Bitwarden Mobile Apps and SDK Report

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

Table of Contents

Summary

In March 2024, Bitwarden engaged with cybersecurity firm Cure53 to perform penetration testing and a dedicated audit of the new Bitwarden mobile apps and SDK. A team of testers from Cure53 were tasked with preparing and executing the audit over two weeks to reach total coverage of the system under review.

Eight issues were discovered during the audit. Four issues were resolved post-assessment. One issue was determined not feasible to address. Three issues are under planning and research.

This report was prepared by the Bitwarden team to cover the scope and impact of the issues found during the assessment and their resolution steps. For completeness and transparency, a copy of the Identified Vulnerabilities section within the report delivered by Cure53 has also been attached to this report.

Issues

BWN-11-001 WP1-2: Deep Link hijacking on bitwarden:// scheme (Low)

Status: Issue under planning and research.

Bitwarden intends to eventually release mobile apps with enhanced deep link capabilities.

BWN-11-005 WP2: Bypassing biometric verification on Android (Medium)

Status: Issue was partially fixed post-assessment, and additional enhancements are under planning and research.

Commits:

- https://github.com/bitwarden/android/commit/f3f35511a42bc4743febb2aeae6dfeb1ec5705af

A gap allowing bypass was patched as part of a move to strong biometrics. Further improvements are being added to use TPM keys for stronger encryption and decryption.

BWN-11-002 WP1-2: Content injection in captcha connector (Low)

Status: Issue under planning and research.

A fixed set of messages will be added and localized within the apps.

BWN-11-003 WP1-2: Lack of root detection and anti-debugging defenses (Low)

Status: Accepted.

Bitwarden supports a multitude of platforms and does not intend to add any warnings on rooted devices.

BWN-11-006 WP3: Master password not cleared from memory (Info)

Status: Issue was fixed post-assessment.

Pull requests:

- https://github.com/bitwarden/sdk/pull/724
- https://github.com/bitwarden/sdk/pull/725

A custom global allocator was added on the SDK that zeroes out any Rust-allocated memory when it's freed. This replaces the old Sensitive pattern, and should handle all possible leaks of the master password from the SDK side.

BWN-11-007 WP3: Unforced unwraps may cause panic (Info)

Status: Issue was fixed post-assessment.

Pull requests:

- https://github.com/bitwarden/sdk/pull/682

Unwraps were replaced with proper error handling.

BWN-11-008 WP3: Outdated Docker image with critical vulnerabilities (Low)

Status: Issue was fixed post-assessment.

Pull requests:

- https://github.com/bitwarden/sdk/pull/681

A distro-less base container image was used to therefore not create any future vulnerability sources.

BWN-11-004 WP1: Keychain data persists after uninstallation (Low)

Status: Issue under planning and research.

Bitwarden intends to eventually release an iOS app with different handling of available Keychain data.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, each ticket has been given a unique identifier (e.g., *BWN-11-001*) to facilitate any future follow-up correspondence.

BWN-11-001 WP1-2: *Deep Link* hijacking on *bitwarden://* scheme *(Low)*

The testing team noted that the application utilizes the *bitwarden:// Deep Link* scheme on both iOS and Android. This approach has been deemed deprecated and insecure.

Deep Links constitute URLs that take users directly to specific content in an application. Whenever users click or redirect to a URL that matches the registered scheme, they will be taken to the activity that handles it. Whilst these can enhance the user experience by providing direct access to specific content within an application, crucial security features are otherwise lacking.

Furthermore, *Deep Links* remain an attractive vector for attackers to instigate phishing attacks, spread malware, or perform other malicious activities by leveraging this weakness to register the same custom URL handler.

To mitigate this issue, Cure53 recommends adopting Android *App Links* rather than *Deep Links* to ensure that only the legitimate application is capable of handling redirect URI. For iOS, one can advise implementing iOS *Universal Links* as a valid replacement.

BWN-11-004 WP1: Keychain data persists after uninstallation *(Low)*

Testing confirmed that the application stores the user's corresponding *access* token in the iOS Keychain. All data held within it persists on the device, even after the app has been uninstalled. This may introduce security flaws in certain scenarios. For instance, if a user sells their device without performing a factory reset. In this circumstance, the buyer of the device can gain access to the previous user's application account and data just by reinstalling the application.

This vulnerability was assessed as *Low* due to the app's requirement for the master password on each occasion. Exploiting this shortcoming would demand additional effort from the attacker, who would need to extract the *authentication* token from the Keychain in some way.

To mitigate this issue, Cure53 recommends ensuring that all Keychain data associated with the application is wiped during the initial post-installation launch. This proactive step will effectively safeguard against the risk of a subsequent user inadvertently accessing the previous user's accounts and data on the device.

BWN-11-005 WP2: Bypassing biometric verification on Android *(Medium)*

Testing confirmed that the Android application's biometric authentication implementation is bypassable, allowing an attacker with physical access to obtain unauthorized access to the victim's private information without providing valid biometrics. This vulnerability stems from the misimplementation of the *onAuthenticationSucceeded* method in *BiometricPrompt*, which should be triggered when a user is successfully authenticated by the system. Typically, it includes the logic responsible for unlocking the application. However, the associated configuration solely relies on calling this method, without considering the *CryptoObject* containing a reference to the Keystore entry that should be unlocked.

Exploiting this vulnerability involves hooking into the application process and directly calling the *onAuthenticationSucceeded* method, bypassing the expected biometric verification process. Notably, the following exploit was achieved on an Android device running API 33.

Steps to reproduce:

1. Install Frida on a testing environment following this tutorial: *https://medium.com/infosec-adventures/introduction-to-frida-5a3f51595ca1*
2. Ensure that the target Android device is connected to the testing environment.
3. Launch the Bitwarden application on the device and enable biometrics verification.
4. Download the Frida script from the link below and execute it.

   Command:
   ```
   frida -U -f com.livefront.bitwarden -l fingerprint-bypass.js
   ```

5. Observe that the Frida script hooks into the application process, enabling interception of method calls and modification of their behavior.
6. Perform the biometric authentication process within the application as normal.
7. Note that the application unlocks without requiring valid biometrics.

Frida script:
https://gist.github.com/Kahoul99/332e1543dc09a93d6f23045624f8a239

To mitigate this issue, Cure53 recommends implementing secure fingerprint scanning, which is achievable by utilizing the Android Keystore. One effective approach in this respect would involve initializing a cipher object with an Android Keystore key, implementing the *onAuthenticationSucceeded* callback to decrypt crucial data using the cipher object, and concluding the process by calling *BiometricPrompt.authenticate* with a crypto object created using the cipher object and callback. For detailed implementation steps and supplementary guidance concerning secure local authentication, please refer to the resources available online in blog posts[1] and example projects[2].

[1] https://labs.withsecure.com/publications/how-secure-is-your-android-keystore-authentication
[2] https://github.com/WithSecureLabs/android-keystore-audit/tree/master/keystorecrypto-app

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, whilst a vulnerability is present, an exploit may not always be possible.

BWN-11-002 WP1-2: Content injection in *captcha* connector *(Low)*

Testing of the *captcha* mobile connector revealed that the website displays an arbitrary message encoded in the URL. This message could be used to carry out social engineering attacks against users.

Steps to reproduce:

1. Encode the desired content as JSON:

```
{"captchaRequiredText":"test"}
```

2. Construct an URL by base64-encoding the JSON and appending it to the *captcha* mobile connector URL:
https://vault.bitwarden.com/captcha-mobile-connector.html?data=eyJjYXB0Y2hhUmVxdWlyZWRUZXh0IjoidGVzdCJ9&parent=https://vault.bitwarden.com/&v=1

Cure53 recommends allowing only a defined set of messages to be displayed on the *captcha* connector page rather than displaying any arbitrary message. This could be implemented by adding an allow-list for messages.

BWN-11-003 WP1-2: Lack of *root* detection and anti-debugging defenses *(Low)*

Testing confirmed that the current implementations of the apps offer neither *root* nor jailbreak detection. They also omit anti-debugging mechanisms. As stipulated in the *OWASP MASTG*[3] guidelines, it is important for every Android and iOS application to incorporate these features to reduce the risk of tampering, and to strengthen the resilience of the mobile apps more generally.

To address this concern, Cure53 suggests implementing alerts for users who are operating the app on rooted devices. This proactive measure would effectively inform users about the potential risks associated with utilizing the app on rooted devices, especially considering Bitwarden's handling of highly sensitive data.

[3] https://github.com/OWASP/owasp-mastg

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

BWN-11-006 WP3: Master password not cleared from memory *(Info)*

While auditing the *bitwarden* crate in the Bitwarden SDK monorepo, it was found that - contrary to the security whitepaper - the master password is not cleared from memory. The Bitwarden team has acknowledged the existing limitations of this internal and unfinished SDK. As such, this issue is only documented here for future reference.

Affected file:
sdk/crates/bitwarden/src/auth/login/password.rs

Affected function:
login_password

Affected file:
sdk/crates/bitwarden/src/client/encryption_settings.rs

Affected function:
new (constructor)

It should be noted that both affected functions currently take the password as an immutable string reference. Clearing the password from memory thus requires a redesign where both functions take ownership of the password. Additionally, it should be noted that the constructor of *EncryptionSettings* only needs the password to derive the master password hash. This has already happened in the *login_password* case, and passing the cleartext password again is unnecessary.

Cure53 recommends reviewing whether passing the master password in cleartext is necessary in all cases. The missing functionality for clearing the master password from memory should be implemented as well.

BWN-11-007 WP3: Unforced *unwraps* may cause panic *(Info)*

While reviewing the *bitwarden* crate in the SDK's monorepo, a number of unnecessary uses of *unwrap* were identified. While these do not pose a security problem as such, they increase the complexity of error handling for users of this security-focused SDK.

It should be pertinently noted that only usages of *unwrap* that can directly be replaced by error propagation are reported here.

Affected file:
sdk/crates/bitwarden/src/mobile/vault/client_sends.rs

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Affected code:
```
let data = std::fs::read(encrypted_file_path).unwrap();
...
let data = std::fs::read(decrypted_file_path).unwrap();
```

Affected file:
sdk/crates/bitwarden/src/mobile/vault/client_attachments.rs

Affected code:
```
let data = std::fs::read(decrypted_file_path).unwrap();
...
let data = std::fs::read(encrypted_file_path).unwrap();
```

Cure53 recommends changing the reported cases to error propagation. All of the usages of *unwrap* should be carefully reviewed, as they may break the designed error handling mechanisms in Rust and foster panics.

BWN-11-008 WP3: Outdated Docker image with critical vulnerabilities *(Low)*

During the source code review of the *bws* crate containing code for the Bitwarden secrets manager, it was found that the application is running in a Docker container. A security scan of the built container revealed a total of 78 known vulnerabilities, among them even a critical one[4].

Quite clearly, utilizing a Docker image with known vulnerabilities to deploy secret handling software may provide avenues for attackers who seek to leak client secrets or authentication sessions.

During this evaluation, it was determined that merely updating the packages within the container does not address the underlying issues. Instead, transitioning to a newer base image is essential. The experiments, especially those involving *ubuntu:latest,* showed that using a newer base image would lead to a functional client and improved container security.

Affected file:
sdk/crates/bws/Dockerfile

Affected code:
```
[...]
###############################################
#                  App stage                  #
###############################################
FROM debian:bookworm-slim

ARG TARGETPLATFORM
LABEL com.bitwarden.product="bitwarden" [...]
```

[4] https://avd.aquasec.com/nvd/2023/cve-2023-45853/