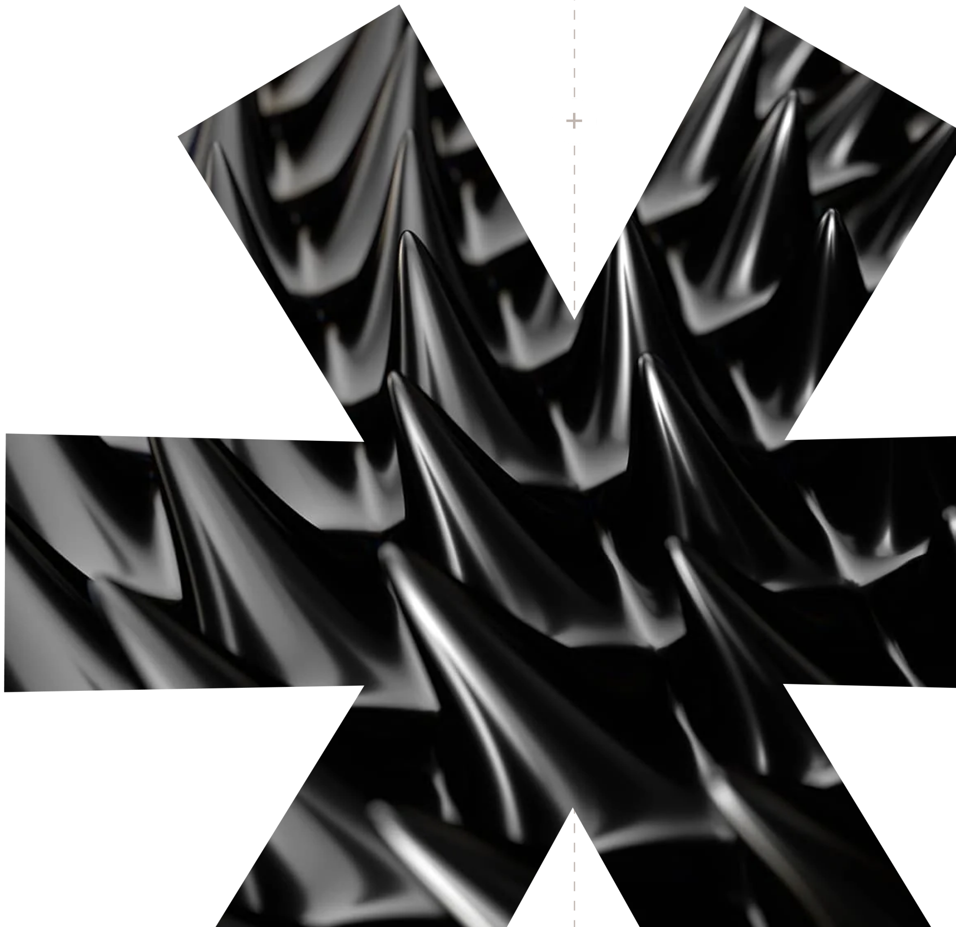
thesisdefense

Security Audit Report

Mezo

Portal Smart Contracts

Initial Report // March 08, 2024
Final Report // March 14, 2024



Team Members

Ahmad Jawid Jamiulahmadi // Security Auditor
Mukesh Jaiswal // Security Auditor
Bashir Abu-Amr // Head of Delivery

Table of Contents

About Thesis Defense

Thesis Defense serves as the auditing services arm within Thesis, Inc., the venture studio behind tBTC, Fold, Taho, Etcher, and Mezo. Our team of security auditors have carried out hundreds of security audits for decentralized systems across a number of technologies including smart contracts, wallets and browser extensions, bridges, node implementations, cryptographic protocols, and dApps. We offer our services within a variety of ecosystems including Bitcoin, Ethereum + EVMs, Stacks, Cosmos / Cosmos SDK, NEAR and more.

Thesis Defense will employ the Thesis Defense Audit Approach and Audit Process to the in scope service. In the event that certain processes and methodologies are not applicable to the in scope services, we will indicate as such in individual audit or design review SOWs. In addition, Thesis Defense provides clear guidance on successful Security Audit Preparation.

Section_1.0

Scope

Technical Scope

Mezo Portal

- **Repository:** https://github.com/thesis/mezo-portal/tree/main/solidity
- **Audit Commit:** 39a312c36a9e1abd5a7f3d982fad2b5ed452c8eb
- **Verification Commit:** 0000ff5c322edeb69b18072c7cd2455b8afc8bf2
- **Files in Scope:**
    - Portal.sol
    - BitcoinDepositor.sol

tBTC v2

- **Repository:** https://github.com/keep-network/tbtc-v2/tree/main
- **Audit Commit:** 9e047d11703415e1a1844a64b4985a181570fcdd
- **File in Scope:**
    - AbstractTBTCDepositor.sol

Documentation

- Mezo Security Audit Document
- Mezo Release 1 Specification Document
- tBTC Repository Documentation

Section_2.0

Executive Summary

Schedule

This security audit was conducted from March 4, 2024 to March 8, 2024 by 2 senior security auditors for a total of 2 person-weeks.

Overview

Thesis Defense conducted a manual code review of Mezo's Portal Smart Contracts implementation.

The Mezo Portal smart contracts are intended to allow BTC and tBTC holders to deposit and lock tokens in order to earn points. These points will be used to determine airdrops from the Mezo Blockchain mainnet. The smart contracts are intended to be integrated with a frontend interface and the tBTC bridge, which were considered out of scope for this audit.

Threat Model

For this review, our team considered a threat model whereby the smart contracts assume all external components as untrusted. For those components that are out of scope for this review, we considered the components to be untrusted, but functioning as intended. The Mezo Portal smart contracts are designed to be integrated with the tBTC v2 bridge. We assumed to tBTC the bridge behaves as expected. The smart contracts are designed to be integrated with a user interface which we considered untrusted. Furthermore, we assume that the governance of the Portal smart contracts is sufficiently decentralized and not malicious.

Security by Design

We identified issues in some design elements of the smart contracts. We found that the `BitcoinDepositor` smart contract does not check that the user selected locking period specified in the extra data section of the Bitcoin script is in range, which could make the user unable to withdraw their tokens from the protocol (Issue 1). We recommend a solution in the smart contract.

We also found that the smart contracts do not implement a 2-step process for transferring the smart contract ownership address (Issue 3).

In looking at the out-of-scope tBTC bridge implementation, to have a more precise understanding of the smart contracts in scope, we discovered an unlikely but possible condition in the tBTC bridge where users would be unable to make deposits. We recommend that the user interface alert users of the current status of the bridge (Issue 2).

In our audit, our team also noted a lock period entered as non-integer weeks (4.5 or 5.5 weeks), the deposit interval is rounded down. Although this is intended behavior, we recommend clarifying this to the users.

Secure Implementation

We conducted an in-depth examination and manual review of the files in scope and found them implemented in adherence to best practices.

Use of Dependencies

The project uses an OpenZeppelin library whose version is not the most recent. We recommend using the most recent version that includes up to date security fixes (Issue 7).

Tests

There is sufficient testing implemented, which covers most of the functionality of the `Portal` and `BitcoinDepositor` smart contracts.

Project Documentation

The files are well commented and adhering to NatSpec, but there are some instances of code comments that should be updated to reflect the implementation more accurately (Issue 4).

Section_3.0

Key Findings Table

Issues	Severity	Status
ISSUE #1 `BitcoinDepositor` Might Fail to Finalize Some Deposits	Low	Not Fixed
ISSUE #2 Optimistic Pause of Bridge (Out-of-Scope)	Low	Not Fixed
ISSUE #3 Lack of a Two-Step Process for Ownership Change	Low	Not Fixed
ISSUE #4 Update Code Comments to Reflect the Implementation	None	Fixed
ISSUE #5 Check the Sanity of Lock Interval Parameters When Setting `minLockPeriod` and `maxLockPeriod`	None	Fixed
ISSUE #6 Prevent Adding a Supported Token With `None` Ability	None	Fixed
ISSUE #7 Use Latest Open Zeppelin Library Implementation	None	Fixed
ISSUE #8 Pin and Lock Pragma	None	Fixed
ISSUE #9 Implement 0 Address Check	None	Fixed
ISSUE #10 Prevent Resetting the `depositInfo.unlockAt`	None	Fixed
ISSUE #11 Check for Equality When Setting `minLockPeriod` and `maxLockPeriod`	None	Fixed

Severity definitions can be found in Appendix A

Section_4.0

Findings

We describe the security issues identified during the security audit, along with their potential impact. We also note areas for improvement and optimizations in accordance with best practices. This includes recommendations to mitigate or remediate the issues we identify, in addition to their status before and after the fix verification.

ISSUE#1

BitcoinDepositor Might Fail to Finalize Some Deposits

✓ Low   ☒ Not Fixed

Location

BitcoinDepositor.sol#L258-L263

Description

The `finalizeDeposit` function in the `BitcoinDepositor` smart contract finalizes tBTC deposits revealed to the tBTC bridge, and deposits them on behalf of the actual depositor to the Portal smart contract. The Bitcoin transaction token, which is a P2(W)SH Bitcoin script, includes the deposit owner address and deposit lock time in seconds in the `depositor-extra-data` section of the token.

Deposits made to the `Portal` smart contract can be locked for a user specified period which must be into a specific range, i.e. more than `minLockPeriod` and less than `maxLockPeriod`. Deposits with a locking period less than the current minimum locking period and more than the current maximum locking period will be disallowed and the transaction will be reverted.

Therefore, if the locking period specified in the `depositor-extra-data` section in the P2(W)SH Bitcoin script is not in the specific range, the `BitcoinDepositor` smart contract will not be able to finalize the deposit in the `finalizeDeposit` function since the transaction is reverted due to a call to the `depositFor` function in the `Portal` smart contract which prevents deposits with a `lockingPeriod` less than the current minimum locking period or more than the current maximum locking period if the `lockingPeriod` is not zero.

Impact

Since the `finalizeDeposit` function in the `BitcoinDepositor` smart contract is always reverted due to the incorrect locking period, the user's deposit will be stuck in the `BitcoinDepositor` smart contract.

Recommendation

Resolving this issue requires careful consideration of potential security risks and complexity that can be introduced with a specific solution. However, we can recommend the following solutions: Before sending the `depositFor` transaction to the `Portal` smart contract, in the `finalizeDeposit` function in the `BitcoinDepositor` smart contract, check if the `lockingPeriod` provided is within the range of the minimum locking period and the maximum locking period, and if not:

- Transfer the minted tBTC tokens directly to the deposit owner, or
- Deposit tokens to the `Portal` smart contract without locking them (deposit with zero locking period), or
- Deposit tokens to the `Portal` smart contract with the current maximum or minimum locking period.

Verification Status

This issue can be mitigated in the Mezo interface dApp, which was out of scope for this security audit.

ISSUE#2

Optimistic Pause of Bridge (Out-of-Scope)

✓ Low   ☒ Not Fixed

Location

BitcoinDepositor.sol#L183

Description

The tBTC bridge handles user deposits through optimistic minting and sweeping. Optimistic minting allows the minting of tBTC prior to the `TBTCVault` receiving the `Bank` smart contract balance. Two permissioned sets, Minters and Guardians, operate in a 1-of-n mode. Minters monitor revealed deposits and can request the minting of tBTC, with any single Minter capable of initiating this action. A delay, known as `optimisticMintingDelay`, occurs between the Minter's request and the actual minting of tBTC. Within this delay period, any Guardian has the authority to cancel the minting process.

In Sweeping the bridge active wallet periodically signs a transaction that unlocks all of the valid, revealed deposits above the dust threshold, combines them into a single UTXO, and the balances of depositors in the `Bank` smart contract are increased when the Simple Payment Verification sweep proof is submitted to the bridge.

As a result, there are conditions when tBTC minting is not allowed:

- When the optimistic minting is paused, Minters will not be able to put in a request for an optimistic minting of tBTC
- When the wallet is in the state `MovingFunds`, the wallet is expected to move outstanding funds to another wallet. The wallet can still fulfill pending redemption requests, although new redemption requests and new deposit reveals are not accepted.

As a result, when optimistic minting is paused and the wallet is in the state `MovingFunds`, the user will not be able to reveal their deposited BTC to the `BitcoinDepositor` smart contract.

Impact

While the likelihood of such a situation occurring is minimal, it is still possible. If it does happen, users will have the option to claim their deposited BTC after the expiration of the lock time interval. This effectively renders the user's assets inaccessible for the duration of the lock period, and the user is unable to earn points.

Recommendation

We recommend that users are alerted in the interface about the paused status of the bridge and wallet status before allowing any deposit transactions.

Verification Status

The Mezo team stated that the Mezo interface dApp will restrict deposits when the tBTC bridge is in optimistic pause mode.

ISSUE#3

Lack of a Two-Step Process for Ownership Change

Low Fixed

Location

Portal.sol#L9

Description

The `Portal` smart contract uses OpenZeppelin's `OwnableUpgradeable` library for the transfer of ownership of a smart contract from one owner address to another. This library does not implement a two-step ownership transfer.

Impact

A two-step process for ownership transfer significantly reduces the probability of incorrectly transferring ownership of a smart contract which would result in the permanent loss of control of the smart contract.

Recommendation

We recommend implementing a Two-Step process for ownership change. We recommend using OpenZeppelin's Ownable2StepUpgradeable library for this purpose.

ISSUE#4

Update Code Comments to Reflect the Implementation

None Fixed

Location

Portal.sol#L305

Portal.sol#L315

Description

There are comprehensive code comments in the smart contracts that adhere to NatSpec guidelines. However, our team found instances of an inaccurate code comment.

Impact

None – no security impact.

Recommendation

We recommend updating the referenced code comments.

ISSUE#5

Check the Sanity of Lock Interval Parameters When Setting `minLockPeriod` and `maxLockPeriod`

None Fixed

Location

Portal.sol#L171-L182

Portal.sol#L189-L200

Description

The `setMinLockPeriod` and `setMaxLockPeriod` functions in the Portal smart contract do not check if the newly set locking period is normalized to weeks. Additionally, the `setMinLockPeriod` function currently lacks appropriate input validation for a minimum lock period of one week. The absence of a check allows the lock duration interval to be set to values that are not normalized.

Without input validation, the minimum lock period can be set to any value that is less than 1 week, which is not consistent with the intended functionality and could lead to unexpected outcomes.

Impact

None – no security impact.

Recommendation

We recommend that the locking periods supplied in the aforementioned functions are checked for normalization. We also recommend implementing a sanity check on the minimum lock duration parameter to enforce the minimum one-week lock interval.

ISSUE#6

Prevent Adding a Supported Token With `None` Ability

None Fixed

Location

Portal.sol#L149-L166

Description

The `addSupportedToken` function in the Portal smart contract is used to add a new token to the list of supported tokens. A new supported token should be added with a token ability of `Deposit` or `DepositAndLock`. However, this function allows adding a supported token with the `None` token ability.

Impact

None – no security impact.

Recommendation

We recommend adding to a check to prevent adding a supported token with the None token ability.

ISSUE#7

Use Latest Open Zeppelin Library Implementation

None Fixed

Location

package.json#L48

Description

The current version of the smart contracts relies on a prior release of the OpenZeppelin library, wherein an issue with Base64 encoding exists. While this problem does not currently impact the functionality of the `Portal` smart contracts, upgrading the library helps the contract mitigates potential security issues related to the known issue as the project evolves over time.

Impact

None – no security impact.

Recommendation

We recommend that an upgrade be made to the Open Zeppelin library version to 5.0.2.

ISSUE#8

Pin and Lock Pragma

None Fixed

Location

https://github.com/thesis/mezo-portal/contracts

Description

When using the pragma directive in Solidity, it is essential to specify the exact version of the Solidity compiler that your smart contract is compatible with. This practice, known as locking the pragma, ensures that your contract is compiled and executed as intended, avoiding potential issues caused by compiler version differences.

Impact

None – no security impact.

Recommendation

We recommend specifying the most recent, exact version of the Solidity compiler.

ISSUE#9

Implement 0 Address Check

None Fixed

Location

https://github.com/thesis/mezo-portal/blob/39a312c36a9e1abd5a7f3d982fad2b5ed452c8eb/solidity/contracts/Portal.sol#L380

Description

In the location referenced above, a zero address check is missing validating the correctness `depositOwner` address, thereby preventing an incorrectly set value. Zero address checks are essential when an address is the receiver of a token or value.

Impact

None – no security impact.

Recommendation

We recommend adding a zero address check for `depositOwner` in the `_depositFor` function.

ISSUE#10

Prevent Resetting the `depositInfo.unlockAt`

None Fixed

Location

Portal.sol#L361-L369

Description

The referenced `if` condition in the `lock` function inside the `Portal` smart contract does not revert if the newly provided unlocking time is the current unlocking time hence resetting it with the same value.

Impact

None – no security impact.

Recommendation

We recommend preventing the resetting of the `depositInfo.unlockAt` by adding an equality check in the revert condition referenced above.

ISSUE#11

Check for Equality When Setting `minLockPeriod` and `maxLockPeriod`

None Fixed

Location

Portal.sol#L171-L178

Portal.sol#L190-L196

Description

When setting `minLockPeriod` and `maxLockPeriod` in the `setMinLockPeriod` and `setMaxLockPeriod` functions respectively in the `Portal` smart contract, `minLockPeriod` can be set to `maxLockPeriod` and vice versa since the referenced revert conditions in both functions don't check for equality of `minLockPeriod` and `maxLockPeriod` to revert consequently. This might result in unexpected behavior.

Impact

None – no security impact.

Recommendation

We recommend that a check for equality in the referenced functions be also implemented to prevent unexpected behavior.

Section_5.0

Appendix A

Severity Rating Definitions

At Thesis Defense, we utilize the Immunefi Vulnerability Severity Classification System - v2.3.

Severity	Definition
Critical	• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results • Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield • Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties • Permanent freezing of funds • Permanent freezing of NFTs • Unauthorized minting of NFTs • Predictable or manipulable RNG that results in abuse of the principal or NFT • Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content) • Protocol insolvency
High	• Theft of unclaimed yield • Theft of unclaimed royalties • Permanent freezing of unclaimed yield • Permanent freezing of unclaimed royalties • Temporary freezing of funds • Temporary freezing NFTs
Medium	• Smart contract unable to operate due to lack of token funds • Enabling/disabling notifications • Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol) • Theft of gas • Unbounded gas consumption
Low	• Contract fails to deliver promised returns, but doesn't lose value
None	• We make note of issues of no severity that reflect best practice recommendations or opportunities for optimization, including, but not limited to, gas optimization, the divergence from standard coding practices, code readability issues, the incorrect use of dependencies, insufficient test coverage, or the absence of documentation or code comments.

Section_6.0

Appendix B

Thesis Defense Disclaimer

Thesis Defense conducts its security audits and other services provided based on agreed-upon and specific scopes of work (SOWs) with our Customers. The analysis provided in our reports is based solely on the information available and the state of the systems at the time of review. While Thesis Defense strives to provide thorough and accurate analysis, our reports do not constitute a guarantee of the project's security and should not be interpreted as assurances of error-free or risk-free project operations. It is imperative to acknowledge that all technological evaluations are inherently subject to risks and uncertainties due to the emergent nature of cryptographic technologies.

Our reports are not intended to be utilized as financial, investment, legal, tax, or regulatory advice, nor should they be perceived as an endorsement of any particular technology or project. No third party should rely on these reports for the purpose of making investment decisions or consider them as a guarantee of project security.

Links to external websites and references to third-party information within our reports are provided solely for the user's convenience. Thesis Defense does not control, endorse, or assume responsibility for the content or privacy practices of any linked external sites. Users should exercise caution and independently verify any information obtained from third-party sources.

The contents of our reports, including methodologies, data analysis, and conclusions, are the proprietary intellectual property of Thesis Defense and are provided exclusively for the specified use of our Customers. Unauthorized disclosure, reproduction, or distribution of this material is strictly prohibited unless explicitly authorized by Thesis Defense. Thesis Defense does not assume any obligation to update the information contained within our reports post-publication, nor do we owe a duty to any third party by virtue of making these analyses available.