Dartez library (Naan project) security review: executive summary

Overview and Executive summary

In January 2023, Tezos Foundation requested Cossack Labs to offer an opinion on improving security and cryptography aspects of Dartez library source code and cryptographic design.

The Dartez library contains a cryptographic core for developing applications in the Naan ecosystem that operate with Tezos Network. The Dartez library is created using the Dart programming language and compatible with the Flutter framework. It may power iOS, Android, or Web applications.

The Dartez library contains the following components: operations for managing private keys and mnemonics, cryptographic primitives, transmitting and signing transactions, estimating fees, interacting with the Tezos Network through RPC, and regex-based Michelson parser.

Dartez library is created by the Dartez development team and currently used for Naan wallets, but the long-term business expectations are that the Dartez library might be used by other community members.

Goals

Review goal: assess and improve the security and cryptographic aspects of the Dartez library as part of the Tezos ecosystem, with a focus on the functionality it provides and with the understanding that the library will be used in cryptocurrency wallets.

Risk statement

The Dartez library doesn't have public risk statements and security claims. Its primary users are developers who intend to incorporate the library into their cryptocurrency wallet applications; secondary users are cryptocurrency wallet users.

We assume that developers expect that the library provides a well-rounded security comparable with other fundamental SDKs available in the Tezos ecosystem (Taquito Tezos toolkit):

> *The Dartez library does not leak secrets (account keys, user passwords), and prevents performing unauthorised actions/transactions within SDK scope.*

We must warn that even if the SDK provides comprehensive security guarantees, it does not prevent developers from misusing the library when integrating it into their apps.

Security components should be sound against the following risk statement (C.A.S.E. model):

> *Financial loss (consequence) due to active and passive adversaries (source) exploiting security and cryptography flaws (event) in Dartez library, resulting in unauthorised usage of secrets (assets) to perform unauthorised actions.*

The risk statement above is limited to the library itself, so the user's platform compromise is out of direct influence. However, using additional techniques, the Dartez library could make a risk statement less possible by designing easy-to-use and hard-to-misuse API to raise the bar for a loss event. These security improvements have also been suggested in the current review.

In scope for risk statement: private key life cycle (seed creation, seed storage, seed import/export), RPC communication, transactions correctness (transactions falsification, signing, abuse), cryptography design and implementations around key lifecycle and transactions, usage of 3rd party libraries and components, data leakage from the library, Michelson parser (security considerations applicable to general risk model).

Out of scope of risk statement: universal Dartez library usage (API misuse risks by 3rd party apps), formal verification of Michelson parser (outside of usage patterns by Dartez library).

Coverage:
This security review has been based on a private repository located at https://github.com/Tezsure/Dartez/ with the audit tag https://github.com/Tezsure/Dartez/releases/tag/1.1.0

Methodology

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

- **Short risk / threat model clarification:** a risk/threat modelling exercise to prioritise risks and outline a general risk model to validate against.
- **Cryptographic design and implementation review:** auditing the cryptographic core, cryptographic primitives usage and implementation, API and tests in accordance with the selected threat model.
- **Application security review:** review of RPC communication with Tezos Network, testing for Tezos Network specific issues (replay attacks, re-entrance attacks, gas issues).
- **Application security review of surrounding tools**: review of tests, helper scripts, building process, deployment flow.
- **Consistency audit** of Michelson parser: grammar, compatibility, corner cases, within usage patterns of Dartez library.
- **Reporting, providing engineering assistance** to the Dartez development team in mitigation of found security issues.

Triaging issues: due to the specific risk statement and review goal, and taking into account that Dartez is a library / SDK, discovered issues could not be triaged using a common methodology like CWE or CVSS. The outcomes and loss magnitude in the general context are significantly different compared to a regular security assessment, and vulnerability severity scores would be misleading.

Thus, we've conducted a simplified risk assessment, formulated a trust and risk model that reflects the chosen risk statement, and used it to triage vulnerabilities relevant to risks with the application.

Findings summary

During the current review, we focused not only on broken and missing security controls but on improvements in the design, implementation and reliability of Dartez as the library. We found a couple of broken/missing security controls in these areas:

- Library core
- Cryptography
- Infrastructure
- Michelson Parser
- Code quality

As a result, a list of security findings has been identified and communicated to the Naan team, along with recommendations:

Findings area	High	Medium	Low
Library core	4	2	2
Cryptography	6	3	3
Infrastructure	1	4	4
Michelson parser	1	8	3
Code quality	0	2	4

Overall: 47 findings (12 high, 19 medium, 16 low).

We identified a significant number of "high" and "medium" issues. The majority of "low" issues are not related to bugs, but rather to enhancements, improvements, and the building of the missing processes and controls.

This initial review was performed during January-February 2023. It consisted of around 282 person-hours of work, allocated between design review, risk/threat modelling, implementation review, cryptographic review, code quality review, Michelson parser review, and reporting.

Verification of fixed issues

Verification of implemented fixes was performed during April-May 2023 and showed a significantly decreased amount of the remaining issues:

Findings area	High	Medium	Low
Library core	0	0	0
Cryptography	0	0	0
Infrastructure	0	0	0

Michelson parser	0	0	0
Code quality	0	0	0

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

- “Fixed”, or “Partially fixed” — 47 findings,
- “Scheduled, not fixed” — 0 finding,
- “Acknowledged, but won’t do” and “not an issue” — 0 findings.

Full report

The full report package consists of four separate files.

- The initial executive summary that provides a high-level overview of the performed audit and found issues.
- The updated version of executive summary (this document), produced after verification of the fixed issues.
- The initial technical report which describes the Dartez architecture, risk model, threat model, already implemented security controls, cryptography primitives inventory, cryptography operations review, application security review, code quality, Michelson parser analysis, review of supply chain risks.
- Appendix A provides technical details of each finding, its fix and verification status.

Conclusions

We’ve reviewed Dartez library’s source code, including handling of operations, such as fee estimation, RPC communication, key management, cryptography usage and the Michelson parser. We have also assessed the surrounding tools, example apps, tests and dependencies.

As the Dartez library is used as the cryptographic core for cryptocurrency wallets, it should not only protect the wallet's sensitive assets, but also provide a secure-by-default API for application developers to prevent misuse. We emphasise that auditing a library that implements a security function without also auditing the corresponding application leaves a lot of room for security flaws.

Our impressions after the initial audit

Our general conclusion is that the security controls implemented by the Dartez development team are insufficient to achieve the security claims and prevent risk statements to a high level. Our research concluded that the Dartez library contains a number of critical and high issues, the majority of which is related to the core functionality: transactions, cryptographic operations, and the Michelson parser.

The Dartez development team has implemented some security measures, such as creating developer-facing documentation and examples to protect against misuse, encrypting sensitive wallet keys in memory and decrypting them before use. However, in this instance, the encryption key is stored in a nearby memory value, rendering the encryption less effective.

25% of discovered issues are related to the cryptographic core of Dartez: ambiguous seed derivation, non-compliance with blockchain protocols, poor generation of encryption keys, the use of potentially biassed PRNG, and issues with cryptography 3rd party dependencies.

In addition, cryptocode does not follow typical crypto coding guidelines: no careful error handling, no tests for cryptographic operations, and usage of confusing naming, which makes it difficult to review, improve, and maintain. The cryptographic operations are scattered across numerous code modules, and some functionality is duplicated by third-party libraries.

Dartez has Michelson parsers, which take roughly 2/3 of the library source code. The parser is required, as Dartez works directly with Tezos contracts and provides functionality for their serialisation, origination, calling, etc.

25% of our findings are related to the parser: lack of testing, unsupported Michelson commands, potential code injections, unvalidated inputs, incomplete code. Considering all our findings, we cannot recommend using this parser in the real-world, as it could lead to exploiting the Dartez library, performing fake contracts, generating fraudulent transactions, etc.

Noting that our analysis was limited to the parser's usage within the Dartez library and not as a standalone parser, so the list of parser-related issues is not exhaustive. We can extrapolate that the parser code contains a magnitude more bugs. We strongly advise a re-assessment of the parser code once the current issues have been resolved.

~20% of our findings are related to dependencies. Dartez, as a library, could be a target for supply chain attacks against the application that will integrate it. At the same time, we found that Dartez library uses 14 direct third-party dependencies, only two of which are deemed “good enough” from a security perspective. We provided numerous recommendations on minimising the number of dependencies, and building the dependency management process.

As the primary objectives of this engagement were to not only to assess, but to improve the library security, we provided design and implementation recommendations aimed at not only in resolving the immediate issues, but laying the groundwork for the project's long-term stability and maintenance.

Our impressions after verification of fixed issues

The Dartez team was fixing the findings during March-May 2023. They have prioritised fixing the issues of Michelson parser first.

Michelson parser fixes: many issues were fixed at the root cause and some, like encoding messages, were delegated to the Tezos node. The developers took our suggestions into account and added a large number of parser tests with the real-world contracts. However, the parser code remains relatively intricate and challenging to comprehend, potentially leading to complications in future support of the library.

Code quality and dependencies management: the team implemented the limits in code, removed unused parameters and established the baseline for the dependency management process. The Dartez team removed all unused or superfluous dependencies, and replaced the orphaned packages with the new ones. For example, `flutter_sodium` was replaced with the `sodium_libs`, however some of the old binding for the web version of Libsodium remained in place, due to limited support of some functionality in `sodium_libs`.

Lastly, the team has addressed the cryptographic issues: eliminating some third-party code and replacing hand-crafted functionality (such as seed and key generation) with the functions provided by existing dependencies. Additionally, user experience enhancements were made through API modifications and restrictions, documentation updates, and ensuring seamless integration with other Tezos platform tools.

We would like to note that after fixing the issues, some areas of Dartez library could be improved further. Some code pieces are hard to follow, particularly the code of the Michelson parser, which was validated only through a large number of parser tests that address the found issues. The non-parser functionality test suite is still small, with only 16 test functions for 2000 lines of non-parser code. Furthermore, as of the time of writing, all fixes were still in the private GitHub repository, and the new Dartez library had not yet been released.

We thank the Dartez development team for their work and contributions to the project's improvement. The Dartez project has progressed and now provides better security guarantees.

Findings details

We triaged findings into categories depending on the area, type of issue and severity of the outcome.

Types: broken controls (security controls that are implemented but don't satisfy security requirements), missing controls (lack of protections), enhancements (security-related suggestions to implement).
Areas:

- Core (C) — issues in main functionality and logic.
- Crypto (CR) — cryptography related issues.
- Code quality (CQ) — typos, documentation and general code style
- Infrastructure (I) – supply chain, CI, testing
- Parser (P) - Michelson parser issues

The statuses of issues could be the following:

- **Acknowledged** – the issue is acknowledged by the Dartez team.
- **Fixed** – the issue was confirmed by the Dartez team, the fix was implemented in full, and validated by Cossack Labs team.
- **Partially fixed** – the issue was confirmed by the Dartez team, the fix was partially implemented and validated by Cossack Labs team.
- **Scheduled, not fixed** – the issue is confirmed by the Dartez team, planned and scheduled for later, currently not fixed.
- **Not an issue** – the issue was rejected by the Dartez team due to technological / platform limitations, making it impossible to solve; or the issue was a conscious design decision.

Findings and references

ID	Finding	Severity / priority	Type	Area	Status
C-001	Limit number of retries in "getOperationStatus"	High	missing	core	Fixed
C-002	"fee" is unused in some methods	High	broken	core	Fixed
C-003	"appendRevealOperation" doesn't estimates fee and gas if key is not revealed	High	broken	core	Fixed
C-004	Transactions: "TezosMessageUtils.readAddressWithHint" parses kt1 with error	High	broken	core	Fixed
C-005	Check http response codes	Med	missing	core	Fixed
C-006	"fee", "gas_limit" and "storage_limit" are ignored	Med	broken	core	Fixed
C-007	Reduce number of requests in "FeeEstimator"	Low	improve	core	Fixed
C-008	Warn users about unprotected connection	Low	improve	core	Fixed
CR-001	Fix Secp256k1 key derivation: don't use seed a private key, follow bip-32 instead	High	broken	crypto	Fixed
CR-002	Non-standard and ambiguous derivation of seed from mnemonic	High	broken	crypto	Fixed
CR-003	SignerCurve.SECP256R1 is unhandled	High	broken	crypto	Fixed
CR-004	Generate password allows empty passwords (password length has wrong param type)	High	broken	crypto	Fixed

ID	Finding	Severity / priority	Type	Area	Status
CR-005	Keys from "octez-client" cannot be used in Dartez	High	broken	crypto	Fixed
CR-006	"signOperationGroup" doesn't validate the key type	Med	broken	crypto	Fixed
CR-007	Don't encode public key by yourself	Med	improve	crypto	Fixed
CR-008	"TezosMessageUtils.simpleHash" ignores the size argument	Med	broken	crypto	Fixed
CR-009	Improve cryptographic design: use data classes instead of array for returning keys to avoid mistakes	Low	improve	crypto	Fixed
CR-010	Duplicate functionality of seed derivation	Low	improve	crypto	Fixed
P-001	Parser cannot parse escaped characters in strings: code injection could lead to RCE	High	broken	parser	Fixed
P-002	Parser: incorrect support of "list" in parameter and storage	Med	broken	parser	Fixed
P-003	Parser: preprocessor issues could cause malicious output	Med	broken	parser	Fixed
P-004	Parser: semicolon without space is not parsed	Med	broken	parser	Fixed
P-005	Parser doesn't support "view" sections	Med	missing	parser	Fixed
P-006	Parser: incorrect output type	Med	broken	parser	Fixed
P-007	Parser: wrong annotation handling	Med	broken	parser	Fixed

ID	Finding	Severity / priority	Type	Area	Status
P-008	Parser: contract code, storage and parameters use the same function	Med	broken	parser	Fixed
P-009	Parser: Michelson tokenizer issues	Med	broken	parser	Partially fixed
P-010	Parser is not finished: many TODO comments	Low	improve	parser	Fixed
P-011	Parser doesn't support some instructions	Low	missing	parser	Fixed
P-012	Parser: overview of current issues and precautions	Low	broken	parser	Fixed
I-001	"generateMnemonic" uses biased RNG	High	broken	infra	Fixed
I-002	Dartez from pub.dev is broken	High	broken	infra	Fixed
I-003	Lack of dependency and vulnerability management process	Med	improve	infra	Fixed
I-004	Replace "flutter_sodium" with "sodium_libs"	Med	improve	infra	Fixed
I-005	"sec" dependency is superfluous and potentially dangerous	Med	improve	infra	Fixed
I-006	Use of dependencies w/o copyright note	Med	broken	infra	Fixed
I-007	Suspicious dependencies	Low	improve	infra	Fixed
I-008	Seed derivation is superfluous	Low	improve	infra	Fixed
I-009	No CI for testing and dependency management	Low	improve	infra	Fixed

ID	Finding	Severity / priority	Type	Area	Status
I-010	Consider excluding "pubspec.lock" from version control system	Low	improve	infra	Fixed
I-011	"kepler" dependency is unused	Low	improve	infra	Fixed
CQ-001	Throw errors instead of printing	Med	missing	code	Fixed
CQ-002	Transactions: no error thrown by "encodeOperation" if operation doesn't exist	Med	broken	code	Fixed
CQ-003	SoftSigner.createSigner can accept inconsistent parameters	Low	improve	code	Fixed
CQ-004	Improve documentation	Low	improve	code	Partially fixed
CQ-005	Avoid "magic" values in code	Low	improve	code	Fixed
CQ-006	Example project cannot be built on Android	Low	broken	code	Fixed

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, software engineering, 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

0.1	28 February 2023	Cossack Labs team	Initial version of the executive summary, technical report and appendix with technical issues.
0.2	09 May 2023	Cossack Labs team	Updated executive summary based on the verification of fixes. Added "Our impressions after verification of fixed issues" section to conclusion. Updated statuses of issues based on fixes by the Dartez team: added verification status to each issue.