COSSACK LABS

Temple Wallet Mobile security review summary

Overview and Executive summary

In October 2021, Tezos Foundation and Madfish requested Cossack Labs to offer an opinion on improving security and cryptography aspects of Temple Wallet Mobile source code and cryptographic design.

Temple Wallet Mobile, a mobile cryptocurrency wallet for the Tezos ecosystem, provides users with the ability to manage XTZ tokens (any FA 1.2 and FA 2 tokens) and interact with decentralized applications (DApps). Temple Wallet Mobile allows the users to connect with Temple Wallet Web Extension.

As Temple Wallet Mobile doesn't have a public set of security claims and risk statement, we assume that wallet's users expect the same level of security as any "hot" non-custodial crypto-wallet (based on other wallets and their explicit guarantees):

Temple Wallet Mobile stores users' accounts securely, does not leak keys, passwords, sensitive data, prevents unauthorized access to the wallet and prevents unauthorized transaction sending and signing.

The risk statement above is limited to the mobile application itself, so the user's mobile device and mobile OS compromise are out of direct scope.

We have found out that if Temple Wallet Mobile users understand:

- their responsibilities of protecting their account credentials,
- limitations of mobile applications and hot non-custodial wallets,
- their responsibility of ensuring mobile device and mobile OS security prior to unsealing the wallet,

… then users' data, keys and transactions are acceptably protected with Temple Wallet Mobile's security measures.

During the current review, security improvements and "raising the bar" security controls have been advised to improve the security of Temple Wallet Mobile and support the security claim.

We found a number of broken/missing security controls, risky design decisions and application security bugs in these areas:

- cryptographic design and implementations,
- application security,
- platform trust (iOS-specifics, Android-specifics, React Native specifics),
- code quality,
- supply chain risks,
- educating users about limitations of hot non-custodial wallets.

Under unfavourable circumstances could lead to losing users' password, account mnemonics and private keys, triggering unauthorized transactions, or even flooding the network with malformed transactions.

As the main goals of this engagement were to improve the security of Temple Wallet Mobile, we provided numerous suggestions on application security, cryptography usage and design, mitigating platform-specific risks (mobile and React Native), implementing defenses against reverse engineering and tampering, and improving general stability and maintainability of Temple Wallet Mobile by building defense-in-depth protections.

As a result, a list of security findings has been identified and communicated to Temple Wallet Mobile team, along with recommendations:

Findings area	High	Medium	Low
Software design	1	11	1
Cryptography (design & implementation)	4	7	5
Application security	4	8	4
Platform security	1	8	1
Code quality	0	0	4
Supply chain risks	0	2	0

Findings with all levels of severity have been reported, since many low-priority ones may become stepping stones in multi-step attacks.

As result of the engagement, 78% of found issues were fixed and 6% were scheduled to be resolved in future Temple Wallet Mobile releases. The security posture of the application was significantly improved.

This review was performed during November-December 2021. It consisted of around 270 person-hours of work, allocated between design review, risk/threat modelling, implementation review, cryptographic review, untie 3rd party libraries dependencies, and verification of the chosen security controls.

The verification of the fixed issues was performed during March-April 2022.

Goals and approach

Review goal: improve security and cryptographic aspects of Temple Wallet Mobile and surrounding ecosystem relevant to security of key in the non-custodial wallet itself and transactions it signs.

Risk statement and security claim: Security components inside Temple Wallet Mobile should be sound against the following risk statement (C.A.S.E.):

> *Financial loss (consequence) due to active and passive adversaries (source) exploiting application security and cryptography flaws (event) in Temple Wallet Mobile, resulting in unauthorized usage of secrets, keys and other identifying material (assets) to perform unauthorized transactions.*

Scope for risk statement: limited to mobile application itself – local storage of sensitive data, transaction communication, preventing unauthorized access to the wallet, preventing adversarial inputs or tricking the users into signing unauthorized transactions. Protections against typical mobile and React Native platform weaknesses and misconfiguration. Protections against mobile devices compromise

(jailbreak / root verification, remote device attestation). Decentralized application's behaviour is out of direct scope.

In scope: mnemonics, seed and private keys life cycle (creation, storage, import/export), transactions correctness (transactions falsification and abuse), cryptography design and implementations around key lifecycle and transactions, secure management of sensitive data, usage of 3rd party libraries and components, usage of platform-specific security controls, data leakage from the wallet application, application <> backends communication (relevant to the threat modelling).

Non-scope for risk statement:

- The functionality and safety of Tezos mainnet is out of scope of the current review.
- It is understood that Temple Wallet Mobile security controls might be partially or fully compromised if the mobile device is jailbroken / rooted, or is under Pegasus-like attack.
- It is understood that anyone with access to the users' mobile device has a partial or full control over their wallet.
- Security and safety of 3rd party backends and DApps that Temple Wallet Mobile communicates with, is out of scope of the current review.
- 3rd party implementations of cryptographic primitives are considered trusted, or can be replaced by well-studied ones.

Coverage:

This security review has been based on:

- Code: commit 6f8d2b7018c52ffa81118271a0a55a6114c97c21
  https://github.com/madfish-solutions/templewallet-mobile/commit/6f8d2b7018c52ffa81118271a0a55a6114c97c21

  Initial code commit was a212fd52bf3bad2deff6b5572c2f31465e063a89, it was re-agreed between Cossack Labs and Madfish teams at the project kick off stage. The goal of changing commit ID was to incorporate the latest improvements made by the Madfish team to the Temple Wallet Mobile.

  Madfish team has created a fork of repository pointed on a 6f8d2b commit that contains a "code freeze" version (no changes were added):
  https://github.com/madfish-solutions/templewallet-mobile-audit/tree/6f8d2b7018c52ffa81118271a0a55a6114c97c21/

- Design documentation and components description provided by the Madfish team.

The review was conducted using the following devices: iPhone XS (iOS 14.7.1), iPhone 7 (iOS 13.1.3), iPhone 11 Pro (iOS 14.6), iPhone 12 Pro (iOS 15.0.2), Xiaomi Redmi Note 9 (Android 10), AVD (Android 7.1.1), AVD (Android 10).

Methodology

Cossack Labs' review has constituted of a number of activities:

- **General risk model clarification and security review**: formulating realistic risk models and threat vectors that affect user safety.

- **Research of fundamental issues of cryptocurrencies in context of interacting with wallet**: analyzing common issues inherent to blockchain/network and linking them to particular wallet's issues. The research was made in full during Temple Wallet Web extension audit, parties agreed not to perform the same research again, but to include only details relevant for the Temple Wallet Mobile.
- **Design/architecture review**: proactively seeking design flaws that lead to leakage of sensitive data stored within a Temple Wallet Mobile or manipulating transaction flow. As Temple Wallet Mobile uses React Native platform, design review was undertaken with the respect of React Native specifics.
- **Cryptographic design and implementation review**: verifying whether the chosen combination of cryptographic primitives and their implementation actually embodies desired security properties. Analysing a high-level encryption scheme, its drawbacks and suggesting improvements.
- **Application, platform, data security**: ensuring that application level security controls are implemented well, researching platform-specific threats and providing recommendations, providing recommendations against supply chain risks, providing recommendations against reverse engineering and tampering.

Triaging issues: due to the specific risk statement / review goal, issues discovered could not be triaged using a common methodology like CWE or CVSS – the outcomes, loss magnitude in the general context are significantly different compared to a regular security assessment and some of the vulnerability severity scores would be misleading.

Thus, we've conducted a simple risk assessment, formulated a simple trust and risk model that reflects the chosen risk statement, and used it to triage vulnerabilities relevant to risks with Temple Wallet Mobile. To avoid comparing apples vs oranges, we've also added type to findings – representing the corresponding domain ("architecture", "application security", "cryptography").

Findings summary

Our general conclusion is that the set of security controls implemented by the Madfish team can achieve the security claims / prevent risk statement to a satisfactory degree. However, mobile applications provide a set of device- and platform-specific security controls (like, biometrics authentication, or hardware key storage) that could significantly "raise the bar" for attackers.

Types: broken controls (security controls that are implemented but doesn't satisfy security requirements), missing controls (lack of protections), enhancements (security-related suggestions to implement).
Areas: cryptography, software design, application security, code quality, platform, supply chain.

Findings and references

After issues verification, the statuses are the following:

Fixed – the issue is confirmed by the Madfish team, the fix was implemented in full, and validated by Cossack Labs team.
Partially fixed – the issue is confirmed by the Madfish team, the fix was implemented partially, and validated by Cossack Labs team.
Scheduled, not fixed – the issue is confirmed by the Madfish team, planned and scheduled for

later, currently not fixed.
Won't do – the issue was confirmed by the Madfish team, but the security risks were accepted, no fix was implemented. The finding is relevant, but fixing or implementing it might be difficult because of technical reasons (platform limitations) or product reasons (UX trade-offs, questionable value for users, cross-compatibility with other Tezos wallets). Madfish team will re-evaluate these findings during planning of the next product features.
Not an issue – the issue was rejected by the Madfish team due to technological / platform limitations making it impossible to solve.

ID	Finding	Severity / priority	Type	Area	Status
D-001	Do not suggest reusing the password when importing mnemonics	High	enhance	design	Won't do
D-002	Wallet password improvements: follow NIST guidelines	Med	enhance	design	Partially fixed
D-003	Add additional auth step before critical actions	Med	enhance	design	Won't do
D-004	Clear Keychain on app reinstall	Med	missing	design	Fixed
D-005	Educate users on using the Temple Wallet	Med	missing	design	Scheduled, not fixed
D-006	Password not removed on wallet reset	Med	enhance	design	Fixed
D-007	Lack of password change functionality	Med	missing	design	Scheduled, not fixed
D-008	Confusing and not user-friendly errors on transaction flow	Med	enhance	design	Fixed
D-009	QR-code wallet sync: limit QR-code exposure on the screen	Med	missing	design	Fixed
D-010	QR-code wallet sync: Use one-time password	Med	broken	design	Won't do
D-011	QR-code wallet sync: Protect against malicious QR-codes	Med	missing	design	Fixed
D-012	Force application update feature	Med	missing	design	Fixed
D-013	Add security.txt to the project	Low	enhance	design	Fixed

C-001	PBKDF2: increase rounds or switch to another KDF	High	enhance	crypto	Fixed
C-002	Potentially insecure CPRNG is used, change CPRNG	High	broken	crypto	Fixed
C-003	Improper error handling may lead to an empty IV in AES-CBC	High	broken	crypto	Fixed
C-004	Secrets are encrypted with AES-CBC — use AEAD instead	High	enhance	crypto	Fixed
C-005	Use the same CPRNGs across the app	Med	enhance	crypto	Fixed
C-006	Abandoned SpongyCastle library is used for PBKDF2 on Android	Med	broken	crypto	Fixed
C-007	Keystore decryption from Kukai v1 – fix or stop supporting	Med	broken	crypto	Fixed
C-008	Consider stop using react-native-aes-crypto library and switch to another lib instead	Med	enhance	crypto	Fixed
C-009	Too many cryptographic libraries that provide similar functionality – wide attack surface	Med	enhance	crypto	Fixed
C-010	Poor memory management of secrets	Med	enhance	crypto	Fixed
C-011	Lack of proper testing of cryptographic code	Med	missing	crypto	Not an issue
C-012	Android Shelter encryption doesn't depend on the explicitly selected cipher mode, pay attention	Low	enhance	crypto	Fixed
C-013	Cryptography export regulations: filing an annual self-classification BIS report	Low	enhance	crypto	Fixed
C-014	Password verification uses a static public value, easing password brute-force	Low	broken	crypto	Fixed
C-015	Untie encryption keys from application password	Low	enhance	crypto	Won’t do
C-016	Reactive crypto code might be difficult to maintain	Low	enhance	crypto	Won’t do
A-001	Transaction fee manipulation	High	broken	appsec	Fixed

A-002	Exposing secrets to 3rd-party keyboard software	High	broken	appsec	Fixed
A-003	No validation for RPC node names	High	missing	appsec	Partially fixed
A-004	No integrity check for DApps list	High	missing	appsec	Fixed
A-005	React-native-keychain library: storage option is not specified, credentials are stored without security guarantees	Med	enhance	appsec	Fixed
A-006	Lack of brute-force countermeasures for user password	Med	missing	appsec	Fixed
A-007	Mnemonics and private keys are saved to keyboard cache and autocorrection dictionary	Med	broken	appsec	Fixed
A-008	QR-code wallet sync: No brute-force countermeasures during wallet sync via QR-code	Med	missing	appsec	Fixed
A-009	User password could be disclosed for unlimited time	Med	broken	appsec	Partially fixed
A-010	Insecure TLS settings are supported	Med	broken	appsec	Partially fixed
A-011	Input validation: kukai and sync mnemonics not checked for bip39	Med	broken	appsec	Fixed
A-012	HTTP links are allowed	Med	broken	appsec	Fixed
A-013	Add additional level of encryption for wallet password	Low	enhance	appsec	Won't do
A-014	HTTP traffic allowed for localhost	Low	enhance	appsec	Fixed
A-015	React-native-keychain library: accessibility option is not specified, keychain is available without passcode	Low	enhance	appsec	Fixed
A-016	No certificate pinning	Low	missing	appsec	Partially fixed
P-001	Lack of protections against Android overlay attacks	High	missing	platform	Fixed

P-002	Require device-level passcode	Med	missing	platform	Fixed
P-003	Revise application permissions	Med	enhance	platform	Fixed
P-004	Activity (com.templewallet.MainActivity) possibly not protected	Med	broken	platform	Won't do
P-005	Potentially weak Android app bundle signature	Med	broken	platform	Fixed
P-006	Lack of reverse-engineering protections for iOS	Med	missing	platform	Scheduled, not fixed
P-007	Lack of reverse-engineering protections for Android	Med	missing	platform	Scheduled, not fixed
P-008	Validate Android device with SafetyNet	Med	missing	platform	Fixed
P-009	Validate iOS device with App Attest	Med	missing	platform	Fixed
P-010	Enable Proguard in Android release builds	Low	missing	platform	Fixed
CQ-001	Code quality issues discovered by SASTs (SonarQube, WebStorm)	Low	enhance	code	Partially fixed
CQ-002	Android Studio project too many errors	Low	enhance	code	Fixed
CQ-003	Xcode project warnings	Low	enhance	code	Not an issue
CQ-004	Errors and warnings in security-related features	Low	broken	code	Fixed
S-001	Vulnerabilities in third-party dependencies (includes critical and high)	Med	broken	supply chain	Fixed
S-002	Lack of dependency and vulnerability management process	Med	missing	supply chain	Fixed

After initial audit and communicating with Madfish team, assisting them in triaging and mitigating the issues, Madfish team has provided the following statuses for each issue:

"Acknowledged, to be fixed" – 48 findings,
"Acknowledged, scheduled, not fixed" – 4 findings
"Acknowledged, but won't do" and "not an issue" – 9 findings.

After Madfish team has fixed acknowledged issues, and Cossack Labs team has verified the fixes, the current status is the following:

"Fixed", or "Partially fixed" – 48 findings,
"Postponed, to be fixed later" – 4 finding,
"Acknowledged, but won't do" and "not an issue" – 9 findings.

Conclusion

In this report, we outlined the process and findings of our security assessment of Temple Wallet Mobile, its architecture, theoretical and practical concerns. We reviewed Temple Wallet Mobile secure storage, its interaction with users, utilisation of Android and iOS specific platform features, and communication with the Tezos network, intermediate backends and DApp.

We have found out that if Temple Wallet Mobile users understand:

- their responsibilities of protecting their account credentials,
- limitations of mobile applications and hot non-custodial wallets,
- their responsibility of ensuring mobile device and mobile OS security prior to unsealing the wallet,

… then, users' data, keys and transactions are acceptably protected with Temple Wallet Mobile's security measures.

We would like to note a solid security-oriented engineering effort of the Madfish team in building and securing the app. They've implemented many security controls, like application locking, protecting user accounts with a password and data-at-rest encryption, user confirmation on transactions, hiding sensitive data from the screen, encrypting mnemonics during import from other wallets, and many more, to the best of their developer qualification.

However, we found a number of broken/missing security controls that under unfavourable circumstances could lead to losing users' password, account mnemonics and private keys, triggering unauthorized transactions, or even flooding the network with malformed transactions.

As the main goals of this engagement were to improve the security of Temple Wallet Mobile, we provided numerous suggestions on application security, cryptography usage and design, mitigating platform-specific risks, implementing defenses against reverse engineering and tampering, and improving general stability and maintainability of Temple Wallet Mobile by building defense-in-depth protections.

An update after verification of fixed issues:
After the mitigations and fixes applied by the Madfish team, the number of unresolved issues decreased significantly to 1 high, 8 medium, and 4 low issues. We would like to note that we are very satisfied with Madfish reactions and mitigations: the team not only fixed the existing issues but planned future improvements to harden implemented security controls and add more security features.

Madfish has scheduled 4 issues to be fixed in the near future, after the first production release. In general, the security posture of Temple Wallet Mobile has been greatly improved, but not all security concerns were eliminated.

About Cossack Labs

Cossack Labs is a provider of data security tools (cryptographic and data security frameworks), bespoke solutions and consulting services, with a focus on sensitive data protection in modern systems. Cossack Labs' experts participating in this audit, have decades of hands-on practical experience, appropriate formal education and academic degrees in cryptography, data security and general information security. Cossack Labs' security engineers are acknowledged contributors to popular industry standards (OWASP MASVS/MSTG) and hold appropriate certifications (CISSP).

Due to the nature of our skillset, our review aims not only to detect potential weaknesses but also to provide clear, actionable advice for developers to rapidly improve security in their applications, as communicated by thinking-alike engineers.

Cossack Labs can be contacted at: cossacklabs.com / info@cossacklabs.com