USDT token smart contract

Tezos Foundation

Independent security assessment report

inference

Report version: 2.0 / date: 09.05.2022

inference

Table of contents

Version / Date	Description
1.0 / 31.03.2022	Final version
2.0 / 09.05.2022	Version 2, which considers the update on the USDT token smart contract allowing updatable entrypoints.

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of the USDT token smart contract developed by Papers.

Inference AG performed the security assessment based on the agreed scope, following our approach and activities as outlined in the “Project overview” chapter between 15th March 2022 and 21th March 2022. This security assessment resulted in the issues reported in section Security issues and Observations.

Feedback from Papers was then received in March 2022 and Inference performed a follow-up assessment. Based on the activities in our follow-up assessment, we were able to state that our reported security issues and observations had all been resolved by Papers.

First week May 2022, we reviewed the changes of the USDT token smart contract, that implements updatable entrypoints, missing in the previous version. Based on our activities, we have not discovered any security issues or observations.

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
Issues	
S-TTC-001: FA2 non-compliance - transfer entrypoint	high / closed
S-TTC-002: FA2 non-compliance - non existing account	low / closed
S-TTC-003: Removing administrators	low / closed
Observations	
O-TTC-001: Documentation	- / closed
O-TTC-002: TZIP-016 metadata missing	- / closed
O-TTC-003: Testing	- / closed
O-TTC-004: Code optimization	- / closed

Project overview

Scope

The scope of the security assessment was:

- Token smart contract:
  - SmartPy code: tether_token.py and the included files in order to compile the smart contract
  - Michelson code: hangzhou / KT18m8eCUoj1zjQKhr9hpqjrDdXXUnCySDGu

The following files from the Github repo were also part of the assessment:

- README.md
- entrypoint.md

All files in scope were made available via a source code repo: https://gitlab.papers.tech/papers/customer-tezos-foundation and our initial security assessment considered the commit “ace199fcac0fc9cc7997c990de7060fe81866bb3”

In our follow-up assessment we considered changes to the initial scope set out above. Our follow-up assessment considered:

- SmartPy code: Gitlab commit: 1f019ef42c871384d716b17d449de274f8afa8bc
- Michelson code: hangzhou / KT1QkSNHX5q38yhbAfQhcoLRFa7VcKJkHPRX

Our assessment of the updated version 2.0 of the USDT token smart contract (upgradable entrypoints) considered the changes to the code analysed in the follow-up assessment:

- SmartPy code: Gitlab commit: c520e4037b96d67c782522106d0f73e8b234ee04
- Michelson code: ithaca / KT1NRodBppPQpfcmsPd2TcjHhjzinge4vbU7 including the code in upgradable entrypoints present at block level 462589 (origination operation).

Scope limitations

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

- Any potential adversarial activities conducted by the contract’s admins were out of scope.

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 appendix Adversarial scenarios. These were identified together with the Madfish 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
- Source code review of the deployed smart contract in Michelson
- Review of testing code

We applied the checklist for smart contract security assessments on Tezos, version 1.0 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 specific tests
- Admin / Operator functions
- Signature replay
- Other topics & test cases

Our activities for the follow-up assessment were:

- Source code review of changes in the smart contract code in SmartPy
- Source code review of changes in the deployed smart contract in Michelson
- Reassessing security issues and observations from initial assessment in case they are claimed to be resolved

Our activities for the assessment of the version 2 of the USDT token contract (upgradable entrypoints) were:

- Source code review of changes in the smart contract code in SmartPy
- Source code review of changes in the deployed smart contract in Michelson

Security issues

S-TTC-001: FA2 non-compliance - transfer entrypoint

The “transfer” entrypoint is not compatible with the FA2 standard (TZIP12). In particular, the ordering of input parameters does not follow the FA2 standard.

Thus, due to this non-compliance, other solutions/products can not automatically interact with this token smart contract. For example, existing DEXs will not be able to integrate this token smart contract.

Probability - High.
Impact - Low.
Severity - High.

Recommendation:
We recommend adapting the transfer entrypoint structure in order to be compliant with the FA2 standard (TZIP-012)

Comment from Papers:
Will be fixed.

Results from follow-up assessment:
Fixed. Status updated from “open” to “closed”.

S-TTC-002: FA2 non-compliance - non existing account

The USDT smart contract fails if a transfer of zero (“0”) tokens is requested from an address which does not exist in the ledger.

The FA2 standard (TZIP-012), in section “core transfer behaviour”, states:

- If the token owner does not hold any tokens of type `token_id`, the owner's balance is interpreted as zero. No token owner can have a negative balance (TZIP-012).
- Transfers of zero amount MUST be treated as normal transfers (TZIP-012).

Thus, based on these two claims, Inference’s interpretation is that such transfers must not fail.

This poses a risk that requested transfers are failing in certain situations/solutions interacting with the USDT token smart contract.

Probability - Low.
Impact - Low.
Severity - Low.

Recommendation:
We recommend analysing the situation and potentially adapting the USDT token smart contract in order to be compliant with the FA2 standard (TZIP-012)

Comment from Papers:
Will be fixed.

Results from follow-up assessment:
Fixed. Status updated from “open” to “closed”.

S-TTC-003: Removing administrators

The USDT token smart contract does not have any control in place to prevent the last registered administrator from being removed.

This poses a risk that the USDT token smart contract can no longer be administered if the last administrator gets accidentally removed.

Probability - Low.
Impact - High.
Severity - Low. (Note: down-rated, since we rate the possibility as negligibly low as long as the teams apply good practices in administration of the smart contract.)

Recommendation:
If the goal is to always have an administrator registered in the USDT token smart contract, we recommend considering a control in the smart contract ensuring the last administrator account cannot be removed.

If the goal is to cease administrator control at a later stage, this may be acceptable, but we recommend considering implementing a two-step procedure to fully cease administrator control.

However, in both cases, we recommend that, at the very least, the behaviour of the "remove_administrator" entrypoint is clearly documented.

Comment from Papers:
Will be fixed.

Results from follow-up assessment:
The code has not been changed, but the documentation has been amended. We consider this appropriate. Status updated from "open" to "closed".

Observations

O-TTC-001: Documentation

Documentation is available. However, we made the following observations:

- wrong descriptions in entrypoint.md for parameters of the entrypoints “set_administrator” and “remove_administrator”
- missing description for the order in which the operations in the data type “set” have to be returned by the lambda function executed in the entrypoint “execute” for it to be executed sequentially by the Tezos blockchain.
- missing description for the entrypoint “propose_administrator” and “revoke”.

It is also unclear, based on the documentation, why there are both a “burn” and a “revoke” entrypoint, which both behave exactly the same.

Recommendation:
We recommend improving the documentation.

Comment from Papers:
Will be improved.

Results from follow-up assessment:
The documentation has been updated: “burn” and “revoke” entrypoints are now implemented differently and documentation has been accordingly updated. Status updated from “open” to “closed”.

O-TTC-002: TZIP-016 metadata missing

The token smart contract does not implement TZIP-016 for contract metadata. The FA2 standard (TZIP-012) does not require the implementation of TZIP-016. However, it is unclear to what extent other solutions/products automatically and correctly support the token smart contract under review if the token smart contract does not implement TZIP-016.

Recommendation:
We highly recommend implementing TZIP-016 in order to ensure correct indexing, displaying, and interacting with the token smart contract.
Additionally, we also recommend using the TZIP-016 error feature in order to provide more detailed error descriptions.

Comment from Papers:
Will be added.

Results from follow-up assessment:
A big_map named “metadata” has been added. This data structure can be populated with data during origination. Status updated from “open” to “closed”.

O-TTC-003: Testing

The smart contracts in scope have defined several test cases in SmartPy. However, we noted:

- the entrypoints “set_administrator”, “propose_administrator”, “remove_administrator”, “execute”, “update_operators”, and “balance_of” are not covered at all.
- the entrypoint “mint” and “burn” do not check whether the “total_supply” of the token gets correctly updated.
- the entrypoint “transfer” does not have any checks to verify whether it is not possible to transfer tokens, if either the sender or the destination is a frozen account.
- the entrypoint “transfer_frozen_assets” does not have any checks to verify whether a transfer is not possible if the account is not frozen.

Recommendation:
We recommend defining and executing test cases in SmartPy appropriately covering all entrypoints.

Comment from Papers:
Test cases will be added in SmartPy.

Results from follow-up assessment:
The recommend SmartPy test cases have been added. Status updated from “open” to “closed”.

O-TTC-004: Code optimization

The SmartPy and the Michelson code for the entrypoints “transfer” and “transfer_frozen_assets” have a lot of code in common.

Recommendation:
We recommend implementing a dedicated function in SmartPy covering the common code and, furthermore, we also recommend defining this function as a lambda function.

Comment from Papers:
Will be considered.

Results from follow-up assessment:
The code sections have been placed into a lambda function. Status updated from “open” to “closed”.

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 Papers developers and checked during our security assessment.

Scenario	Impact rating & descriptive	Assessment result
Creation or destruction of tokens by others than smart contract admins.	High due creation / destruction of assets by not authorised personnel.	**Ok** Nothing identified.
Transfer of tokens by neither owner, authorised operator, nor smart contract admins.	High due to theft of assets.	**Ok** Nothing identified.
Transfer of freezed tokens by others than smart contract admins.	High due to bypassing potential legal / regulatory restrictions.	**Ok** Nothing identified.
Execution of admin functions by others than smart contract admins.	High in case of mint, burn, and unfreeze Medium in case of freeze Low in case of set_token_metadata.	**Ok** Nothing 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
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