Plenty Labs - PlentySwap v3

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 21.09.2023

inference

Table of contents

Version / Date	Description
1.0 / 21.09.2023	Final version

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Plenty Labs' PlentySwap v3.

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 17th of July 2023 and the 13th of September 2023. Feedback from the Plenty Labs' team was received and Inference performed a follow-up assessment.

Based on our scope and our performed activities, our security assessment revealed several low-risk security issues. Additionally, different observations were also made, which if resolved with appropriate actions, may improve the quality of PlentySwap v3.

This report only shows remaining open or partly resolved issues and observations.

Overview on issues and observations

At Inference AG we separate the findings that we identify in our security assessments in two categories:

- Security issues represent risks to either users of the platform, owners of the contract, the environment of the blockchain, or one or more of these. For example, the possibility to steal funds from the contract, or to lock them in the contract, or to set the contract in a state that renders it unusable are all potential security issues;
- Observations represent opportunities to create a better performing contract, saving gas fees, integrating more efficiently into the existing environment, and creating a better user experience overall. For example, code optimizations that save execution time (and thus gas fees), better compliance to existing standards, and following secure coding best practices are all examples of observations.

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

	Severity / Status
Issues	
S-PLS-001: Huge price difference locks Segmented CFMM	low / open
Observations	
O-PLS-001: Documentation on limitations of price changes	- / open
O-PLS-002: Wrong computation for negative denominators	- / open

Project overview

Scope

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

- Segmented CFMM:
  - CameLIGO code: /src/ligo/core.mligo including all code & files to create the contract with entrypoint "main"
  - Michelson: /src/michelson/core.tz
- Factory:
  - CameLIGO code: /src/ligo/factory.mligo, including all code & files to create the contract with entrypoint "main"
  - Michelson: /src/michelson/core.tz
- Farm:
  - CameLIGO code: /src/ligo/farm.mligo, including all code & files to create the contract with entrypoint "main"
  - Michelson: /src/michelson/farm.tz

All files in scope were made available via a source code https://github.com/Plenty-network/plentyswap-v3, and our initial security assessment considered commit “40066e093066394b1e7fb40168c6f4d1558d64bd”.

Our follow-up assessment considered commit: “99f7dbb6806766f15fc80f3f10be0ac83ea4d0c4”.

Scope limitations

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

- Any potential adversarial activities conducted by the administrators/owners of the contract or operational errors by administrators/owners were out of scope.
- The key management of associated secret keys has not been assessed.
- Since anyone can deploy PlentySwap v3 smart contracts and have the initial ability to configure any token pair, we did not assess the potential risks associated with users interacting with such deployments that could potentially involve invalid or malicious tokens.
- Analysis of impact of the economic situation including the appropriate settings of initial or default values such as limits, tick spacing, etc., but also approximations.

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 Plenty Labs' team and checked during our security assessment.

Activities

Our security assessment activities for the defined scope were:

- Source code review of smart contract code written in CameLIGO
- Source code review of the smart contract in Michelson

We applied the checklist for smart contract security assessments on Tezos, version 2.0 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 CameLigo
- 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

S-PLS-001: Huge price difference locks Segmented CFMM

The Segmented CFMM (*CFMM from here on*) smart contract calculates the square root price based on an index value, utilizing a helper lookup function called “ladder”. This ladder function facilitates the calculation of square root prices up to about 5.88e+22. However, if the square root price exceeds this limit, the lookup in the ladder function will fail, resulting in the CFMM smart contract becoming locked.

Furthermore, the CFMM smart contract does not take into account the decimals of a token in its calculation. Consequently, the price limit can potentially be reached if there is a significant disparity in the values of the handled tokens and/or if there is a significant difference in the decimals of tokens (25 digits or more).

Probability - Low, even in Segmented CFMMs that have significant differences in decimal precision, this scenario requires extreme changes in prices.
Impact - High, the Segmented CFMM would be inaccessible and locked in its state once it reaches any of these extreme states.
Severity - Low, as the scenario is an extreme edge case.

Recommendation:
Our recommendation is to implement logic to assess the difference in precision for the tokens contained in the CFMM smart contract.

Comment from Plenty Labs:
We consider it highly improbable that the necessary price fluctuations to trigger a lockout of the CFMM would occur in any practical market scenario.

Observations

O-PLS-001: Documentation on limitations of price changes

The evaluation of the “floor_log_half_bps” functionality shows that the price change has to be contained in the range (0.7, 1.5). Any price change outside this range will cause the smart contract to fail.
When pointed out to Plenty Labs, they replied that this is an appropriate range for their use case.

Recommendation:
We suggest adding documentation regarding this limitation, to inform users of the platform.

Comment from Plenty Labs:
A warning has been documented in the swap section of the specification, which can be found here:
https://github.com/Plenty-network/plentyswap-v3/blob/master/SPECIFICATION.md#swaps

O-PLS-002: Wrong computation for negative denominators

The following function

let ceildiv_int (numerator : int) (denominator : int) : int = - ((- numerator) / denominator)

does not behave correctly in scenarios where denominator is a negative value, causing the overall computation to return inaccurate values.

We have not identified any security issues in the current code under review, since this ceildiv_int function is not feeded with negative value.

Recommendation:
Our recommendation is to implement an assert statement to ensure no wrong negative denominators are provided as parameters to the function.

Comment from Plenty Labs:
Given that the function is not utilised in situations where the denominator could be zero, and there are no anticipated future changes in the repository, we have opted to omit the assertion.

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 and checked during our security assessment.

Scenario	Impact rating & descriptive	Assessment result
As a normal user, add myself as an admin.	High	**Ok** Nothing identified that could circumvent the checks in the smart contract.
Influence the price by swapping large amounts to gain an economical advantage.	High	**Ok** Nothing identified.
Deploy a Segmented CFMM with tokens that have a huge difference in precision, to draw users to it and cause it to reach an extreme price value to lock users' funds.	High	**Not ok** See our reported security issue **S-PLS-001** While the attack vector is possible, the only gain would be to cause a financial loss to users of the Segmented CFMM. There is no economical gain for the attacker, making the attack scenario less feasible.
Receiving more tokens in a token transfer than the correct calculated amount of tokens.	High	**Ok** Nothing identified.
Attaining a greater liquidity position than the accurately calculated amount based on the provided tokens.	High	**Ok** Nothing identified.
Acquiring more tokens than the liquidity position, including rewards, would allow.	High	**Ok** Nothing identified.
Farm: Removing the stakes of another user, completely or partially.	High	**Ok** Nothing identified.
Farm: Withdrawing and claiming a higher reward than what is expected for the stake given.	High	**Ok** Nothing identified.
Farm: Participating in an incentive before it starts or after it is over.	High	**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
CFMM	Constant Function Market Maker
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