Recurity Labs GmbH
https://www.recurity-labs.com

1Password Events Reporting API Security Assessment

Project No. 378.2205
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	Johan Rydberg Möller	2022-12-27	Initial draft
0.2	Sascha Schirra	2022-12-28	Technical review
0.3	Sascha Schirra	2023-01-04	Ratings converted to CVSS
0.4	Nico Lindner	2023-01-06	Editorial review
0.5	Nico Lindner	2023-03-13	Incorporating feedback of AgileBits
1.0	Nico Lindner	2023-03-13	Final version
1.1	Nico Lindner	2023-03-14	Correcting a typo in chapter 3.1

Table of Contents

Terms and Definitions

Term	Definition
API	Application Programming Interface
CLI	Command Line Interface
ID	Identification
IP	Internet Protocol
JWT	JSON Web Token
OWASP	Open Web Application Security Project
VPN	Virtual Private Network

1 Executive Summary

Recurity Labs was tasked to perform a security assessment of the 1Password Events Reporting API, a feature which allows admins to retrieve reports about 1Password activity via an API directly. The events currently supported are sign-in attempts, item usage and audit events.

The 1Password Events Reporting API has been assessed in previous assessments. In scope of the current security review were the new auditevents feature, and IP and JWT-based request throttling, for which testers were given access to a test account and were instructed on how to create the `Bearer` tokens needed to query the API. Testers were also given access to source code for the solution. With this, the assessment included both dynamic and automated testing as well as a cursory source code review.

The 1Password Events Reporting API was found to be robust and only one security-relevant discovery was made during testing. This finding concerns potentially dangerous behaviour related to the handling of networking headers, which should be investigated further. Attempts were made to abuse this behaviour in order to bypass request throttling, but the results were negative. For this reason, the finding is classified as *Informational* at this time. The full details of this observation can be obtained from chapter 3.1.

1.1 Table of Findings

The following table summarizes the findings Recurity Labs made during the assessment. The individual results were evaluated according to CVSSv3.1[1] on request by AgileBits. The CVSSv3.1 vector used for the calculation can be found in section *Overview* of the respective finding(s), detailed in the sub-chapters of section 3 of this document.

ID	Description	Chapter	CVSS	Severity
378.2205.1	Potentially Dangerous Handling of X-Forwarded-For Header	3.1	N/A	N/A

1.1.1 Qualitative Severity Rating Scale

All CVSS scores can be mapped to the qualitative ratings defined in the table[2] below:

CVSS Score	Rating
0.0	None
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

[1] https://www.first.org/cvss/v3-1/

[2] https://www.first.org/cvss/specification-document, chapter 5

2 Project Background

Recurity Labs was tasked to perform a security assessment of the 1Password Events Reporting API, a feature which allows admins to retrieve reports about 1Password activity via an API directly.

2.1 Team

The test was performed by Johan Rydberg Möller of Recurity Labs in the time period between December 19th and 23rd in 2022. Support was provided by AgileBits whenever requested. Florian Grunert of Recurity Labs served as the responsible project manager.

2.2 Analyzed System

Testers were given access to a test account and environment located at `pentesttempaccount.b5test.com`, and used that application and the Command Line Interface (CLI) to generate the `Bearer` tokens with the necessary permissions and flags to access the 1Password Events Reporting API itself, located at `events.b5test.com`.

Testers were also given access to documentation describing the 1Password Events Reporting API, and a report describing previous security issues.

Testers were also given access to the source code of the Events reporting functionality as part of a larger repository. The relevant code breaks down as follows:

```
github.com/AlDanial/cloc v 1.92  T=4.81 s (385.5 files/s, 124881.9 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Go                            1655          55358          71121         448529
Markdown                        85           2157              0           5868
Assembly                        31           2353            354           5380
JSON                             3              0              0           3679
YAML                            29             92             13           1112
make                            18            209             70            861
Bourne Shell                     4             78            371            601
Protocol Buffers                 1             86            238            406
Cucumber                         9             92              2            369
Razor                            1             32             10            268
C/C++ Header                     1             39            335            111
Bourne Again Shell               2             13              6             78
Dockerfile                       3             17              0             53
SVG                              2             29              0             51
C                                1             10              7             28
JavaScript                       2              0            100             14
TOML                             1              6              0             11
INI                              1              2              0             10
CSV                              3              0              0              7
CSS                              1              1              1              2
vim script                       1              0              0              1
-------------------------------------------------------------------------------
SUM:                          1854          60574          72628         467439
-------------------------------------------------------------------------------
```

2.3 Procedures

The 1Password Events Reporting API has been tested for common OWASP Top 10 vulnerabilities, as well as specific issues related to JWT spoofing and manipulation, data leakage between accounts, authentication bypasses, path traversal issues etc.

The API endpoints `itemusages`, `auditevents`, `signinattempts` and `introspect` have been tested for common security issues using both manual and automated testing.

The throttling mechanism in particular has been tested and found to be effective. Attempts to bypass the restriction by jumping through multiple IPs using a VPN, or by manipulating the JWT Bearer token has also been performed, but were not successful.

During a cursory review of the provided source code, it was discovered that the application is configured to expose the Swagger API service at `/api/swagger`, but only for certain environments. Attempts have been made to circumvent this restriction and access the Swagger feature outside these environments (by manipulating the host and other networking headers), but this was not successful.

Automated testing has also been performed in order to discover possible instances of request smuggling and cache poisoning.

2.3.1 Source Code Review

A source code review of the 1Password Events Reporting API has been performed, though it must not be considered exhaustive given the time boxed nature of this assignment. The review focused on the identification of general coding mistakes and logic flaws that may be difficult to find using dynamic testing, as well as on discovering hidden routes or functionality, reviewing the code responsible for request throttling, and the code parsing the JWT Bearer tokens.

3 Findings in Detail

This section provides technical details on the findings made during this security assessment. Each finding is described and rated according to the following criteria: vulnerability type, CVSSv3.1 base score and CVSSv3.1 vector.

Please note that the finding IDs mentioned in the following chapters are solely meant to be unique. Potential gaps in the numbering scheme of finding IDs do not constitute an error. When providing feedback, please reference the *Finding ID*.

3.1 Potentially Dangerous Handling of X-Forwarded-For Header

Overview

ID	378.2205.1
Type	Observation
CVSS Score	N/A (N/A)
CVSS Metrics	N/A
Location	1Password Events Reporting API

Remark

This finding is purely informational and does not indicate a security risk at this time.

Details

During testing, it was discovered that the application will behave differently depending on what values are sent in the `X-Forwarded-For` header.

For example, changing the value of the header to `localhost`, will result in a *403 Forbidden* error. Changing it to `127.0.0.1`, however, will return the expected result to the API query. This behaviour is explained by reviewing the source code located in the file listed below, which will reject the header if it does not contain a valid IP.

File:

```
b5-main/cmd/b5streamingapi/vendor/go.1password.io/activedefence/pkg/remote/remoteip.go
```

Excerpt:

```
// XForwardedForIPHandler is a middleware that parses the X-Forwarded-For header
// and sets the results in the request Context. See OriginIPFromContext
func XForwardedForIPHandler(next http.Handler) http.Handler {
      return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
             originIP, err := GetIP(r)
             contextOriginIPValue := func() (string, error) {
                    return originIP, err
             }
             ctx := WithOriginIP(r.Context(), contextOriginIPValue)
             r = r.WithContext(ctx)
             next.ServeHTTP(w, r)
      })
}

// GetIP returns the remote IP of a request. It returns an error the IP is invalid
func GetIP(r *http.Request) (string, error) {
      xff := r.Header.Get("X-FORWARDED-FOR")
      if len(xff) > 0 {
             return parseXForwardedFor(xff)
      }
```

```
	ip, _, err := net.SplitHostPort(r.RemoteAddr)
	if err != nil {
		return "", errors.Wrapf(err, "failed to SplitHostPort %s", r.RemoteAddr)
	}
	return ip, nil
}

func parseXForwardedFor(s string) (string, error) {
	if len(s) > maxXForwardedForHeaderLength {
		return "", fmt.Errorf("parseXForwardedFor failed, X-FORWARDED-FOR header
too long")
	}

	ips := strings.Split(s, ",")

	ip1 := strings.TrimSpace(ips[0])

	ip2 := ""
	if len(ips) > 1 {
		ip2 = strings.TrimSpace(ips[1])
	}

	ipLast := strings.TrimSpace(ips[len(ips)-1])

	if ip1 != "" && net.ParseIP(ip1) == nil {
		return "", fmt.Errorf("parseXForwardedFor failed to net.ParseIP IP1 %q",
ip1)
	}

	if ip2 != "" && net.ParseIP(ip2) == nil {
		return "", fmt.Errorf("parseXForwardedFor failed to net.ParseIP IP2 %q",
ip2)
	}

	if net.ParseIP(ipLast) == nil {
		return "", fmt.Errorf("parseXForwardedFor failed to net.ParseIP IPLast %q",
ipLast)
	}

	return ipLast, nil
}
```

The implementation is meaningful and sound, however, the fact that the application will accept IP values in this header originating from data supplied by the end user may point to an underlying security issue, and a deviance from best practices. If the application trusts this header to accurately specify the remote IP address of the client, the IP can be spoofed.

Attempts were made to abuse this behaviour in order to gain access to hidden functionality (the Swagger feature at `/api/swagger` for instance), and to bypass IP throttling, but, in the timeframe of this project, were unsuccessful.

Reproduction Steps

To reproduce this issue, please consider the following steps:

- Use an interception proxy, such as Burp Suite[3]
- Add the `X-Forwarded-For` header to a request, e.g. to `/api/v1/auditevents`
- Change the header value.

Observe the resulting behaviour.

[3] https://portswigger.net/burp

Recommendation

It should be noted that this finding is purely informational and does not indicate a security risk at this time. However it should be investigated further as this behaviour indicates a deviance from best practices and a potential issue, which appears to currently be mitigated by some other factor, but which may become an issue in the future.

Feedback provided by AgileBits (2023-02-22)

```
We have investigated this observation and determined that it is not exploitable bypass
under our current infrastructure.
Our AWS environment is augmenting the header with the true client or proxy IP, and our
throttler is correctly correlating the requests.
```

Comment by Recurity Labs (2023-03-13)

The above comment provided by AgileBits indicates that the potential issue was sufficiently reviewed, but no actual risk was identified by AgileBits.

However, no further in-depth review of this issue has been performed by Recurity Labs.