Ubinetic's Commitment Pool v2 & On-Demand Oracle for Youves

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 17.06.2024

inference

Table of contents

Version / Date	Description
1.0 / 17.06.2024	Final version

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Ubinetic's Commitment Pool v2 & On-Demand Oracle for Youves.

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 28th of September 2023 and the 17th of June 2024. Feedback from Ubinetic's team was received and Inference performed a reassessment.

Based on our scope and our performed activities, our security assessment revealed several findings which, if resolved with appropriate actions, may improve the quality of Ubinetic's Commitment Pool v2 & On-Demand Oracle for Youves.

This report only shows remaining open or partly resolved findings.

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	
UCO-001: Kickout mechanism leads to gas war scenario	- / open
UCO-002: Removing administrators	- / open
UCO-003: Results of views are potentially outdated	- / open

Project overview

Scope

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

- Commitment Pool v2
  - contracts/tracker/commitment_pool_v2.py
  - Commit: b06824da25396b0947ed291344cdeb03318a5c26
- On-Demand Oracle
  - contracts/oracle/on_demand_oracle.py
  - Commit: 142e8f2687f6f03f4b9198324a47ae585d07a03f

All files in scope were made available via a source code repo: https://gitlab.papers.tech/papers/ubinetic/sap-smart-contract.

The scope of our reassessment was:

- Commitment Pool v2
  - contracts/tracker/commitment_pool_v2.py
  - Michelson code: KT1ENJfVK98HTRfxjFKPqu1R7tFwXVjmLAo2 on Tezos mainnet
  - Commit: b8690138954072bde39b03bb585a019b0c76da4e
- On-Demand Oracle
  - contracts/oracle/on_demand_oracle.py
  - Michelson code: __SNAPSHOTS__/compilation/oracle/all/OnDemandOracle/step_000_cont_0_contract.tz
  - Commit: 9166f0aec1847b4c33942d26f4cf63c4942873df

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.
- Any potential centralization risks 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 Ubinetic's 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 SmartPy
- Review of testing code

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 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

UCO-001: Kickout mechanism leads to gas war scenario

When using the kickout functionality to remove a user that has overstayed their staking period, the reward factor is re-computed at the start of the “internal_kickout” function.

As a result of this approach, the later a user is kicked out, the more rewards are gained by the kicker: this leads to a waiting game scenario, where, as soon as someone inserts a “kickout transaction” in the mempool, a gas war starts in order to be included as the first kicker, and reap all the benefits of the reward mechanism.

Recommendation:
Different approaches can solve this issue. A few suggestions include revising the reward mechanism, so that all kickers who attempted to kick a user out in the same block share the reward, or to implement front-running protections.

Comment from Ubinetic:
The kickout mechanism is used to force someone out if their cooldown period expired, otherwise they can stay in the pool and collect rewards indefinitely. The gas war scenario from our perspective is not an issue, because both would lead to the expected outcome of having the account with expired cooldown period being kicked out of the pool.

UCO-002: Removing administrators

The entrypoint “remove_administrator” allows removing all admins, which would make the contract no longer administrable.

Recommendation:
We recommend analysing the situation, ensuring with appropriate documentation and procedures that admins dealing with these contracts are not mistakenly removing the last existing admin.

Comment from Ubinetic:
The remove administrator entrypoint allowing to remove the last admin is not a blocking issue as the contract will still function as intended even without an admin. Moreover, the contract will have the DAO as the admin which should allow us plenty of time to catch this in case there will be a proposal to remove the last admin.

UCO-003: Results of views are potentially outdated

The smart contracts in scope implement on-chain views that might not always reflect the most recent data.

In a scenario where “update_parameters” has not been executed for a while, the result of the views “get_accumulated_rewards” and “get_voting_details” might return values that are outdated.

Recommendation:
We recommend making sure that users get up to date values, or at least that they are aware of the potential risk they take when relying on their return values.

Comment from Ubinetic:
We added the update_parameters EP to circumvent this issue, but it doesn’t always solve the issue as the view can be called without updating the parameters first.

We updated the status from “open” to “partially closed”. We have not closed the issue to raise awareness of this situation among potential users of those on-chain views.

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
Commitment - commit entrypoint: Set a cooldown period longer than the max defined cooldown period.	**Ok** Nothing identified.
Commitment - commit entrypoint: Set a very short cooldown period.	**Ok** Possible, but leads to no or a very small stake weight.
Commitment - commit entrypoint: Extend cooldown periods of other users.	**Ok** Nothing identified.
Commitment - commit entrypoint: Shorten the duration of an existing stake.	**Ok** Nothing identified.
Commitment - commit entrypoint: Reduce staked amount of existing stake.	**Ok** Nothing identified.
Commitment - commit entrypoint: Create a stake with an amount below the minimum stake amount.	**Ok** Nothing identified.
Commitment - commit entrypoint: Commit again when stake is in cooldown and/or kickout phase.	**Ok** Nothing identified.
Commitment - enter cooldown entrypoint: Enter a cooldown for another user.	**Ok** Nothing identified.
Commitment - enter cooldown entrypoint: Bypass cooldown period.	**Ok** Nothing identified.
Commitment - enter cooldown entrypoint: Restart/reset cooldown period.	**Ok** Nothing identified.
Commitment - withdraw entrypoint: Withdraw before the cooldown duration has expired.	**Ok** Nothing identified.

Commitment - withdraw entrypoint: Withdraw if not in the cooldown period.	**Ok** Nothing identified.
Commitment - withdraw entrypoint: Withdraw for a different user.	**Ok** Nothing identified.
Commitment - kickout entrypoint: Kickout a user whose kickout period has not started yet.	**Ok** Nothing identified.
Commitment - kickout entrypoint: Prevent kickout.	**Ok** Nothing identified.
Oracle: Bypass the check on the number of valid certificates required.	**Ok** Nothing identified.
Oracle: Set the new lambda to execute without having the appropriate permission to do so.	**Ok** Nothing identified.
Oracle: Add myself or any other random address as a valid data source.	**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
Origination	Deployment of a smart contract
SmartPy	High level smart contract language. Website: https://smartpy.io/
TZIP	Tezos Improvement Proposal