TrustSwap by Team Finance

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 12.04.2023

inference

Table of contents

Table of contents **2**
Summary **4**
Overview on issues and observations 5
Project overview **6**
Scope 6
Scope limitations 6
Methodology 7
Objectives 8
Activities 8
Security issues **9**
Observations **10**
O-TFT-001: Use of appropriately designed tokens 10
O-TFT-002: Unsafe management of critical address 11
O-TFT-003: Lack of documentation 12
Disclaimer **13**
Appendix **14**
Adversarial scenarios 14
Risk rating definition for smart contracts 15
Glossary 16

Version / Date	Description
1.0 / 12.04.2023	Final version.

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Team Finance's TrustSwap smart contract.

Inference AG performed the security assessment based on the agreed scope, following our approach and activities as outlined in the "Project overview" chapter between the 22nd of February and 24th of March 2023. Feedback from the Team Finance 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 initial security assessment revealed several vulnerabilities and a few observations, which, if resolved appropriately, may improve the quality of Team Finance's TrustSwap smart contract.

Based on our activities in our follow-up assessment we can confirm that any reported security issues have been resolved. This report only shows remaining open or partly resolved security issues and 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
Security issues	
There are no open known security issues.	
Observations	
O-TFT-001: Use of appropriately designed tokens	- / partially closed
O-TFT-002: Unsafe management of critical address	- / open
O-TFT-003: Lack of documentation	- / partially closed

Project overview

Scope

The scope of the security assessment were the following TrustSwap smart contracts by Team Finance:

- Lock:
  - SmartPy code: lock.py
  - Michelson code: lock.tz
- Vest
  - SmartPy code: vest.py
  - Michelson code: vest.tz
- Staking:
  - SmartPy code: staking.py
  - Michelson code: staking.tz

Our initial assessment considered commit "fd3ccae6ecd9e7a5d67112a1874e03ea7074c105" in the source code repository https://github.com/trustswap/tf-tezos-smart-contracts .

Our reassessment considered commit "b9d157ae1499b14ff762d8abc03e110f36332667".

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.
- The oracles are deemed as trusted sources by Team Finance. Thus, potential risks of a misbehaving price oracle 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 the appendix named Adversarial scenarios. These were identified together with Team Finance's 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 SmartPy
- Source code review of smart contract code in Michelson

We applied the checklist for smart contract security assessments on Tezos, version 1.2 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 SmartPy,
- Source code review of changes in the smart contract code in Michelson, and
- 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-TFT-001: Use of appropriately designed tokens

The distribution of the reward tokens in the vest.py smart contract will leave some minimal amounts of reward tokens in the smart contract.

The vest smart contract allows adding any FA1.2 compatible smart contract as reward tokens. Thus, if the users of the vest add pools with a reward token having too few decimals the leftover in the smart contract may be of significant value.

Recommendation:
We recommend analysing the situation and potentially considering measures such as providing clear documentation and guidelines for users on the required characteristics of the reward token smart contract.

Comment from Team Finance:
Warning added in the code as a comment.

Results from follow-up assessment:
We updated the status only from “open” to “partially closed” in order to raise awareness of this situation to potential users of the vest smart contract.

O-TFT-002: Unsafe management of critical address

The “transferLock” entrypoint in lock.py allows the transfer of the ownership of a lock without intermediate steps.

This poses the risk that the ownership could potentially be transferred to an unusable address leading to locked funds in the contracts.

Recommendation:
While not being a security risk, there are multiple approaches we would recommend to improve the situation.
We recommend implementing a two-step procedure to change any critical privileged addresses. In the first step the current privileged address proposes a new address, followed by a second step in which the newly proposed address accepts this proposal. The change of the critical privileged address is only done after the acceptance at the second step.
As an alternative, it should be clearly stated to the user that the functionality presents this risk, and that it should be used with extreme care, only after checking that the new owner is a valid account that will be able to access the lock and the funds stored inside it.

At the very least we recommend analysing the situation, providing appropriate documentation and procedures so that users dealing with these contracts are not accidentally locking funds in the contract forever.

Comment from Team Finance:
Acknowledged.

Results from follow-up assessment:
No change.

O-TFT-003: Lack of documentation

There is a lack of documentation: in particular the following aspects are not covered or not sufficiently covered in our opinion:

- *Description of the smart contract, its use cases, and potential risks.*
  Missing information about the design and behaviour of the smart contracts. There is no explanation on how the smart contracts work and what the potential risks and constraints are. Please also see reported observations "O-TFT-001" and "O-TFT-002".
  Missing or wrong information about entrypoints. In particular, there is no description of what the parameters are. For instance, the description for the "lockTokens" EP does not describe in detail what the required parameters are and what they represent.

This observation affects both smart contracts in scope.

Recommendation:
We recommend creating meaningful and sufficient documentation in order for users to understand what the smart contracts are doing, what the risks are, and how they can be used.

Comment from Team Finance:
Various comments added to the code.

Results from follow-up assessment:
We updated the status from "open" to "partially closed", since some documentation has been added as comments in code.

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 Team Finance's TrustSwap smart contract developers and checked during our security assessment.

Scenario	Assessment result
Lock: Creating a "lock" with the purpose of extracting tokens of other "locks".	**Ok** Nothing identified.
Lock: Withdrawal of locked funds before defined time has passed.	**Ok** Nothing identified.
Vest: Withdrawal of more funds than allowed.	**Ok** Nothing identified.
Staking: Creating a "pool" with the purpose of extracting tokens of other "pools".	**Ok** Nothing identified.
Staking: Withdrawal of more funds than allowed.	**Ok** Nothing identified.
Staking: Withdrawal of more rewards than allowed.	**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
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.dev/
TZIP	Tezos Improvement Proposal