DAO for Youves

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 2023-05-17

inference

Table of contents

Version / Date	Description
1.0 / 17.05.2023	Final version.

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Youves' DAO .

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 2023-04-21 and the 2023-04-28. Feedback from Youves' development team was received and Inference performed a reassessment.

Based on our scope and our performed activities, our security assessment revealed security issues. Additionally, different observations were also made, which if resolved with appropriate actions, may improve the quality of Youves' DAO.

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
Security issues	
There are no open known security issues.	
Observations	
O-YDA-001: Potential gas optimizations	- / open
O-YDA-002: Unhandled tez payment	- / open

Project overview

Scope

The scope of the security assessment was the DAO smart contract:

- contracts/dao/dao.py and the included SmartPy files in order to compile the smart contract with the “DAOContract” as a compilation target
- compiled contract in Michelson: __SNAPSHOTS__/compilation/dao/all/DAOcontract/step_000_cont_0_contract.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 7bf3040e5eea6691dda91428e42d9cb90d618351.

Our reassessment considered commit: 4108b627eb12348b85d9b14cabc95a1f9fdfed13

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 and the DAO 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 development team and checked during our security assessment.

Activities

Our security assessment activities for the defined scope were:

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

Security issues

There are no open known security issues.

Observations

O-YDA-001: Potential gas optimizations

During the review of the code, some opportunities to improve the performance of the smart contract were identified. If properly implemented, these improvements would allow to save gas costs when using the contract.

Most improvements were implemented, the following one is the only one that has not been taken into account as outlined in our recommendation:

Recommendation
The variables used to track yes, no and the total number of votes in the “end_vote” EP are computed several times in the function, as they are not stored in an “sp.local”. This results in a much larger Michelson code and higher gas costs.

Comment from Youves:
The end vote entrypoint is called very sparsely so we are okay with it.

O-YDA-002: Unhandled tez payment

The entrypoint “propose” accepts tez payments even when the escrow amount is paid using FA2 or FA1.2 tokens.

In the current scenario, if users mistakenly sent tez with their contract call, the funds would be temporarily locked in the contract, and it would require a proposal to return them to their owners.

Recommendation
Two possible approaches can be taken to mitigate the issue:

- Modify the entrypoint code so that it checks whether the call should contain tez, and revert in case unnecessary funds have been sent to the smart contract;
- Report to the users of the platform that they should take care when performing transactions and make them aware of the risks of sending unnecessary funds to the smart contract.

Comment from Youves:
We will make our users aware of this, but the funds are never locked completely as they can always be transferred by a proposal lambda.

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	Assessment result
Render DAO contract unusable e.g. by submitting a malformed proposal.	**Ok** Nothing identified.
Potential DoS, if either the escrow amount or the minimum threshold are set to a value considered sufficiently low by an attacker: • An attacker willing to sacrifice the escrow amount could constantly submit fake proposals, clogging the mechanism and making it impossible to submit anything useful	**Ok** The team is aware of the issue and they will pick what they believe to be appropriate values to make this scenario economically unfeasible.
Potential race condition: the execution and cancellation periods can overlap. If the author of the proposal is not quick to execute their proposal and waits for a long enough period of time, there might be a scenario where the baker can determine the future of a proposal by picking either the execution or the cancellation request and inserting them in the block.	**Ok** The team is aware of this issue and they will pick long periods of time, to allow the author to have plenty of time to execute their proposal before they risk cancellation.
An attacker might be able to lower the quorum threshold up to its admissible minimum by constantly submitting useless proposals, exploiting the fact that users often do not vote when not interested. This might lower the threshold so that a coordinated group of attackers can then manage to pass a potentially dangerous proposal.	**Ok** The team is aware of the issue, but they believe that an elevated escrow amount is a sufficient countermeasure to prevent fake submission by potential attackers.

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