GotRoot?

Software Assurance
Fall 2019
Dr. Gandhi

Robert Ernewein

Scott French

Toussia Minoungou

Casey Schmitz

Bitwarden

- Online password management service
- Completely open source software
- Core services are free
- Universally compatible
- Robust features
- Good reputation for security
- TypeScript/JavaScript

Assurance Claims

The System prevents network eavesdropping during client-server communications. (6|7⚠️)

- System Documentation
- Third-Party Audit / Bug Bounty
- Qualys SSL Report

The System is acceptably secure against login attacks. (4|6⚠️)

- Static Code Analysis
- 2FA Implementation

Assurance Claims

The System prevents unauthorized access to secret data. (6|7⚠️)

- Account Management
- Data Encryption Policies

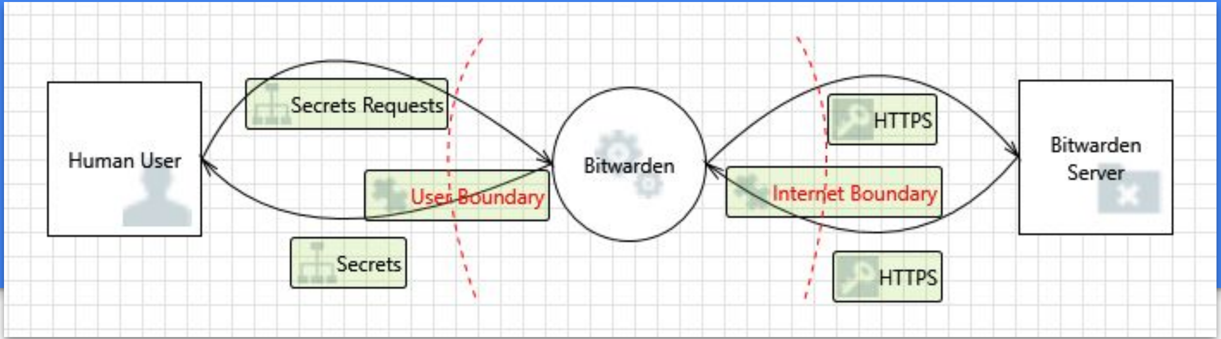
The System adequately limits clear text exposure of user's secret data. (3|5⚠️)

- Sensitive Data Handling
- Asymmetric Encryption
- Security Assessments

The System adequately ensures the availability of secret data. (3|4⚠️)

- Cloud Server Security Policies
- On-Premise Backup

Gaps



- External Entity Human User Potentially Denies Receiving Data
  - No audit log for attempts to sync with the Bitwarden server
- Potential Data Repudiation by the Authentication Module
  - No audit log for authentication attempts in the CLI client
- Spoofing of the Bitwarden Server External Destination Entity
  - Authenticity of the destination server is reliant on client configuration

Findings - Strategy

- Half of the team would focus on automated analysis while the other half would focus on manual analysis
- Since most of the security features are client side we put our focus there, in particular on the command line client

Findings - Manual Analysis

- Combination of code and live analysis
- OWASP Cheat Sheet was a good resource
- Live analysis required significant lab setup
    - Linux gateway was established using sslsplit to act as MITM TLS proxy
    - Bitwarden’s command line app has built in trust store so hex editor was used to replace one of the certs with one created for TLS proxy

Findings - Manual Analysis Results

Password Strength (CWE-521)

- Bitwarden uses a "Master Password" chosen by the end user
- Bitwarden does enforce minimum length of 8 characters
  - NIST SP800-63b - < 8 is considered weak
- Bitwarden appears to have a decent password validation tool but it is only suggestive

Findings - Manual Analysis Results

Password Strength (CWE-521) recommendations:

The master password is a critical component of protecting the end users sensitive data. Bitwarden does enforce a mi has a decent password validation tool that appears to account for length, complexity, and easily guessed passwords. We would like to see it prevent end users from using weak passwords.

Findings - Manual Analysis Results

Password Storage (CWE-916)

- Bitwarden use PBKDF2 for its Key Derivation Function
  - OWASP - Argon2 should be first choice
- Minimum iterations of 5000 is enforced in code
  - NIST SP800-63b - min should be >= 10K
- Bitwarden uses email for key derivation salt
  - OWASP - salt should be cryptographically-strong random data, and minimum of 15 characters long

Findings - Manual Analysis Results

Password Storage (CWE-916) recommendations:

- Support other KDFs such as Argon2. We see evidence in code Bitwarden is already anticipating using multiple KDFs functions in the future, but for now it's only PBKDF2
- Raise minimum iterations to acceptable minimum based on today's hardware
- Use cryptographically-strong salts for KDFs that require them

Findings - Manual Analysis Results

Insufficiently Protected Credentials (CWE 522)

- Bitwarden's command line tool supports passing master password (as well as session ID) as command line argument
  - $ bw unlock myPassword321
  - $ bw list items --session lbSk42dtYFFs.........

Findings - Manual Analysis Results

Insufficiently Protected Credentials (CWE 522)

- Process list can be viewed with all command line arguments by any user on system

```
baduser@sandbox:~$ ps -eo  pid,user,args | grep [b]w
  2612 user     bw unlock myPassword123
```

Findings - Manual Analysis Results

Insufficiently Protected Credentials (CWE 522) recommendations:

- Bitwarden should remove option to pass password as a command line argument and instead rely on passing password interactively

Findings - Automated Analysis

SonarQube SonarScanner

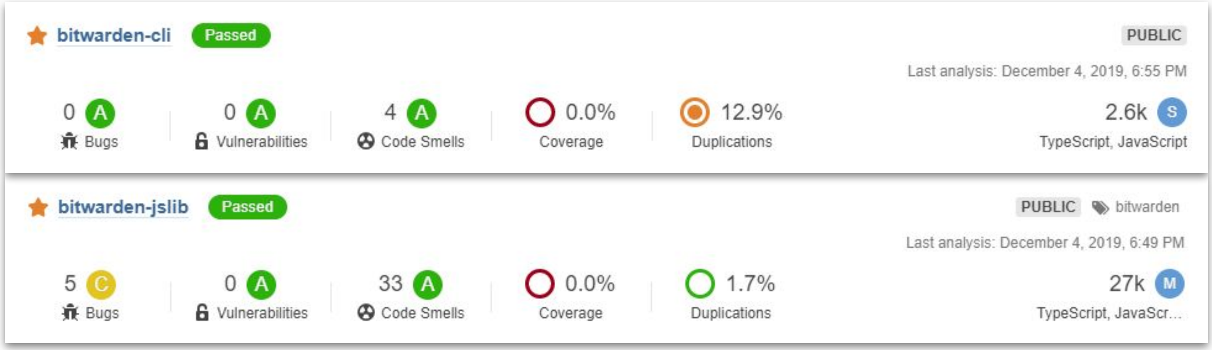
- 350+ Static Analysis rules

Targets

- Command-Line Interface
- Shared TypeScript Library

Measures

- Bugs, Vulnerabilities, Security Hotspots, Code Smells



bitwarden-cli Passed PUBLIC

Last analysis: December 4, 2019, 6:55 PM

Bugs	Vulnerabilities	Code Smells	Coverage	Duplications	TypeScript, JavaScript
0 A	0 A	4 A	0.0%	12.9%	2.6k S

bitwarden-jslib Passed PUBLIC bitwarden

Last analysis: December 4, 2019, 6:49 PM

Bugs	Vulnerabilities	Code Smells	Coverage	Duplications	TypeScript, JavaScr...
5 C	0 A	33 A	0.0%	1.7%	27k M

Findings - Automated Analysis Results

Weak Cryptography (CWE-916) [rule]

- Inferior NodeJS *crypto* library (OWASP)
- Salt + relatively strong legacy encryption algorithms

Regex Denial of Service (CWE-624) [rule]

- Regexes implementing grouping with repetition (OWASP)
- Ex: Email Validation

```
^([a-zA-Z0-9])(([\-.]|[_]+)?([a-zA-Z0-9]+))*(@){1}[a-z0-9]+[.]{1}(([a-z]{2,3})|([
a-z]{2,3}[.]{1}[a-z]{2,3}))$
```

Findings - Automated Analysis Results

Command Injection (CWE-88) [rule]

- Explicit sanitization of CLI input
    - Hardcoded whitelist of command and argument compositions

Contributions

Proposed Design Change: Pre-Hashing the Master Password

I referenced the OWASP cheat sheet for password storage and the pre-hashing of passwords, after reviewing the BitWarden cli/jslib source code, as a means to further secure the master password and subsequent hash and encryption key generation.

The enforcement of strong passwords in v1.8.0 is laxed, and should be addressed. An alternative would be to pre-hash the master password with SHA-256, then iterate the result with a stronger algorithm, such as ARGON2. Pre-hashing provides uniform password lengths, devoid of special characters, and consistent response times regardless of the passwords complexity.

Posted to the BitWarden community forums.

Contributions

Communications/Code Change

There were several posts and developer responses to the hard coding of a 5,000 iteration minimum at both the client and server side. Those issues were closed, and no longer accepting contributions on the subject.

Concurrent Login/Vault Synchronization Issue

While observing the behavior of the local vault (data.json), gaps/omissions were noted in the password history when multiple devices were logged in concurrently. Each key in the vault contains a history of password changes. Using the Command Line Interface (CLI), the vault is downloaded from the server via a forced sync at login. Changes to the vault occur locally, but only sync to the server when commanded or at logout. Therefore, each device is unaware of changes made by the other. Under these conditions, the devices overwrite each others changes, corrupting the vault. Without implementing code changes to allow for dynamic synchronization of local vault instances, the only means to mitigate the issue is to limit concurrent logins to a single device, which is infeasible.

Summary

- Mature open source software that recently had a code audit done by a third party security company with no major findings;
- No coding errors;
- Some standard coding flaws but nothing major;
- In sum, Bitwarden can be trusted by users who are willing to manage some of their secret data, for example their passwords.

References

8 Bit Solutions. (2019). What encryption is being used?. Retrieved from https://help.bitwarden.com/article/what-encryption-is-used/

8 Bit Solutions. (2018). Bitwarden Security Assessment Report [PDF file]. Retrieved from https://cdn.bitwarden.net/misc/Bitwarden%20Security%20Assessment%20Report.pdf

Hackerone. (2019, May 30). Bitwarden: Vulnerability Disclosure Program. Retrieved from https://hackerone.com/bitwarden/

Qualys. (2019). SSL Server Test. Retrieved from https://www.ssllabs.com/ssltest/index.html

8 Bit Solutions. (2019). Create Account. Retrieved from https://vault.bitwarden.com/#/register

References

8 Bit Solutions. (2019). Set up two-step login (2FA). Retrieved from https://help.bitwarden.com/article/setup-two-step-login/

8 Bit Solutions. (2019). I forgot my master password. Retrieved from https://help.bitwarden.com/article/forgot-master-password/

8 Bit Solutions. (2019). How is my data securely transmitted and stored on Bitwarden servers?. Retrieved from https://help.bitwarden.com/article/how-is-data-securely-transmitted-and-stored/

Spearrin, K. (2019, February 22). Any thoughts on this independent security study? Apparently all the major PW managers are insecure. Retrieved from https://community.bitwarden.com/t/any-thoughts-on-this-independent-security-study-apparently-all-the-major-pw-managers-are-insecure/4663/7

References

8 Bit Solutions. (2019). How do you keep the cloud servers secure?. Retrieved from https://help.bitwarden.com/article/cloud-server-security/

8 Bit Solutions. (2019). Backing up your on-premises hosted data. Retrieved from https://help.bitwarden.com/article/backup-on-premise/

MITRE. (2019, June 20). CWE-521: Weak Password Requirements. Retrieved from https://cwe.mitre.org/data/definitions/521.html

NIST. (2017, June). NIST Special Publication 800-63B: Digital Identity Guidelines: Authentication and Lifecycle Management. Retrieved from https://pages.nist.gov/800-63-3/sp800-63b.html

MITRE. (2019, June 20). CWE-916: Use of Password Hash With Insufficient Computational Effort. Retrieved from https://cwe.mitre.org/data/definitions/916.html

References

OWASP. (n.d.). Password Storage Cheat Sheet. Retrieved from https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

MITRE. (2019, June 20). CWE-522: Insufficiently Protected Credentials. Retrieved from https://cwe.mitre.org/data/definitions/522.html

Sonar Source. (2019, December). Hashing data is security-sensitive. Retrieved from https://rules.sonarsource.com/typescript/RSPEC-4790

MITRE. (2018, December 27). CWE-624: Executable Regular Expression Error. Retrieved from https://cwe.mitre.org/data/definitions/624.html

OWASP. (2017, July 5). Regular expression Denial of Service - ReDoS: Evil Regexes. Retrieved from https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS#Evil_Regexes

References

OWASP. (2017, July 5). Regular expression Denial of Service - ReDoS: Examples. Retrieved from https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS#Examples

OWASP. (2018, September 23). CWE-88: Improper Neutralization of Argument Delimiters in a Command ('Argument Injection'). Retrieved from https://cwe.mitre.org/data/definitions/88.html

Sonar Source. (2019, December). Using command line arguments is security-sensitive. Retrieved from https://rules.sonarsource.com/typescript/RSPEC-4823

Ernewein, R. (2019, December 10). Prehashing Passwords. Retrieved from https://community.bitwarden.com/t/pre-hashing-passwords/9329

8 Bit Solutions. (2018, November 8). Bitwarden Security Assessment Report [PDF file]. Retrieved from https://cdn.bitwarden.net/misc/Bitwarden%20Security%20Assessment%20Report%20-%20v2.pdf