tzBTCv2 for Papers

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 24.04.2024

inference

Table of contents

Version / Date	Description
1.0 / 24.04.2024	Final version.

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of tzBTCv2.

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 22nd of March 2024 and the 24th of April 2024. Feedback from Papers was received and Inference performed a follow-up assessment.

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 tzBTCv2.

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-TZB-001: Service fee increase	- / open
O-TZB-002: Multiple gatekeepers	- / open
O-TZB-003: Gas exhaustion if too many signers	- / open
O-TZB-004: BTC split in many small UTXOs	- / open

Project overview

Scope

Smart contract security assessment

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

- tzBTC ledger smart contract:
  - SmartPy code:
    contracts/tzbtc_ledger.py, including all imported files and code required to compile the smart contract
  - Compiled Michelson code:
    __SNAPSHOTS__/compilation/all/tzBTCLedger/step_000_cont_0_contract.tz

The files in scope were made available via a source code repo:
https://gitlab.papers.tech/papers/customer-tezos-foundation/tzbtc-v2/smart-contracts/
and our initial security assessment considered commit
“7b17a5f803cc81f9cae15ec000b2bc8291e1274b”.

Our reassessment considered commit:
“10030ca17dbe5c4e1455990d1ef8b5a3f3513db8”

We also reviewed the Michelson code of the deployed tzBTC ledger smart contract on the Tezos mainnet: KT1DuiB3C9dcLKmRewiVWEVVE5nbXngo2Ci5

Acurast scripts security assessment

Also in scope for the security assessment were the following Acurast scripts:

- Signer scripts:
  - /src/methods/signer/confirmChangeUTXO.ts
  - /src/methods/signer/confirmUTXO.ts
  - /src/methods/signer/signBurn.ts
- Gatekeeper scripts:
  - /src/methods/gatekeeper/gatekeeperDepositForwardTransaction.ts
  - /src/methods/gatekeeper/gatekeeperWithdrawForwardTransacation.ts

The files in scope were made available via a source code repo:
https://gitlab.papers.tech/papers/customer-tezos-foundation/tzbtc-v2/tzbtc-acurast-jobs/
and our initial security assessment considered commit
“a6a60d9f8a4b6073b5ee245919d2e7997017caa5”.

Our reassessment considered commit:
“8940a830e32de55cce1bcc90a3f9c6510b01bc66”

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
- Deployment and initial configuration of the deployed smart contract was out of scope.
- Smart contracts for tokens not in scope were considered trusted.
- The key management of associated secret keys has not been assessed.

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 Papers 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
- Source code review of the smart contract in Michelson
- Security assessment of the Acurast scripts
- Static analysis of the Acurast scripts

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 SmartPy
- Source code review of changes in the smart contracts in Michelson
- Source code review of changes in the Acurast scripts
- 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-TZB-001: Service fee increase

If the administrator of the smart contract increases the service fee while a mint or burn operation is in progress, the execution of the "mint" or "confirm_burn" entrypoints may fail, if the amount to be minted / burned no longer covers the increased service fee.

We have classified this as an observation only and not as a security issue, as changing the service fee can only be performed by administrators of the contract.

Recommendation:
We recommend storing the service fee along with the other mint or burn request data when a mint or burn request is initiated and utilising the stored service fee. This approach resolves our reported issue and also prevents the requester from facing suddenly increased service fees after submitting their burn or mint request.

Comment from Papers:
We will document it so that we are aware that we shouldn’t update the service fee if we have opened UTXOs.

O-TZB-002: Multiple gatekeepers

The smart contract permits the registration of multiple gatekeepers. Additionally, it maintains a single whitelist of verified addresses. Each registered gatekeeper has the ability to add or remove addresses from this whitelist.

However, this setup may pose issues when gatekeepers represent different institutions with varying standards, such as differing approaches to conducting KYC (Know Your Customer). Consequently, there is a risk that an address validated and whitelisted by gatekeeper A may not meet the verification and whitelisting criteria of gatekeeper B.

We have categorised this as an observation only and not a security concern, based on Paper's information that the contract will exclusively be utilised by one institution, and considering that gatekeepers can only be added by the administrator of the smart contract.

Recommendation:
We recommend implementing separate whitelists for each gatekeeper if the same smart contract needs to be utilised by multiple institutions.

Comment from Papers:
The gatekeepers registered will all be representing the same institution. Of the 3 registered gatekeepers only 1 gatekeeper, the backend, will whitelist addresses which successfully passed KYC and were able to sign a message. As there are no plans to change the gatekeeper setup, we do not see any necessity to separate whitelists per gatekeeper.

O-TZB-003: Gas exhaustion if too many signers

If there are too many registered signers, there's a risk of the "candidate_utxo_map" growing excessively large as each UTXO candidate's data structures (MAP and SET) within the big map value expand. Consequently, the smart contract might encounter gas exhaustion, making transactions impossible if loading a candidate's value from the "candidate_utxo_map" fails to deserialize successfully.

We have classified this as an observation only and not as a security issue, as registering signers can only be performed by administrators of the contract.

Recommendation:
We recommend limiting the total number of registered signers to prevent the risk of gas exhaustion.

Comment from Papers:
The number of trusted signers is and always will be limited. Currently to 10 trusted signers. While this number could theoretically be increased for redundancy, it will be impractical to ever have more than a very low 2 digit number of signers (eg 15, or 20).

O-TZB-004: BTC split in many small UTXOs

In the event that the BTC held in custody are fragmented into numerous small UTXOs, the conversion of tzBTC to BTC might fail due to the "confirm_burn" entrypoint, which constrains the maximum of aggregated UTXOs to a specific limit set by the parameter "max_utxo_per_tx_count."

Nevertheless, a limitation, set by the parameter "max_utxo_per_tx_count," is crucial to the present smart contract design. It serves to prevent gas exhaustion in scenarios where an excessive number of UTXOs have to be aggregated for a single burn operation.

If there are an excessive number of UTXOs to process, failure could also occur due to the limiting parameter "max_btc_network_fee". However, this parameter is essential to ensure that gatekeepers cannot spend BTC gas fees limitlessly.

We've categorised this as an observation only. This stems from our understanding, following feedback from the Papers' team, that administrators of the BTC custody multisignature wallet possess the capability, through their keys, to consolidate various small UTXOs in such an event into fewer ones. Subsequently, the administrators of the tzBTC ledger contract are capable of updating the registered UTXOs in the contract's storage.

Recommendation:
We suggest monitoring UTXO fragmentation and potentially adjusting the backend server algorithm if UTXO fragmentation becomes excessive. The backend server algorithm may be optimised to prevent or even reduce UTXO fragmentation.

Comment from Papers:
As pointed out by Inference, the UTXOs will be monitored and can always be:
1) consolidated manually via the key holder setup as a medium of last resort. Depending of the real world usage,
2) the backend server algorithm picking the UTXOs can be optimised to keep UTXO fragmentation within an ideal range.

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
As a normal user, add myself as an admin.	**Ok** Nothing identified.
Gatekeeper can call mint entrypoint multiple times and/or without an underlying confirmed UTXO.	**Ok** Nothing identified.
Gatekeeper spends all BTC in custody.	**Ok** In order to burn the corresponding tzBTC provided by the user, these tzBTC have to be in custody. The gatekeeper can influence the bitcoin network fee, but this amount is limited in the smart contract. Additional checks are in the script executed by the Custodian Acurast Processor.
A not whitelisted user can propose a burn.	**Ok** Not possible.
A user can propose a burn less than the required min amount.	**Ok** Not possible.
A user receives more than the amount intended to be burned (minus fees).	**Ok** Nothing identified.
A user can prevent the fees or part of them in a burn.	**Ok** Nothing identified.
A user can receive the BTC and cancel the burn to receive the tzBTC.	**Ok** Nothing identified.
A single signer blocks other signers from confirming UTXOs.	**Ok** Nothing identified.
A single hacked / rogue signer.	**Ok** Nothing identified as long as there is no “X out of X” scheme in place.

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
Acurast	Decentralised serverless cloud solution
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