, replacing `<ACCOUNT_ALIAS>` with the account alias chosen when setting-up the account in the tool:

```
eval $(./op-internal-linux signin --account <ACCOUNT_ALIAS>)
```

- Execute the following *1Password CLI* command, replacing the `<PAYLOAD>` with the `JSON` object obtained from the browser:

```
./op-internal-linux request POST /api/v3/serviceaccounts '<PAYLOAD>'
```

- Observe the server response as follows:

```
Error response from B5: Server: (500) (Internal Server Error), Server experienced a
fatal error.
```

Recommendation

HTTP Error 500 codes should be evaluated in-depth to identify their root cause. Recurity Labs recommends the server to handle improper API calls accordingly, and issue appropriate error messages in response(s).

4 Explanation of Classification

This section provides a description of Recurity Labs' vulnerability rating scheme. Each finding is rated by its title, its type as well as the effort and impact of a successful exploitation. The meaning of the individual ratings is provided in the following sub-sections.

4.1 Type

The type of the result is explained in the following table:

Rating	Description
Configuration	The finding is a misconfiguration resulting in security issues.
Design	The finding is the result of a design flaw.
Code	The finding is a flaw in the source code of the object in-scope.
Observation	The finding is an observation reported for the sake of completeness.

4.2 Effort

The effort classification represents both knowledge and skills of a potential attacker as well as the availability of tools and technical resources. Here, the maximum of all three requirements is decisive.

Rating	Description
Extensive	The attack is only feasible with extremely high capabilities. The attack can most likely be performed by federal and multinational attackers only.
High	The attack can only be performed effectively by specialists within several months. In single cases, lower efforts are possible.
Medium	The attack can be performed effectively by specialists within several weeks. In single cases, lower efforts are possible.
Low	The attack can be performed by skilled attackers instantly and requires no further arrangements.
Trivial	The attack is already automated or can be performed with standard tools. Further special skills are not required.

4.3 Impact

The impact always depends on the actual object of research and is not comparable to impacts discussed in other documents.

Rating	Description
Critical	The vulnerability is a systemic error or permits compromising the system completely and beyond the scope of the assessment.
High	The vulnerability permits compromising the systems in-scope completely.
Medium	The vulnerability does not exceed security rules but permits the enumeration of other systems or enables to DoS them.
Low	The vulnerability has no direct security consequences but provides information that can be used for or aid in subsequent attacks.
Informational	The vulnerability provides data about internal processes within the system in-scope or can be used to obtain further information about the system.