tzConnect's FA2 smart contract templates

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 13.01.2023

inference

Table of contents

Version / Date	Description
1.0 / 13.01.2023	Final version

Summary

Inference AG was engaged by Tezos Foundation to perform an independent security assessment of the tzConnect's FA2 smart contract templates[1].

Inference AG performed the security assessment based on the agreed scope, following our approach and activities as outlined in the "Project overview" chapter between 25th October 2022 and 09th December 2022. Feedback from the tzConnect team was received during the course of the assessment and Inference performed a follow-up assessment.

Based on our scope and our performed activities, our security assessment revealed a few observations, which, if resolved appropriately, may improve the quality of the tzConnect's FA2 smart contract templates.

[1] tzConnect's code repo is a fork with some changes from https://github.com/oxheadalpha/smart-contracts

Overview on issues and observations

Details for each reported issue or observation can be obtained from the "Security issues" and "Observations" sections.

Row header	Severity / Status
Security issues	
There are no open known security issues.	
Observations	
O-TFA-001: Documentation	- / open
O-TFA-002: Sending tez to contracts	- / open
O-TFA-003: Testing	- / open

Project overview

Scope

The scope of the security assessment was the following sets of smart contracts:

- FA2 - multiple asset version with compile entrypoint “multi_asset_main”: multi_asset/ligo/src/fa2_multi_asset.mligo
- FA2 - single asset version with compile entrypoint “single_asset_main” and no “owner hooks”: single_asset/ligo/src/fa2_single_asset.mligo

Our initial assessment considered commit “7ae89e8604145e94b6adc6dcd55f4f4403be5dfa” in the source code repository https://github.com/tzConnectBerlin/tqtezos-smart-contracts.

Our reassessment considered commit “6d47144a3fd552d24bca07083cf0e20bb3c7916d”.

Scope limitations

Our security assessment is based on the following key assumptions and scope limitations:

- Any potential adversarial activities conducted by the administrator of the contract or operational errors by administrators were out of scope.
- Content of metadata and token metadata were out of scope. For instance, any off-chain views have not been assessed.

Methodology

Inference's methodology for smart contract security assessments on Tezos is a combination of a source code review of the smart contract source code in the high-level language (e.g. Ligo or SmartPy), and the resulting compiled code in Michelson. This approach provides additional assurance that the compiler producing the Michelson code is correct, and does not introduce any security issues. Furthermore, this approach fosters a better understanding for smart contract users of what has been reviewed and what has been effectively deployed on-chain.

In order to ensure a high quality in our security assessments, Inference is using subject matter experts having a high adversarial scenario mindset to spot potential issues in smart contracts under review. Additionally, we apply checklists derived from good practices and commonly known issues based on the Tezos smart contract assessment checklist to document our work and to ensure good issue coverage.

Furthermore, Inference maintains regular communications with the smart contract development team to ensure a correct understanding of the smart contract solution and environment, but also to make teams aware of any observations as soon as possible.

Inference's internal quality assurance procedures ensure that results of security assessments are challenged for completeness and appropriateness by a second independent expert.

Objectives

The objectives are the identification of security issues with regards to the assessed smart contracts and their conceptual design and specification. The security assessment also focuses on adversarial scenarios on specific use cases which have been listed in the appendix named Adversarial scenarios. These were identified together with the tzConnect's FA2 smart contract template developers and checked during our security assessment.

Activities

Our security assessment activities for the defined scope were:

- Source code review of smart contract code in Ligo

We applied the checklist for smart contract security assessments on Tezos, version 1.1 obtained from https://github.com/InferenceAG/TezosSecurityAssessmentChecklist. We applied the following security checklist tables:

- System / Platform
- Storage
- Gas issues and efficiency
- Code issues
- Transaction
- Entrypoint
- Admin / Operator functions
- Other topics & test cases

Our activities for the follow-up assessment were:

- Source code review of changes in the smart contract code in Ligo,
- Reassessing security issues and observations from the initial assessment if claimed to be resolved.

Security issues

There are no open known security issues.

Observations

O-TFA-001: Documentation

Detailed documentation for the different smart contracts is missing.

Recommendation:
We recommend adding documentation/specifications which provides at least general information about the following:

- Description of what an entrypoint is for and what it should do
- Meaning and types of input parameters / arguments
- Pre-conditions (e.g. who can call the entrypoint)
- Post-conditions
- Returned operations (transfers, contract origination, etc.)

Comment from the tzConnect team:
These contracts will not be further developed so we do not regard making tests or documentation as critical

O-TFA-002: Sending tez to contracts

Entrypoints which do not expect any tez to be sent do not reject any sent tez. Thus, if tez are mistakenly sent with a transaction to the smart contract, then the sent tez are locked into the smart contract since there is no withdrawal function available.

This observation affects all versions of the smart contracts in scope.

Recommendation:
We recommend rejecting any sent tez in the case where the tez are not expected to be sent to an entrypoint. This should be at least implemented for non privileged entrypoints.

Comment from the tzConnect team:
We are OK with the possibility of users sending tez to the contract and as above we don't want to be making changes to these contracts at this late stage.

O-TFA-003: Testing

Only a few basic test cases are defined.

This observation affects all versions of the template smart contracts in scope.

Recommendation:
We recommend covering all entrypoints and code with appropriate test cases. Test cases should also include test cases expecting to fail in order to verify whether the contract is correctly working (e.g. access-restricted entrypoints or failing conditions).

Comment from the tzConnect team:
These contracts will not be further developed so we do not regard making tests or documentation as critical.

Disclaimer

This security assessment report ("Report") by Inference AG ("Inference") is solely intended for Tezos Foundation ("Client") with respect to the Report's purpose as agreed by the Client. The Report may not be relied upon by any other party than the Client and may only be distributed to a third party or published with the Client's consent. If the Report is published or distributed by the Client or Inference (with the Client's approval) then it is for information purposes only and Inference does not accept or assume any responsibility or liability for any other purpose or to any other party.

Security assessments of a software or technology cannot uncover all existing vulnerabilities. Even an assessment in which no weaknesses are found is not a guarantee of a secure system. Generally, code assessments enable the discovery of vulnerabilities that were overlooked during development and show areas where additional security measures are necessary. Within the Client's defined time frame and engagement, Inference has performed an assessment in order to discover as many vulnerabilities of the technology or software analysed as possible. The focus of the Report's security assessment was limited to the general items and code parts defined by the Client. The assessment shall reduce risks for the Client but in no way claims any guarantee of security or functionality of the technology or software that Inference agreed to assess. As a result, the Report does not provide any warranty or guarantee regarding the defect-free or vulnerability-free nature of the technology or software analysed.

In addition, the Report only addresses the issues of the system and software at the time the Report was produced. The Client should be aware that blockchain technology and cryptographic assets present a high level of ongoing risk. Given the fact that inherent limitations, errors or failures in any software development process and software product exist, it is possible that even major failures or malfunctions remain undetected by the Report. Inference did not assess the underlying third party infrastructure which adds further risks. Inference relied on the correct performance and execution of the included third party technology itself.

Appendix

Adversarial scenarios

The following adversarial scenarios have been identified together with tzConnect's FA2 smart contract template developers and checked during our security assessment.

Scenario	Impact rating & descriptive	Assessment result
Standard smart contract based on the FA2 specification (TZIP-012). No additional, specific adversarial scenarios identified.		

Risk rating definition for smart contracts

Severities are quantified with two dimensions, roughly defined as follows, whereas the examples have to be regarded as indication only:

Probability of occurrence / materialisation of an issue
(bullets for a category are linked with each other with “and/or” condition.)

- Low:
  - A trusted / privileged role is required.
  - Contract may end up in the issue if other conditions, which are also unlikely to happen, are required.
- Medium:
  - A specific role or contract state is required to trigger the issue.
  - Contract may end up in the issue if another condition is fulfilled as well.
- High:
  - Anybody can trigger the issue.
  - Contract’s state will over the short or long term end up in the issue.

Impact:
(bullets for a category are linked with each other with “and/or” condition.)

- Low:
  - Non-compliance with TZIP standards
  - Unclear error messages
  - Confusing structures
- Medium:
  - A minor amount of assets can be withdrawn or destroyed.
- High:
  - Not inline with the specification
  - A non-minor amount of assets can be withdrawn or destroyed.
  - Entire or part of the contract becomes unusable.

Severity:

	Low impact	Medium impact	High impact
High probability	**High**	**Critical**	**Critical**
Medium probability	**Medium**	**High**	**Critical**
Low probability	**Low**	**Medium**	**High**

Glossary

Term	Description
Archetype	High level smart contract language. Website: https://archetype-lang.org/
Ligo	High level smart contract language. Website: https://ligolang.org/
Origination	Deployment of a smart contract
SmartPy	High level smart contract language. Website: https://smartpy.io/
TZIP	Tezos Improvement Proposal