Project Details:

Project Name: **1Password Application Penetration Test and Code Review Report**

Testing Date: July 16 – July 27, 2018

Customer Name and Address:	Consultant Name and Address:
AgileBits, Inc. 317 Adelaide Street West, Suite 910 Toronto, Ontario, M5V1P9, Canada	**AppSec Consulting, Inc.** 6110 Hellyer Avenue, Suite 100 San Jose, CA 95138

Project Engagement Team

Customer Project Manager: Jeffrey Goldberg | jeff@agilebits.com

Consultant Project Manager: Tracy Tran [redacted]

Consultant Account Manager: Brian Shura [redacted]

Security Consultant: Adam Caudill [redacted]

Executive Summary

In June of 2018, AgileBits contracted AppSec Consulting to perform a penetration test and code review of the 1Password application. 1Password is a cloud-based password vault that allows users to generate and store passwords (and other sensitive information) in an encrypted vault. Users can unlock their vault using a master password and a secret key, both of which are only known to the user and not transmitted to or stored on the 1Password server. The application consists of a server-side service written in Go, web-client (TypeScript), and a variety of other clients that are not part of this assessment.

This document represents the final report for the application penetration test and code review security assessment effort for the 1Password application, as performed by AppSec Consulting, Inc., during the period commencing July 16, 2018 and ending July 27, 2018. This testing was conducted against a non-production environment and includes all functional components as currently offered to a AgileBits customer. This environment has been inspected by AppSec Consulting, Inc. and has been granted to be an accurate and reasonable expectation of the services as provided to customers, including realistic data elements.

After consultation with AgileBits, and a review of additional evidence and procedures, this report has been updated by Adam Caudill on 08/22/2018. Finding #1 has been Closed, as the issue applies only to the testing environments, and does not apply to production deployments. Finding #2 has been updated to clarify the high cost per attempt to attackers. Finding #5 has been updated to clarify that the information is stored, and is available in a different portion of the application.

Objectives

This penetration test and code review was conducted against the 1Password application with the following primary objectives:

- Identify and assess the controls in place to protect against both external and internal threats.
- Identify web application, code and server configuration vulnerabilities that put sensitive information at risk.
- Test the application from the standpoint of unauthorized users attempting to gain access as well as authorized users trying to escalate access.
- Provide a detailed risk analysis and remediation advice for each vulnerability identified.

Methodology and Tools Used

Methodology

AppSec Consulting, Inc. performed the following steps when conducting this web application penetration test and code review.

1. **Preparation** – AppSec Consulting, Inc. verified that it had received the following information from AgileBits in preparation for the penetration test.
    a. Application name
    b. Brief description of the application and its purpose
    c. Starting URL(s) for testing the application and URLs or IP addresses for accessing each web server included in the scope of the penetration test

   d. Two sets of login credentials for each level of access that the application provides. For example, if the application provides User and Admin roles, a total of four sets of login credentials will be required (two Users and two Admins).
   e. Application source code
   f. Time windows for when the automated scanning portion of the penetration test can be run without risk of disrupting other users of the application.
2. **Exploration** – AppSec Consulting, Inc. manually explored the entire application in order to become familiar with the application's functionality, purpose, and the sensitivity of information handled by the application.
3. **Automated Vulnerability Scanning** – High-quality commercial vulnerability scanning tools were used to thoroughly scan the application. This scanning process included:
   a. A non-authenticated scan, simulating a user without any login credentials.
   b. An authenticated scan, simulating a logged-in User and a logged-in Administrator.
   c. A server-level scan was run against all web servers included in the scope of the penetration test for server configuration vulnerabilities.
   d. A manual review and analysis was conducted on all scan results, removing any false positives from the results before presenting them to AgileBits.
4. **Code Review** – The source code for security-critical features of the application was reviewed manually, with a focus on areas that typically carry the most risk – for example, authentication, authorization, session management, cryptography, and payment processing code.
5. **Manual Penetration Testing** – The application was manually tested by experienced web application security professionals using AppSec Consulting, Inc.'s systematic testing process. This manual testing process covers all major aspects of web application security, including:
   a. Authentication
   b. Authorization
   c. Session Management
   d. Input/Output Validation
   e. Configuration
   f. Sensitive Data Handing
   g. Privilege Escalation
   h. Logical Vulnerability Checks
6. **Report Preparation** – AppSec Consulting, Inc. took the results of both the automated and manual penetration testing and code review and compiled a consolidated report, detailing all vulnerabilities uncovered during the testing process along with severity levels and recommendations for how to remediate each vulnerability that was identified.

Tools Used

The following tools were used when conducting this penetration test:

- Burp Suite Professional
- Sublime Text for reviewing and searching source code
- The Firefox web browser
- Qualys SSL Labs
- YAWAST

Scope

Application

The assessment was conducted per the following details:

Engagement Date(s)	*Initial Testing* July 16 – July 27, 2018
Location	Amazon AWS
URL(s)	[redacted]
Web Server	Custom
Application Language	Go, TypeScript
Application Components	Server, Web Client
Database Server	MySQL
Environment	QA Environment
Internet Facing?	Yes

Tests

The application was tested for the following categories of technical vulnerabilities:

Platform-level	Application-level
1. Server misconfigurations 2. Published vulnerabilities 3. Forceful browsing 4. Stealth commanding 5. Buffer overflow vulnerabilities 6. Email input/output issues 7. File upload/download concerns	1. Authentication 2. Authorization 3. Parameter tampering 4. Hidden field manipulation 5. Cross-site scripting 6. Cookie manipulation 7. Permissions escalation 8. Session management

Summary of Findings

After conducting extensive testing against the 1Password application, both manual and automated, six vulnerabilities were identified, one of which has been classified as Medium risk.

The most serious issue found was the use of hardcoded credentials for the MySQL server in the source code repository; this could lead to disclosure of data to unauthorized persons *(Updated 08/22/2018: This issue is Closed, as it was found to only apply to testing environments)*. It was also found that the application does not implement an account lockout mechanism, giving an attacker the ability to try a large number of passwords to gain access to an account.

The remaining four findings are all classified as Best Practice, these issues represent opportunities to strengthen the applications security posture and make it more resilient to certain types of attacks.

The security controls observed in the 1Password application were found to be substantial and unusually impressive; the use of filtering and IP blacklisting on requests that include file types not used by the application or parameters that could be used to perform injection attacks provides a substantial defense against automated attacks. The application also uses automated rate limiting, further complicating automated attacks. This feature works so well, that it was necessary to modify the application source code to disable these features in the local testing environment to allow the use of automated tools.

Highly sensitive areas of the application add a random delay to substantially increase the difficulty of finding and exploiting timing side-channels. When combined with the rate limiting feature, this renders exploiting any timing side channel that may exist as impractical at best.

The application follows all industry best practices in regards to TLS configuration, allowing only TLS versions 1.1 and 1.2, preferring cipher suites that utilize forward secrecy, and using AEAD cipher suites where possible. The application sends recommended security-related HTTP headers, including those that control caching, prevent framing, uses a strong Content Security Policy, and a Strict Transport Security policy, among others.

The application has robust error handling and logging, while revealing only generic error messages that do not provide useful information to an attacker. Code is reasonably, though somewhat sparely, commented.

To provide another layer of security, 1Password uses not only TLS to protect data in transit, but also uses an application-layer encryption scheme, so that even if data going over TLS was exposed, the ability of an attacker to utilize the data is reduced.

The application uses an implementation of Secure Remote Password (SRP) to prevent the server from having access to the user's master password. This greatly reduces the chances that an attacker would be able to access user data, even with access to 1Password's data. It must be noted however, this depends on the integrity of the client-side code hosted on the 1Password server; any event that allowed an attacker to modify this code could result in the user's master password of other data being exposed.