Kraken Mobile Wallet iCloud Backup

Security Assessment (Summary Report)

October 8, 2024

Prepared for:
Kraken

Prepared by: **Tjaden Hess and Joop van de Pol**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
497 Carroll St., Space 71, Seventh Floor
Brooklyn, NY 11215
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Mike Shelton, Project Manager
mike.shelton@trailofbits.com

The following engineering director was associated with this project:

Jim Miller, Engineering Director, Cryptography
james.miller@trailofbits.com

The following consultants were associated with this project:

Tjaden Hess, Consultant
tjaden.hess@trailofbits.com

Joop van de Pol, Consultant
joop.vandepol@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
September 03, 2024	Project start
September 10, 2024	Delivery of report draft and readout meeting
September 16, 2024	Delivery of summary report
October 8, 2024	Delivery of summary report with fix review

Project Targets

The engagement involved a review and testing of the following target.

Kraken Mobile Wallet

File	`mobile-opensource-1.8.1.zip`
SHA1 sum	`c68cd16bc92f0e3a95263194e6b72e639218c91c`
Version	`1.8.1`
Type	React Native, Swift
Platform	iOS

Executive Summary

Engagement Overview

Kraken engaged Trail of Bits to review the security of iCloud backups for their mobile wallet. Kraken developed a passkey-based backup flow using the "`largeBlob`" WebAuthn extension. This enables iOS users to back up their Secret Recovery Phrase (SRP) by storing it in a passkey, which is synchronized with the iCloud Keychain.

A team of two consultants conducted the review from September 3, to September 9, 2024, for a total of two engineer-weeks of effort. With full access to source code and documentation as well as the Testflight version, we performed static and dynamic testing of the Kraken Mobile Wallet iCloud Backup, using automated and manual processes.

Observations and Impact

The review focused on the protection of the SRP against unauthorized access, the correct use of WebAuthn, and the handling of edge cases. The review covered the native iOS module implementing all Keychain operations such as passkey creation and metadata storage, as well as the UI components responsible for creating, importing, and deleting backups.

The review did not uncover any major design flaws, and the WebAuthn functionality is used correctly. However, the use of passkeys presents increased opportunities for phishing in the case that the `kraken.com` domains or TLS certificates are compromised. Furthermore, the code is not very robust when handling errors and other edge cases, and it is possible to bypass the lockout feature aimed at preventing brute-forcing of password-protected SRPs.

Recommendations

We recommend clarifying the passkey name and choosing a dedicated root domain to serve as the WebAuthn relying party identifier. Furthermore, we recommend improving the error handling to account for various edge cases, and to improve input validation for all security functions, instead of relying on UI elements to constrain or validate user input.

Fix Review

One consultant conducted the fix review from October 3 to October 4, 2024. Kraken resolved all implementation issues that Trail of Bits raised during the review and plans to resolve issue 4 in the near future with a support documentation update.

Summary of Findings

1. Compromised kraken.com server could phish users' mobile wallet backups [Resolved]

If the `wallet.kraken.com` domain is compromised, an attacker with access to the domain and a corresponding TLS certificate could prompt a user to log in with their `wallet.kraken.com` passkey and thus accidentally reveal their wallet secret phrase.

For example, if the main www.`kraken.com` server has access to the `wallet.kraken.com` TLS certificate private key, an attacker who compromises www.`kraken.com` could redirect users to `wallet.kraken.com` during login. Users who regularly log in to the Kraken exchange with a passkey would likely not notice that the passkey being requested belongs to a different domain than usual. Users may not be aware that mobile wallet seed phrases are also present on their MacBooks and other apple devices.

Several mitigation strategies may be appropriate:

- Include a clear name and warning in the passkey name string. Currently, the name presented to the user on authorization is simply an innocuous-looking identifier for their device.
- Ensure that `www.kraken.com` is hosted in such a way that it does not have access to TLS certificates for `wallet.kraken.com`. For example, avoid issuing wildcard certificates or performing DNS-based certificate authorization for the root `kraken.com` domain.
- Consider changing the relying party name to something that more clearly differentiates the wallet from the Kraken web exchange, such as a separate root domain.
- Enroll in a certificate log monitoring service, ensuring that you are notified of fraudulently issued TLS certificates.

2. Password failure counter resets when providing an empty password [Resolved]

The user can choose to protect the mnemonic and seed using a password, in which case the app encrypts them using a password-derived key before storing them in the iCloud Keychain. When performing a cloud backup of a password-protected mnemonic, the user is prompted for their password to retrieve the mnemonic. If this fails, the app increments a lockout counter that—upon reaching 20 attempts—will prevent the user from trying again for some time. If the function that retrieves the mnemonic succeeds in returning a value, the lockout counter is reset.

```
export async function retrieveMnemonic(appLockSecret?: string, password?: string) {
  const value = await retrieveAppLockProtectedValue(KeychainKey.mnemonicKey,
appLockSecret, 'utf8');
  const isPasswordProtected = await getFromKeychain(KeychainKey.isSeedEncryptedKey);
  if (password && isPasswordProtected) {
    return decryptValue(value, password, 'utf8');
  }
  return value;
}
```

Figure 1: `retrieveMnemonic` always succeeds when the supplied password is empty (`src/secureStore/domains/secure.ts#99–106`)

However, the function that retrieves the mnemonic will attempt to decrypt it only if the mnemonic is configured to be password-protected and if the provided password is non-empty (figure 1). If either of these conditions is not met, it will directly return the value from the Keychain. Therefore, entering an empty password for a password-protected mnemonic will cause the function to return the ciphertext value, and the app will reset the lockout counter.

Typically, a user cannot submit an empty password because the submission button is inactive when the password is empty. However, it is possible to submit an empty password by entering some text, deleting it, and pressing the button before the UI updates. This resets the password lockout counter. This allows a relatively unsophisticated adversary to bypass rate limiting when trying to guess a user's password. This issue does not expose the mnemonic or seed directly, as the returned value is still encrypted. The resulting passkey backup will fail BIP-39 validation and cannot be used.

3. Error handling is not specific [Resolved]

The backup system has three main flows: backup, import, and delete. Each of these flows comprises several steps. If an error is thrown in the following cases, the resulting message and UI displays do not accurately reflect the state of the backup, which may itself become inconsistent:

1. During backup, the user is prompted twice for passkey authorization: once for passkey registration, and once for writing the mnemonic into the passkey. If the user cancels the second prompt, a passkey is created, but the error is handled as if the user canceled passkey registration.
2. After writing the mnemonic into the passkey, the app writes the backup metadata to the Keychain key/value store. If synchronization fails, an error is thrown, and `setCloudBackupCompleted` is not called, even though all backup steps have been completed and the backup metadata is correctly stored in the local store.
3. During delete, the user is prompted for passkey authorization to write `"DELETED"` to the passkey's large blob. Then, the corresponding backup metadata is deleted

from the Keychain key/value store. However, if the user has already deleted the passkey outside of the app, the first step will always fail. Hence, the app will never delete backup metadata nor call `removeCloudBackup`.

4. Again, if the key/value-store synchronization fails, the app throws an error and does not call `removeCloudBackup`, even though all deletion steps have been completed and the backup metadata is correctly removed from the local store.

5. The error messages when deleting a backup may say “Error deleting passkey,” but this should read “Error deleting passkey backup,” as deleting the backup does not remove the passkey.

4. Users may share passkeys with friends and family

iOS supports the sharing of passwords and passkeys between users, such as for family accounts. Users should be made aware that sharing the backup passkey will share access to their cryptocurrency wallet. Further, some users will share devices and iCloud accounts with family members. Users should be made aware that by backing up their wallet to iCloud, the wallet will no longer be protected by the Kraken Wallet password protection feature and will be accessible to anyone with access to the iCloud account and mobile device password.