Teia community marketplace and multisig smart contracts

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 04.02.2022

inference

Table of contents

Version / Date	Description
1.0 / 04.02.2022	Final version for publication

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Teia community marketplace and multisig smart contracts.

We performed the security assessment based on the agreed scope, following our approach and activities as outlined in the "Project overview" chapter from 20th December 2021 to 5th January 2022. Feedback from the Teia community was received in January 2022 and Inference performed a follow-up assessment.

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	
S-TEI-001: Calls to untrusted addresses	low / open
S-TEI-002: Removing users from voting	low / open
Observations	
O-TEI-001: Testing	- / partially closed
O-TEI-002: Error messages	- / closed
O-TEI-003: Changing voting parameters	- / open
O-TEI-004: Documentation	- / partially closed

Project overview

Scope

The scope of the security assessment was the Teia community marketplace and the multisig smart contract from Teia community developers as defined by the following:

- marketplace:
  - https://github.com/teia-community/objkt-swap/blob/3.0.0/smart-py/marketplace.py
  - https://github.com/teia-community/objkt-swap/blob/3.0.0/smart-py/marketplace_test.py
  - Initial assessment:
    - Commit: 95fae892018a23b120e5979b36ab4f53bda89be8
    - Deployed version: hangzhounet2 / KT1EtJbLgMVkXv3W1c2wvpWDkzw8TFczrkid
  - Follow-up assessment:
    - Commit: 916b9355b9ae98bc7e34f1f5a3203e8f80332517
    - Deployed version: mainnet / KT1PHubm9HtyQEJ4BBpMTVomq6mhbfNZ9z5w
- multisig:
  - https://github.com/jagracar/tezos-smart-contracts/blob/1.0.0/python/contracts/multisignWalletContract.py
  - https://github.com/jagracar/tezos-smart-contracts/blob/1.0.0/python/tests/multisignWalletContract_test.py
  - Initial assessment
    - Commit: 7e9fe183f64e160c43e23adcbb63a79d4b94d331
    - Deployed version: hangzhounet2 / KT1BjhEQQM5q5tN1QSPvfLU4f5bFEG3N8bz3
  - Follow-up assessment:
    - Commit: d0845f3d0ff80e00461e98809fcf08123570fdc4
    - Deployed version: mainnet / KT1PKBTVmdxfgkFvSeNUQacYiEFsPBw16B4P

The following additional information was also made available for our security assessment:

- None

Scope limitation

Any potential adversarial activities conducted by the contract's manager role were out of scope. However, during the security assessment, the following adversarial activities by the contract admin have been identified as possible:

- Implement an additional management fee check to filter out transactions with excessively low management fees (similar to S-TEI-001: Calls to untrusted addresses)

In addition to the above generally out-of scope item, we would like to add the following specific limitation:

- The creator address and the royalties can be defined by the swap issuer in the marketplace's "swap" entrypoint. It follows that, at a smart contract level, there are no checks verifying that the creator and royalties have been correctly filled in. Thus, based on this chosen design, our assessment does not list any observations or issues concerning this specific aspect.

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 beens listed in appendix Adversarial scenarios. These were identified together with the Teia community 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 SmartPy
- Source code review of the deployed smart contract in Michelson
- Review of unit testing code in SmartPy

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
- Contract origination and initialisation
- 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 SmartPy
- Source code review of changes in the deployed smart contract in Michelson
- Reassessing security issues and observation from initial assessment in case they are claimed to be resolved

Security issues

S-TEI-001: Calls to untrusted addresses

The marketplace entrypoint “collect” emits the transfer of the swapped object, the payment of the management fee, the payment of the royalties to the “creator”, and the payment of the price to the “swapper”. The marketplace contract does not prevent the destination addresses for the payment transfers from being smart contracts therefore these destinations could influence the control flow.

“Creator” could implement the following control flow scenarios:

- implement a royalties filter where they do not allow any transactions when the royalty is below a certain threshold and where they are registered as a creator.
- Fail all transactions where they are registered as a creator.

For “swapper”, there are no scenarios which could give them additional possibilities other than the ones they already have (cancel swap).

In general, reentrancy to the smart contract is possible, but the contract’s design does not open any adversarial reentrancy attacks.

This observation affects the marketplace smart contract in scope.

Probability - Low
Impact - Low
Severity - Low

Recommendation:
We recommend analyzing the risks of this issue in detail. If the risks can not be accepted a potential solution to mitigate is to check when a swap is created whether the destination addresses are implicit addresses.

Comment from Teia community developers:
We agree that the possibility of a creator using a dedicated smart contract to block certain collects is real. However, we think that it is very unlikely that this happens, since this will play against the creator. If their OBJKTs are difficult to resell, collector will not buy them in the first place.

The artist will also need to create their own smart contract to implement those checks and the smart contract will need to be able to mint OBJTKs in order to be the OBJKT creator.

Even if a creator decides to take this approach, collectors will be able to trade their OBJKTs transferring the tokens directly between the involved parties, or using another site where transfers to KT contracts are not allowed. The result will be that the artist will not receive any royalty at all, which again makes this scenario very unlikely.

We cannot block KT addresses from being the OBJKT creators, because we are already using KT addresses to manage artists collaborations. H=N offers the possibility to artists to collaborate using a dedicated smart contract that stores the collaboration parameters: artists involved, royalties splits, etc. OBJKTs minted under the collaboration have the collaboration contract address as the creator of the OBJKT. When the royalties are sent to the collaboration contract, it calculates the percentage corresponding to each artist and transfers them their respective tez amounts. Not allowing KT addresses as creators would block this site functionality, so we cannot consider it as an option.

Results from follow-up assessment:
This finding has been duly assessed by the Teia community developers and the developers decided to keep the current design. No follow-up assessment performed. Status remains on “open”.

S-TEI-002: Removing users from voting

Since the user to be removed can vote for his/her removal, this may lead to a problem in cases where (inactive) users have lost access to their keys. Thus, these inactive users now have to become active in order to reach quorum in proposals after the user has been removed.

For instance, removing a regularly participating user in a 4-of-5 voting system, will end up in a 4-of-4 voting system, where all remaining users are now crucial.

This observation affects the multisig in scope.

Probability - Low
Impact - High
Severity - Low (Manually overridden from "high" to "low", since we rate the probability as very low)

Recommendation:
We recommend that voting on a "removing user proposal" is not possible for the user subject to the removal.

At least we recommend documenting this risk and clearly outlining the (off-chain) procedures in order to ensure a secure handling of private keys and the secure removal of users.

Comment from Teia community developers:
The proposed scenario is valid and we agree that it could be difficult in some specific cases to reach quorum when some users are inactive or have lost access to their keys. There is however another scenario where a fraction of the users could decide to remove a legitimate user from the multisign against their will. In that case, we think the affected user should be allowed to defend themselves with their vote.
Since it's not possible to deduce inside the smart contract the intention of a remove_user proposal, we think giving the user the possibility to vote is the best solution.
We agree that the problem of inactive users is real and the election of an appropriate minimum_votes parameter value is critical for the correct use of the multisign. We will try to document all these possible problems in the multisign web interface and in dedicated wiki pages.
Results from follow-up assessment:

This finding has been duly assessed by the Teia community developers and the developers decided to keep the current design. No follow-up assessment performed. Status remains on “open”.

Observations

O-TEI-001: Testing

Various test cases are defined for the smart contracts in scope. However, we identified a few scenarios / test cases which should be added in order to fully cover the smart contract's function and its correctness.

This observation affects all smart contracts in scope.

Recommendation:
We recommend adding the following test cases to the smart contracts:

marketplace:

- Checking whether adding a swap with `royalties` higher than 250 is not possible.
- Checking whether the correct amounts have been sent to the correct destination by the "collect" entrypoint (management fee to "admin", royalties to "creator", and buying price to swap issuer).

multisig:

- Covering the default entrypoint in a test case.
- Checking whether the correct amounts have been sent to the correct destination when executing a "`transfer_mutez_proposal`".
- Checking whether double votes (neither positive nor negative votes) are not possible.

Furthermore, we highly recommend defining and implementing on-chain testing for smart contracts.

Comment from Teia community developers:
We agree and we will add those tests.

Results from follow-up assessment:
The recommend SmartPy test cases have been added. An on-chain testing framework has not been built. Status set from "open" to "partially closed".

O-TEI-002: Error messages

Long error messages increase the storage costs which have to be paid during the contract's origination. Furthermore, increased gas costs result from each entrypoint call due to loading a larger storage than strictly necessary.

This observation affects both smart contracts in scope.

Recommendation:
We recommend defining unique error codes for each type of error in a smart contract. In this way, should an error occur, only the corresponding unique error code has to be returned, which can be looked up in an error code map stored offline.

Comment from Teia community developers:
We understand the problem and will modify our error messages with error codes that will be documented somewhere off-chain.
Our initial reasoning was to provide meaningful messages in the users wallets (Temple, Kukai) when a given transaction fails or is rejected. However, it is true that we can try to intercept those errors on the site and replace them with appropriate messages.

Results from follow-up assessment:
Smart contracts are now using error mnemonics, whose detailed error messages can be looked up in the error fields of the TZIP-16 metadata fields. Status set from "open" to "closed".

O-TEI-003: Changing voting parameters

Changing the global parameters “expiration time” or “minimum votes” changes the conditions for all still ongoing proposals.

For instance, decreasing or increasing the minimum votes parameter will respectively lower or increase the quorum required for proposals which are already ongoing

Furthermore, removing a user does not remove the votes which have already been posted on ongoing proposals of this particular user.

Adding new users allows them to vote for proposals which are already ongoing.

This observation affects the multisig in scope.

Recommendation:
We recommend clearly documenting this behaviour so that smart contract users and proposal voters know about and thus can cast duly and informed votes.

Comment from Teia community developers:
We agree that the effect of the change of the multisign parameters on previous proposals is not optimal. It would be better if those changes would only affect newly created proposals. However, keeping the parameters history would complicate a bit more the smart contract code and logic. For example, one could add the expiration_time and minimum_votes parameters to each proposal big map entry, but that would increase the storage costs and might confuse users, because different proposals will have different parameters. Removing the votes from removed users will imply a loop over the votes big map with the implied gas costs, etc. We expect that those parameters will not be changed very often and that the changes will not be too radical (like going for 5 days expiration time to 1year), so the propagation to older proposals will be minimized.
We agree that the multisign documentation should be improved and all peculiar cases should be explained in detail with clear examples.

Results from follow-up assessment:
No follow-up assessment performed. Status remains on “open”.

O-TEI-004: Documentation

The documentation lacks detailed information with regards to:

- smart contract design, specification, internal mechanics, and use cases
- potential risks and pitfalls in the (wrong) usage of the smart contract
- parameters used in storage and as parameters in entrypoints

This observation affects all smart contracts in scope.

Recommendation:
We recommend improving the documentation. The documentation should be directed to smart contract users and should provide sufficient information about how the smart contracts work, on how to use them, and also the risks associated with them, including suggestions of Do's and Don'ts to mitigate those risks.

Comment from Teia community developers:
We agree that some critical documentation is still lacking in both contracts.

We have already started to document the multisign contract in a dedicated web interface, temporally hosted at https://multisign.onrender.com

The H=N web site already contains detailed information on how to interact with the v2 marketplace contract and we will update it as soon as the new Teia marketplace is released.
The documentation in the smartpy code will also be improved with a clear description of the contract storage and entry point parameters.

Results from follow-up assessment:
The documentation of the code repository has been improved. In addition, documentation[1] has been written describing the function of the smart contracts. Inference recommends to also include potential risks and pitfalls in using the smart contracts. Status set from "open" to "partially closed".

[1] https://leonnicholls.medium.com/hic-et-nunc-multi-sig-smart-contract-d1f63fe5d24 and https://leonnicholls.medium.com/hic-et-nunc-v3-marketplace-smart-contract-ca1882b01b66

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 analyzed 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 analyzed.

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 Teia community developers and checked during our security assessment.

marketplace smart contract

Scenario	Impact rating & descriptive	Assessment result
“Collector” extracts more objects than only the bought one.	High: All scenarios allow extraction of tez or objects, which are not for sale or have not been paid for. Thus, there is a financial loss for the “Collector”or “Swapper”. For Teia this would mean a loss of reputation.	**Ok**
“Collector” buys object “A”, but is able to extract item “B” (more valuable than “A”) from the “Swapper”.		
“Swapper” receives the amount for the sold object, but does not have to transfer the specified object to be sold to the “Collector”		
“Swapper” receives the amount for the sold object “A”, but transfers (the less valuable) object “B” to the “Collector”.		
Anyone can steal an object without providing the (full) selling price.		
Anyone can steal the selling amount for an object.		
“Swapper” and/or “Collector” can prevent royalties from being paid.	Medium: Financial impact for the artists. This could lead to claims to Teia from artists, but also resulting in a loss of reputation.	**Not ok** See S-TEI-001

“Swapper” and/or “Collector” can prevent management fees from being paid.	Low: (Small) financial impact for Teia.	**Not ok** See S-TEI-001

multisig smart contract

Scenario	Impact rating & descriptive	Assessment result
Proposals can be executed without reaching a vote quorum.	High: Proposal gets executed without a quorum. Loss of trust from “users” in the solution. Loss of reputation in the market.	**Ok**
Proposal includes a second piggybacked (hidden) proposal. If the first proposal gets approved, the second piggybacked (hidden) proposal also gets executed.	High: Loss of financial assets (tez / tokens). Loss of trust/reputation.	**Ok**
A single voter can prevent the execution of a proposal.	High: Loss of trust/reputation. Potential financial impact in case a liability can not be resolved or not possible to take advantage of good market conditions.	**Ok**
Anyone can prevent the execution of a proposal.		

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 / materialization 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