# Madfish's QuipuSwap 2.0

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 17.10.2022

inference

# Table of contents

| Version / Date | Description |
| --- | --- |
| 0.6 / 12.10.2022 | Draft version for discussion. Updated with mainnet contacts, including changes in auction mock and bucket contracts. |
| 1.0 / 17.10.2022 | Final version |

## Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Madfish's smart contracts for QuipuSwap 2.0.

Inference AG performed the security assessment based on the agreed scope, following our approach and activities as outlined in the "Project overview" chapter between 16th May 2022 and 3rd October 2022. Feedback from Madfish was received during the course of the assessment and Inference performed several follow-up assessments.

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 QuipuSwap 2.0.

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.

| | Severity / Status |
|---|---|
| **Security issues** | |
| There are no open, known security issues. | |
| **Observations** | |
| O-MQS-001: Documentation | - / open |
| O-MQS-002: Tez potentially locked in dex core | - / open |
| O-MQS-003: Testing | - / partly resolved |

# Project overview

## Scope

The scope of the security assessment was the following set of smart contracts for QuipuSwap 2.0:

- Auction:
  contracts/main/auction.ligo and the included Ligo files in order to compile the smart contract
- Auction mock
  contracts/main/auction_mock.ligo and the included Ligo files in order to compile the smart contract
- Baker registry:
  contracts/main/baker_registry.ligo and the included Ligo files in order to compile the smart contract
- Bucket:
  contracts/main/bucket.ligo and the included Ligo files in order to compile the smart contract
- Dex core:
  contracts/main/dex_core.ligo and the included Ligo files in order to compile the smart contract
- Flash swap proxy:
  contracts/main/flash_swaps_proxy.ligo and the included Ligo files in order to compile the smart contract

All files in scope were made available via a Github repo: https://github.com/madfish-solutions/quipuswap-core-v2/ and our initial security assessment considered the commit “773b7ae867cafb5ccbc24eb89c1a64a4e3a12c72”.

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

- Readme.md
- Testing files
  - test/yToken.test.ja and its included files in order to perform defined tests.
  - integration_tests/test_price_feed.py and its included files in order to perform defined tests.

Additionally, we also had access to the following documentation in order to build up our understanding of QuipuSwap 2.0: https://docs.quipuswap.com/smart-contracts/dex-2.0.

In our follow-up assessment we considered changes to the initial scope set out above. Our follow-up assessment considered the commit: “7041159880d70d60822c4f5c481e74f02f850b5e”.

The Michelson code for the following Tezos mainnet contracts were also considered and assessed:

- KT1M8tp17Y2GWsYrZDBRXqdRNawgwyFDzmp4 for auction mock
- KT1F8oyjko98F1L4323zbtBZiREZDcSQPJDi for baker registry
- Bucket (included in dex_core)
- KT1J8Hr3BP8bpbfmgGpRPoC9nAMSYtStZG43 for dex core
- KT1LyF39wwaEWhwC8KyeC46SWtvmVi31iMd1 for flash swap proxy

### 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 were regarded as 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
- 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
- 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 the deployed 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-MQS-001: Documentation

The contract entrypoints and its parameters are described sufficiently in QuipuSwap2.0 documentation. However, we noted a lack of documentation with regards to:

- Swaps with lambdas which have no upfront costs (aka “flash swaps” in QuipuSwap 2.0):
  Information for QuipuSwap 2.0 users on how these swaps work and their usage by users should be added. In particular, a list of “dos and don’ts” with regards to good practices on how to use these kinds of swaps should be added. Some of the “dos and don’ts” can be obtained from the section Adversarial scenarios in this report and in O-MQS-002: Tez potentially locked in dex core.
- Launching exchanges for new currency pairs:
  Information and guidance for QuipuSwap 2.0 users on launching exchanges for new currency pairs is missing. Documentation should include details about risks, characteristics of tokens (fa12 / fa2 conformity, precision, etc.), liquidity, etc.

*Recommendation:*
We recommend adding this missing information and guidance for QuipuSwap 2.0 users.

*Comment from Madfish:*
The documentation is an important part of the product, it will be improved along and after release to make its usage clear and integrations easier.

## O-MQS-002: Tez potentially locked in dex core

If a user returns more tez in swaps which do not have any upfront costs (aka “flash swaps” in QuipuSwap2.0) then the tez in excess of the required amount are locked in the dex core smart contract. Thus received tez by dex core can neither be claimed by the users nor the QuipuSwap admins.

*Recommendation:*
We recommend clearly documenting and advising users on how swaps with no upfront costs work and how to execute them.

*Comment from Madfish:*
This point will be covered in the updated documentation according to the suggestion.

## O-MQS-003: Testing

Various test cases are defined for the smart contracts in scope. However, we observed that some entrypoints are not covered by test cases at all and we also identified several test cases which should be added in order to fully cover the smart contract function and its correctness.

For more details please see our reported results in our provided work documentation.

*Recommendation:*
We recommend covering all entrypoints and code with appropriate test cases.

*Comment from Madfish:*
Having limited resources, we try to ensure the security of the sensitive parts of the code, some minor obvious things can be omitted.

*Results from follow-up assessment:*
Partly resolved. Missing important test cases have been added. However, there are still sections of code which are not fully covered. Status changed from "open" to "partly resolved".

# 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 Madfish developers and checked during our security assessment.

**dex core / baker registry / bucket / flash proxy swap:**

| Scenario | Impact rating & descriptive | Assessment result |
|---|---|---|
| Get a higher share than the invested tokens would grant. | High: Loss of funds by an inappropriate distribution of values between users. | **Ok**<br>Nothing identified. |
| Get more tokens/tez back than the share would grant. | High: Loss of funds by an inappropriate distribution of values between users. | **Ok**<br>Nothing identified. |
| Register an already existing currency pair. | Low: Dilution of liquidity | **Ok**<br>Nothing identified. |
| Obtain more baker rewards than the share would grant. | High: Loss of funds by an inappropriate distribution of values between users. | **Ok**<br>Nothing identified. |
| Obtain more tez interface fees than the accumulated fees. | High: Loss of funds. | **Ok**<br>Nothing identified. |
| Obtain more token interface fees than the accumulated fees. | High: Loss of funds. | **Ok**<br>Nothing identified. |
| Obtain a higher auction fee withdrawal reward. | High: Loss of funds. | **Ok**<br>Nothing identified. |
| Having more baker votes than the share would grant. | Medium: Loss of incoming baking rewards | **Ok**<br>Nothing identified. |
| Permit<br>Submission of a permit allowing you to make a transfer on behalf of somebody else. | High: Loss of assets. Loss of trust/reputation. | **Ok**<br>Nothing identified. |

| | | |
|---|---|---|
| Permit<br>Guessing permits in order to execute a transfer earlier than intended or to execute also in case where the transfer should not be executed. | Medium: Unwanted or prematurely executed transfers. | **Note**<br>Transfer permits can be guessed**.** |
| Permit<br>Preventing the registration of a permit. | Low: No transfers via permit possible | **Note**<br>Permit counter is increased with every submitted permit. Thus, a permit may not be registered if the counter has already been increased in the meantime. |
| Permit<br>A single permit can be used multiple times. | High: Loss of assets. Loss of trust/reputation. | **Ok**<br>Not possible, since the permit is removed once used. |
| Permit<br>Using the permit feature for functions other than transfers. | Unknown: Depends on the function. | **Ok**<br>The permit feature can only be used for transfers. |
| Re-enter entrypoints on dex_core contract (especially in swaps using lambdas). | High: Potential abuse of wrong dex_core states leading to a loss of funds in the DEX. | **Ok**<br>All public entrypoints on dex_core have reentrancy protection implemented. |
| Assets stored or in custody of the flash_swaps_proxy are lost or stolen. | High: Loss of funds. | **Note**<br>The flash_swaps_proxy contract does not store any tez and also has no tokens in custody. Furthermore, the contract fails if tez are sent to the default entrypoint.<br><br>However, depending on what QuipuSwap 2.0 users are doing, tokens may be in custody with the flash_swap_proxy.<br><br>Thus QuipuSwap 2.0 users take upon themselves the responsibility of transferring any tokens in custody and/or remove any authorisation on other contracts (e.g. approve/operator permissions) within their executed lambda call.<br><br>In general we recommend using an own fully-controlled smart contract for interaction and restrict the use of the flash_swaps_proxy contract in order to call the own contract.<br>This way potential airdrops or similar actions are not in custody of the flash_swap_proxy and can be extracted by any flash_swap_proxy user. |

| | | |
|---|---|---|
| (Ab)using flash_swaps_proxy if the contract has been set as an authorised contract in another entrypoint. | Unknown: Depends on the functionality of the entrypoint. | **Note**<br>The flash_swaps_proxy is not used in any entrypoint in scope of this review as a privileged / authorised contract.<br><br>We recommend that QuipuSwap 2.0 users never use the flash_swaps_proxy contract as a privileged / authorised contract in their own smart contracts. |

**auction:**

| Scenario | Impact rating & descriptive | Assessment result |
|---|---|---|
| Getting auctioned tez/tokens without providing any Quipu tokens or providing fewer Quipu tokens than the defined min bid amount, or the current bid. | High: Loss of assets. Loss of trust/reputation. | **Ok.**<br>Nothing identified. |
| Stealing Quipu tokens, locked in a bid. | High: Loss of assets. Loss of trust/reputation. | **Ok.**<br>Nothing identified. |
| Locking or destroying any auctioned tez/tokens or Quipu tokens (bids). | High: Loss of assets. Loss of trust/reputation. | **Ok.**<br>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 |