Temple Wallet web extension security audit report summary

Overview and Executive summary

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

Temple Wallet, a browser extension 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.

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

We have found out that if Temple Wallet users understand:

- their responsibilities of protecting their account credentials,
- limitations of web extensions and hot non-custodial wallets,
- their responsibility of ensuring host OS and browser security prior to unsealing the wallet,

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

However, we found a number of brokens security controls, missing security controls, risky design decisions and application security bugs in these areas:

- cryptographic design and implementations,
- application security,
- platform trust,
- code quality,
- supply chain risks,
- educating users about limitations of web extensions and non-custodial wallets,
- user's susceptibility to deanonymisation.

Under unfavourable circumstances these issues could lead to sensitive data leakage, triggering unauthorized transactions, losing wallet's data, deanonymising users or potentially DoS-ing the network with malformed transactions.

As the main goals of this engagement was to improve security of Temple Wallet, we've also separately provided numerous suggestions on application security, cryptographic flow improvements, mitigating platform-specific risks and improving general stability and maintainability of Temple Wallet by building defense-in-depth protections.

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

Findings area	**High**	**Medium**	**Low**
Software design	4	6	3
Cryptography	2	3	3
Application security	3	3	8
Platform security	0	1	3
Code quality	0	2	2
Infrastructure	0	2	1
Supply chain risks	0	1	0

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

This review was performed during August – October 2021. It consisted of around 240 person-hours of work, allocated between design review, risk/threat modelling, implementation review, cryptographic review and verification of the chosen security controls.

The verification of the fixes issues was performed during January – February 2022.

Goals and approach

Review goal: improve security and cryptographic aspects of Tezos' ecosystem components within Madfish work.

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

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

Scope for risk statement: limited to web extension itself – local storage of sensitive data, transaction communication, preventing unauthorized access to the wallet. Browser/host machine compromise is out of direct scope, however, using additional techniques, Temple Wallet security components could alleviate some of the host compromise risks ("raising the bar for a loss event").

In scope: private key life cycle (seed creation, seed storage, seed import/export), application <> backends communication (for backends relevant to the threat modelling), 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, data leakage from the wallet application.

Non-scope for risk statement:

- The functionality and safety of Tezos mainnet is out of scope of the current review.
- It is understood that anyone with access to the users' local machine and their browser have full control over the wallet in that browser.
- Security and safety of 3rd party backends that Temple Wallet connects to to retrieve history or exchange value, 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 initial security review has been based on:

- Code: commit 1af6e5 626f75a325fdc065396884ec5afcade835f8d238 that corresponds to Release 1.11.6.
- Design documentation and components description provided by the Madfish team.

The review was conducted using the following web browsers: Chrome and Chrome-like (as their market share is ~70%) and Firefox.

Chrome: 92.0.4515.159 (Official Build) (x86_64), 94.0.4606.71 (Official Build) (x86_64), 90.0.4430.93 (Official Build) (x86_64), 94.0.4606.61 (Official Build) (x86_64).
FireFox: 91.0.1 (64-bit), 92.0.1 (64-bit).
Chromium: 95.0.4608.0 (Developer Build) (x86_64), 93.0.4577.82 (Official Build) (x86_64).

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.
- **Design/architecture review**: proactively seeking design flaws that lead to leakage of sensitive data stored within a Temple Wallet web extension or manipulating transaction flow.
- **Cryptographic design and implementation review**: verifying whether the chosen combination of cryptographic primitives and their implementation actually embodies desired security properties.

- **Application security**: ensuring that application-level security controls are implemented well, researching platform-specific threats and providing recommendations, providing recommendations against supply chain risks.

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. To avoid comparing apples vs oranges, we've also added type to issues/improvements - 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, web extension applications operate in a risky environment – their security relies on the browser security and security of the user machine. "Hot" non-custodial wallets require operational safety – users are responsible for safely storing mnemonics and keys of their accounts. Often users might be not aware of the risks of using web extension wallets.

As the main goals of this engagement was to improve security of Temple Wallet, we provided numerous suggestions on application security, cryptographic flow improvements, mitigating platform-specific risks and improving general stability and maintainability of Temple Wallet by building defense-in-depth protections.

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 or in case of significant web development technological changes.
Not an issue – the issue was rejected by the Madfish team, because it is a by-design behaviour of Temple Wallet or Tezos ecosystem.

Temple Wallet web extension security audit report summary

For: Tezos Foundation

Confidential. 09.10.2021, updated 22.02.2022

ID	Problem	Severity / priority	Type	Area	Status
D-001	Educate users on using the Temple Wallet	Med	missing	design	Fixed
D-002	Wallet storage: no authentication, no integrity checks	High	missing	design	Fixed
D-003	Same user avatars across all systems	Med	impr / broken	design	Won't do
D-004	Address book storage and possible leakages	Med / High	broken	design	Fixed
D-005	Wallet password improvements: follow NIST guidelines	Med / High	impr	design	Partially fixed
D-006	No wallet data authentication - possible wallet steal and simplified brute-force	Med / High	missing	design	Won't do
D-007	Lock web extension after timeout	Med	missing	design	Partially fixed (done as much as possible now)
D-008	No functionality to "logout" from wallet (delete current wallet)	Med	missing	design	Fixed
D-009	No functionality to reveal the mnemonics of Imported Accounts	Med	missing	design	Fixed
D-010	Wallet mnemonics: hide from prying eyes in UI, follow NIST password guidelines	Med	impr	design	Won't do
D-011	Mnemonics generation improvements	Low	impr	design	Won't do
D-012	Allow users to generate multiple signature keypairs for each account	Low	impr	design	Not an issue
D-013	dApps interaction: limit permissions	Low	impr	design	Won't do
C-001	PBKDF2: increase rounds or switch to another KDF	High	impr	crypto	Fixed
C-002	Poor memory management of secrets	High	broken	crypto	Fixed
C-003	Storing dApps keys in plaintext in local storage [beacon]	Med	broken	crypto	Scheduled, not fixed
C-004	Storing Beacon Keypair Seed and Beacon Keypair in plaintext in local storage [beacon]	Med	broken	crypto	Won't do
C-005	"vault_check" value is null during encryption	Med	impr	crypto	Fixed

C-006	Lack of zeroing cryptographic keys / material	Low	broken	crypto	Won't do
C-007	Encryption lacks versioning	Low	impr	crypto	Won't do
C-008	LibSodium Beacon keypairs: improve deriving public key from secret key [beacon]	Low	impr	crypto	Fixed
A-001	Transaction fee manipulation	Med / High	broken	appsec	Fixed
A-002	Wallet wipe/delete without password	Med / High	broken	appsec	Won't do
A-003	HTTP links are allowed	High	broken	appsec	Fixed
A-004	Limit time of revealing the Mnemonics / Private key on screen	Med	impr	appsec	Fixed
A-005	Restrict origin of postMessages for content scripts	Med	missing	appsec	Fixed
A-006	Lack of brute-force countermeasures for user password	Med	missing	appsec	Fixed
A-007	Extension collect analytics without user consent	Low	broken	appsec	Fixed
A-008	"unsafe return" code issues	Low	broken	appsec	Fixed
A-009	Directory traversal and Insecure direct object references	Low	broken	appsec	Won't do
A-010	Transaction fee: inconsistent display	Low	broken	appsec	Fixed
A-011	"seed_revealed" is never false	Low	broken	appsec	Fixed
A-012	Import Account – used "account number 1" by default	Low	broken	appsec	Not an issue
A-013	Leaking error messages	Low	broken	appsec	Fixed
A-014	Leaking log message in Safari	Low	broken	appsec	Fixed
P-001	Decrease chances of Spook.js exploit	Med	impr	platform	Partially fixed (done as much as possible now)
P-002	Migrate to chrome manifest v3	Low	impr	platform	Won't do
P-003	Password save in Firefox	Low	impr	platform	Won't do
P-004	Shared extension settings among different wallets	Low	impr	platform	Fixed

CQ-001	Lack of proper testing of transaction signer cryptographic code	Med	missing	code	Fixed
CQ-002	Lack of proper testing of beacon-related cryptographic code [beacon]	Med	missing	code	Fixed
CQ-003	Code quality issues from WebStorm	Low	broken	code	Fixed
CQ-004	SonarQube discovered issues	Low	broken	code	Fixed (done as much as possible now)
INF-001	Server header in mainnet-node.madfish.solutions	Med	broken	infra	Won't do
INF-002	Error response from the server reveals its internal logic	Med	broken	infra	Won't do
INF-003	Use Tezos private node instead of public node	Low	impr	infra	Won't do
S-001	Large amount of high severity issues in dependencies	Med	broken	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" – 30 findings,
"Acknowledged, but won't do" and "not an issue" – 17 findings.

Refer to the separate document to learn Madfish explanations for each currently postponed issue.

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" – 29 findings,
"Postponed, to be fixed later" – 1 finding,
"Acknowledged, but won't do" and "not an issue" – 17 findings.

Conclusion

Web extensions are special types of applications that are bound to browser security, host machine security and user sanity. Understandably, cryptography-related extensions typically receive reasonable criticism. In this context, understanding a realistic risk model and building multiple security protections is essential to make web crypto wallets protected beyond the traditional "do not trust the browser, rely on TLS and adequate appsec measures" approach.
We would like to note a solid security-oriented engineering effort in building the Temple Wallet team. They've implemented many security controls, like wallet locking, data-at-rest

encryption, user confirmation on sensitive actions, filtering dApps, etc, to the best of their developer qualification.

We would like to emphasize Madfish's engineering maturity and commitment to create secure and usable application. During the debrief, triage and mitigations sessions, Madfish team showed strong product security engineering skills.

In this report we evaluated implementation of Temple Wallet web extension, application architecture, theoretical and practical concerns.

We have found out that if Temple Wallet users understand:

- their responsibilities of protecting their account credentials,
- limitations of web extensions and hot non-custodial wallets,
- their responsibility of ensuring host OS and browser security prior to unsealing the wallet,

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

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 the area of cryptography, data security and general information security.

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