Ubinetic's flat curve swap smart contracts for Youves

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 12.08.2022

inference

Table of contents

Version / Date	Description
1.0 / 12.08.2022	Final version

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Ubinetic's flat curve swap smart contracts.

Inference AG performed the security assessment based on the agreed scope, following our approach and activities as outlined in the "Project overview" chapter between 10th July 2022 and 12th August 2022. Feedback from Ubinetic was received and Inference performed a follow-up assessment.

Based on our scope and our performed activities, our security assessment revealed several security issues rated from critical to low severity. Additionally, different observations were also made, which if resolved with appropriate actions, may improve the quality of Ubinetic's flat curve swap smart contracts.

This report only shows remaining open or partly resolved 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
Issues	
There are no open, known security issues.	
Observations	
O-UFC-001: Testing	- / open
O-UFC-002: FA12 non-conformance	- / open
O-UFC-003: Race condition in SetLqtAddress entrypoint	- / open

Project overview

Scope

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

- FA2 flat curve swap
  - contracts/swap/fa2_flat_cfmm.mligo
  - compiled contract in Michelson: __SNAPSHOTS__/compilation/swap/fa2_flat_cfmm.tz
- FA12 flat curve swap
  - contracts/swap/fa12_flat_cfmm.ligo
  - compiled contract in Michelson: __SNAPSHOTS__/compilation/swap/fa12_flat_cfmm.tz
- Liquidity pool
  - contracts/swap/liquidity_pool.mligo
  - compiled contract in Michelson: __SNAPSHOTS__/compilation/swap/liquidity_pool.tz

All files in scope were made available via a source code repo: https://gitlab.papers.tech/papers/ubinetic/sap-smart-contract and our initial security assessment considered commit “a705acd045b9cbed5e61cc8da0010326cf05eb1f”.

Additionally, we also had access to the following documentation in order to build up our understanding of the flat curve swap smart contracts: contract/swap/README.md

Our reassessment considered commit: a2f7b4771cbd77e39cf629af50402f5ed35bc660.

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.
- Smart contracts for tokens not in scope were considered trusted.

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 Ubinetic 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 smart contract in Michelson
- Review of testing code

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
- Transactions
- Entrypoint
- On-chain views
- 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
- Source code review of changes in the smart contracts in Michelson
- Reassessing security issues and observation from initial assessment in case they are claimed to be resolved

Security issues

There are no open known security issues.

Observations

O-UFC-001: Testing

The smart contracts in scope are lacking test cases.

We also noted for all smart contracts in scope that there are no on-chain test cases available.

Comment from Ubinetic:
We are in the process with ECAD Labs to automate deployments and testing using Taqueria.

O-UFC-002: FA12 non-conformance

The liquidity pool smart contract is not compliant with the FA12 specification TZIP-007 as follows:

- the error message “NotEnoughBalance” is a value of type NAT instead of a construct of (nat :*required*, nat :*present*), where *required* is the requested amount of tokens, *present* is the available amount.
- the error message “NotEnoughAllowance” is a value of type NAT instead of a construct of (nat :*required*, nat :*present*), where *required* is the requested amount of tokens, *present* is the current allowance
- The error message “UnsafeAllowanceChange” is a message of type STRING instead of the allowance value upon the contract call as of type NAT.

Furthermore, the liquidity pool smart contract does not follow the recommendation from the specification to store the token balances and allowances in a standardised indexable form.

Comment from Ubinetic:
Ligo only supports errors of type NAT or STRING, therefore it is a limitation of the implementation language.

O-UFC-003: Race condition in SetLqtAddress entrypoint

The SetLqtAddress can be called by everyone as long as the parameter lqtAddress has not yet been set. After setting the lqtAddress the entrypoint no longer can be used to set another lqtAddress.
Thus, depending on how smart contracts are originated and initialised, this may open a race condition where unauthorised third parties call the SetLqtAddress entrypoint first.

Comment from Ubinetic:
The SetLqtAddress is called at deployment time by us and it cannot be called subsequently. This way we assure the correct liquidity address will be set for the flat curve.

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 the Ubinetic developers and checked during our security assessment.

Scenario	Impact rating & descriptive	Assessment result
Swap “cash token” to “token”: Swapper is able to obtain more tokens for the provided cash tokens than these tokens are worth at the time of exchange.	High: Pool drainage.	**Ok** Nothing identified
Swap “cash token” to “token”: Swapper is able to prevent paying (part of) the exchange fee.	Medium: No income for LP	**Ok** Nothing identified
Swap “token” to “cash token”: Swapper is able to obtain more cash tokens for the provided tokens than these cash tokens are worth at the time of exchange.	High: Pool drainage.	**Ok** Nothing identified
Swap “token” to “cash token”: Swapper is able to prevent paying (part of) the exchange fee.	Medium: No income for LP	**Ok** Nothing identified
Remove liquidity: LP is able to withdraw more “tokens” than available LP tokens.	High: Pool drainage.	**Ok** Nothing identified
Remove liquidity: LP is able to withdraw more “cash tokens” than available LP tokens.	High: Pool drainage.	**Ok** Nothing identified
Providing liquidity, if DEX is not in balance, brings any advantages due to an unfair “lqt” distribution preferring one currency.	High: Incentives are not working.	**Ok** Nothing identified
Minting LQT tokens without providing any or enough liquidity.	High: Pool drainage.	**Ok** Nothing identified
Burning LQT tokens without having any or enough liquidity tokens.	High: Pool drainage.	**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