

# Security Audit Report

---

## Threshold

### tBTC Base Smart Contracts

Initial Report // March 29, 2024

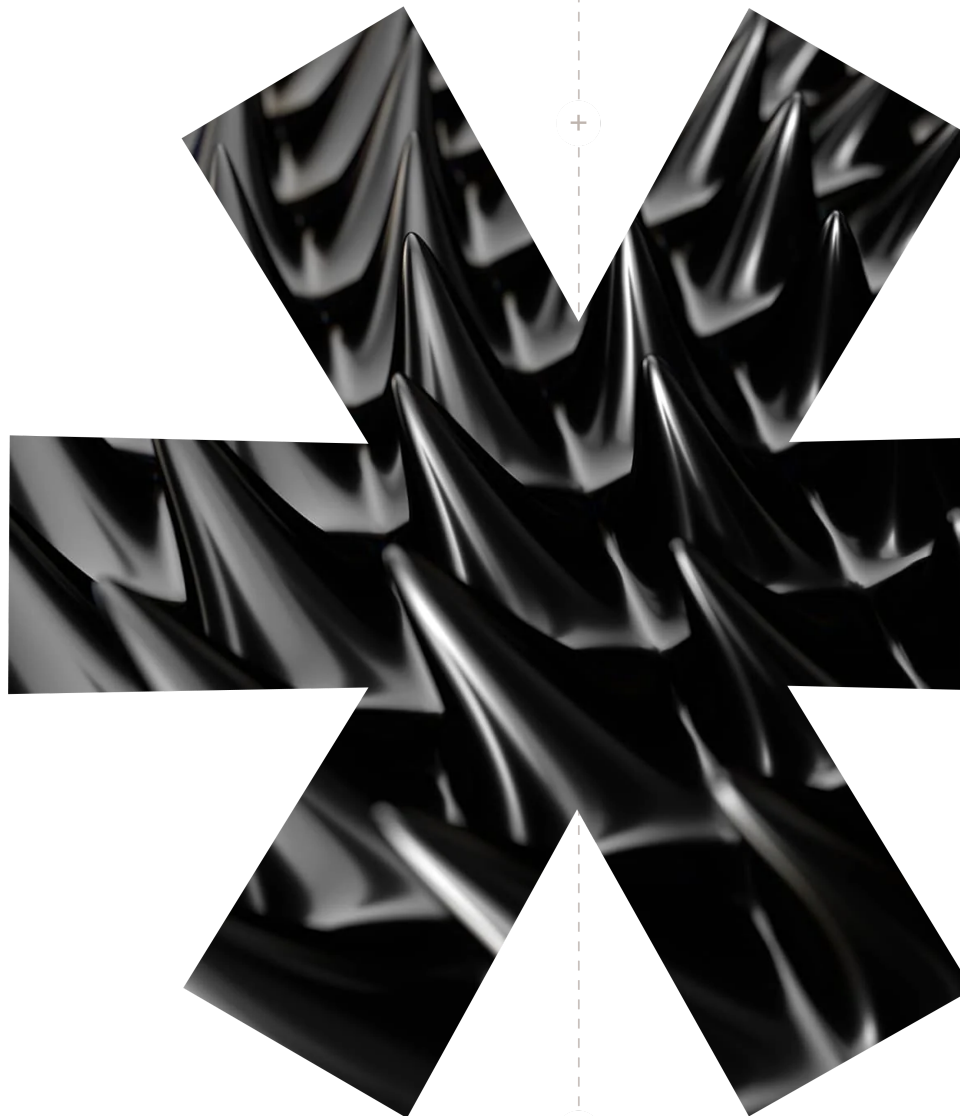
Final Report // April 11, 2024

#### Team Members

Ahmad Jawid Jamiulahmadi // Security Auditor

Mukesh Jaiswal // Security Auditor

Bashir Abu-Amr // Head of Delivery



# Table of Contents

<u>1.0 Scope</u>	3
↳ <u>1.1 Technical Scope</u>	
↳ <u>1.2 Documentation</u>	
<u>2.0 Executive Summary</u>	4
↳ <u>2.1 Schedule</u>	
↳ <u>2.2 Overview</u>	
↳ <u>2.3 Threat Model</u>	
↳ <u>2.4 Security by Design</u>	
↳ <u>2.5 Secure Implementation</u>	
↳ <u>2.6 Application of Best Practices</u>	
<u>3.0 Key Findings Table</u>	6
<u>4.0 Findings</u>	7
↳ <u>4.1 The Out of Scope Relay Bot Is Security Critical</u>	
Medium Not Fixed	
↳ <u>4.2 An Attacker Can Grief the Relay Bot by Frontrunning the initializeDeposit Function</u>	
Medium Partial	
↳ <u>4.3 Relay Bot is Susceptible to Spamming</u>	
Medium Not Fixed	
↳ <u>4.4 No Two-Step Process for Ownership Change</u>	
Low Not Fixed	
↳ <u>4.5 Missing Zero Address Checks</u>	
Low Fixed	
↳ <u>4.6 Use Custom Errors to Save Gas</u>	
None Not Fixed	
↳ <u>4.7 Unlocked Pragma Version</u>	
None Fixed	
↳ <u>4.8 Use Latest Open Zeppelin Library Implementation</u>	
None Not Fixed	
<u>5.0 Appendix A</u>	13
↳ <u>5.1 Severity Rating Definitions</u>	
<u>6.0 Appendix B</u>	14
↳ <u>6.1 Thesis Defense Disclaimer</u>	



# About Thesis Defense

---

Thesis Defense serves as the auditing services arm within Thesis, Inc., the venture studio behind tBTC, Fold, Tahoe, 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

- **Repository:** <https://github.com/keep-network/tbtc-v2/tree/89616d552b9c36036e0725740443d73dfad5b682/solidity/contracts/l2>
- **Audit Commit:** 89616d552b9c36036e0725740443d73dfad5b682
- **Verification Commit:** 1eaab786b924f533c9b2c590d7490d3557f131ab
- Any third-party or dependency library code is considered out of scope, unless explicitly specified as in scope above.

## Documentation

- Technical documentation



# Executive Summary

---

## Schedule

This security audit was conducted from March 18, 2024 to March 29, 2024 by 2 security auditors for a total of 4 person-weeks.

## Overview

Thesis Defense conducted a manual code review of the tBTC Base smart contract implementation. The tBTC Base smart contracts are intended to enable users to bridge BTC to the [Base](#) blockchain, an Ethereum layer-2 (L2) scaling solution. Users make a deposit to a BTC address and an equivalent amount of tBTC is minted on their behalf by the [tBTC Bridge](#) on Ethereum layer-1 (L1). The L1 tBTC is then bridged to the Base L2 using the [Wormhole Portal](#) where the user receives L2 tBTC.

## Threat Model

To begin the security audit, we created a threat and trust model for the implementation. This helps us to define system inputs and outputs and to better define and clarify the scope and areas of concern for the security audit. We considered potential vulnerabilities resulting from malicious depositors, relayers, and miners. We assumed the governance of the protocol to be sufficiently decentralized and not malicious. However, we found that the Relayer Bot plays a security-critical role in the tBTC Base system and should be included in the scope of future audits ([Issue 1](#)). Apart from the issues raised in this report, we assumed the Relayer Bot to function as intended (See [Use of Dependencies](#)).

## Security by Design

We investigated the design of tBTC Base smart contracts to assess the adherence to decentralized system design best practices, and the absence of common design issues that could result in security vulnerabilities. We found that the smart contracts were designed with security in mind including, but not limited to, utilizing appropriate access control, clear code comments regarding prerequisites and expectations for each function call, adherence to Wormhole integration guidelines in the `L1BitcoinDepositor` and `L2BitcoinDepositor` smart contracts to limit attack surfaces, and adherence to tBTC Bridge integration guidelines for deposit initialization and finalization.

However, we found that very low transaction costs on the Base L2 blockchain make the system susceptible to spamming. Although there are potential ways to mitigate this issue, we were unable to identify a remediation that resolves the issue completely ([Issue 3](#)). Furthermore, we found that the `initializeDeposit` function in the `L2BitcoinDepositor` smart contract could be susceptible to race conditions which would result in wasted gas ([Issue 2](#)).

## Secure Implementation

The tBTC Base smart contracts are implemented in accordance with best practices as demonstrated by modularization of functionality, adherence to upgradeability guidelines, and the use of code comments to improve the readability of the code. We did not identify any issues in the implementation. However, we identified two areas for improving the implementation, which include the implementation of zero address checks ([Issue 5](#)) and opportunities for gas optimization by using custom errors ([Issue 6](#)).



# Application of Best Practices

## Project Documentation and Tests

The project documentation available during the security audit was accurate and helpful. The documentation included architectural diagrams, which facilitated understanding and reasoning about the intended functionality of the implementation. While the documentation was limited, it did not inhibit us from comprehensively reviewing the implementation. However, we recommend continuous improvement of project documentation when and where possible.

The code comments in the implementation are comprehensive, providing a thorough explanation and understanding of the code's functionality and intended purpose. In addition, the codebase implements comprehensive test cases, in accordance with best practices.

## Use of Dependencies

The smart contractors make use of a Relayer Bot to perform security-critical functionality. The smart contracts also rely on the tBTC Bridge to function as intended. The Wormhole facilitates cross-chain operations, specifically the transfer of minted L1 chain (Ethereum) tBTC to the L2 chain (Base). Within the Wormhole, relayers are tasked with transmitting Verifiable Action Approvals (VAAs) to the designated destination chain, with the Wormhole `TokenBridge` smart contract maintaining a record of token transfers.

The tBTC Base smart contracts import OpenZeppelin's OwnableUpgradeable library. We recommend using a 2-step process to mitigate against ownership transfer errors that could result in loss of control of the smart contracts ([Issue 4](#)). We also recommend importing the latest libraries, which include the latest security patches ([Issue 8](#)).



# Key Findings Table

Issues	Severity	Status
ISSUE #1 The Out of Scope Relay Bot Is Security Critical	Medium	Not Fixed
ISSUE #2 An Attacker Can Grief the Relay Bot by Frontrunning the <code>initializeDeposit</code> Function	Medium	Partial
ISSUE #3 Relay Bot is Susceptible to Spamming	Medium	Not Fixed
ISSUE #4 No Two-Step Process for Ownership Change	Low	Not Fixed
ISSUE #5 Missing Zero Address Checks	None	Fixed
ISSUE #6 Use Custom Errors to Save Gas	None	Not Fixed
ISSUE #7 Unlocked Pragma Version	None	Fixed
ISSUE #8 Use Latest Open Zeppelin Library Implementation	None	Not Fixed

Severity definitions can be found in [Appendix A](#)



# 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

### The Out of Scope Relay Bot Is Security Critical

Medium

Not Fixed

#### Location

[contracts/l2][<https://github.com/keep-network/tbtc-v2/blob/d11195be85b03b6c57fca935217aa635abd492c1/solidity/contracts/l2>]

#### Description

The Relay Bot plays a central and critical role in facilitating the functionality of the tBTC Base smart contracts implementation. Specifically, the relay bot makes sure that messages are relayed from the `L2BitcoinDepositor` smart contract on the Base blockchain to the `L1BitcoinDepositor` smart contract on Ethereum. The Relay Bot also reads from the Bitcoin blockchain, checking that the deposit being processed has at least 1 confirmation. The Relay calls the `initializeDeposit` and `finalizeDeposit` functions in the `L1BitcoinDepositor` smart contract on Ethereum, on behalf of the depositing user. The `initializeDeposit` function is used to reveal the user's BTC deposit on the tBTC Bridge on the L1 chain. The `finalizedDeposit` function is used to finalize the deposit on the L1 chain after the tBTC mint event has been emitted by tBTC Bridge smart contracts.

Given that the security-critical Relay Bot was out of the scope of this security audit, we had to make many assumptions about how the relay reads events from L2 and L1 chains and interacts with both the `L1BitcoinDepositor` and `L2BitcoinDepositor` smart contracts.

#### Impact

Due to the scope of the audit, we made assumptions about the security and functionality of the Relay Bot. If these assumptions do not hold, this may have critical security implications on the smart contracts.

#### Recommendation

We recommend conducting a rigorous security audit on the implementation of the Relay Bot.

#### Verification Status

The tBTC team acknowledged our recommendation, however, an audit has not been conducted at the time of this verification.



#### ISSUE#2

## An Attacker Can Grief the Relay Bot by Frontrunning the initializeDeposit Function

Medium

Partial

### Location

[L1BitcoinDepositor.sol#L317-L381](#)

[bridge/Deposit.sol#L324](#)

### Description

The `initializeDeposit` function in the `L1BitcoinDepositor` smart contract is used to reveal the user's BTC deposit on the tBTC Bridge on the L1 chain (Ethereum). Anyone, including the Relay Bot, can use this function to initialize the tBTC minting process on tBTC Bridge for BTC deposits revealed on the L2 chain (Base).

In the current implementation, the rough gas cost of the `initializeDeposit` function, in addition to 60,000 gas incentive, will be reimbursed to the caller upon deposit finalization if the reimbursement pool is attached.

If someone front-runs the Relay Bot to call the `initializeDeposit` function, the Relay Bot's transaction will be reverted, wasting a substantial amount of gas since the validation of duplicate deposits is performed very late in the process, in the tBTC Bridge in the `Deposit` smart contract. This is cost-effective to the front-runner, especially since the front-runner can receive the 60,000 incentive gas. Additionally, even if no front-running is intended, in normal cases, the Relay Bot's transactions are susceptible to being reverted due to another potential relay relaying the same deposit, creating a race condition that causes the Relay Bot's transaction to revert.

### Impact

An attacker can grief the Relay Bot by constantly front-running it, causing it to waste a substantial amount of gas since only the successful transactions are reimbursed.

### Recommendation

We recommend adding a procedure to the `initializeDeposit` function to revert early in case of duplicate deposits to prevent gas waste.

The tBTC team suggested a further mitigation whereby anyone is allowed to relay deposits but reimbursement only occurs for a closed set of relayers.

### Verification Status

The tBTC team implemented authorized reimbursements, which can prevent malicious frontrunners. However, the race condition still exists among honest relayers.

#### ISSUE#3

## Relay Bot is Susceptible to Spamming

Medium

Not Fixed

### Location

[contracts/l2/L2BitcoinDepositor.sol#L121-L127](#)





## Description

The Relay Bot is susceptible to spamming by sending dummy or duplicate deposits to the `initializeDeposit` function in the `L2BitcoinDepositor` smart contract since the transaction cost is extremely cheap on the Base blockchain and the body of the `initializeDeposit` function only has one line of code which emits an event for the Relay Bot to pick up the deposit for relaying to `L1BitcoinDepositor` smart contract.

## Impact

An attacker can cause the protocol to waste a substantial amount of gas since the validation of dummy or duplicate deposits is done very late only in the tBTC bridge in the Deposit smart contract.

## Recommendation

We recommend adding a measure to the Relay Bot to filter duplicate or dummy deposits and sending only valid reveals to the `L1BitcoinDepositor` smart contract.

Additionally, we recommend adding a measure to the `initializeDeposit` function to revert early in case of duplicate or dummy deposits to prevent gas waste (see [Issue 2](#)).

## Verification Status

The tBTC team has acknowledged the validity of this issue, noting that the mitigation will be implemented as part of the off-chain relayer bot. Specifically, they note that the relayer bot will validate incoming deposits and call `L1BitcoinDepositor.initializeDeposit` only for the valid deposits. In addition, the bot will make sure that the `initializeDeposit` call will not revert for the given deposit and that the Bitcoin funding transaction of the deposit has at least one confirmation on the Bitcoin chain.

However, the above described mitigation does not fully mitigate the issue since, even in the event that the relayer bot validates only valid deposits, an attacker can still make repeated calls through the smart contract.

ISSUE#4

## No Two-Step Process for Ownership Change

✓ Low

✗ Not Fixed

## Location

[contracts/l2/L1BitcoinDepositor.sol#L21, L18](#)

## Description

The `L1BitcoinDepositor` and `L2BitcoinDepositor` smart contracts use 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.



## Verification Status

The tBTC team has noted that all tBTC contracts use OwnableUpgradeable with a Threshold Council SAFE multisig as actual owner. In addition, they stated their intention to assess the suggested remediation with stakeholders and possibly apply it in the future.

ISSUE#5

### Missing Zero Address Checks

✓ Low

☑ Fixed

#### Location

[contracts/l2/L2BitcoinDepositor.sol#L66-L67](#)

[contracts/l2/L1BitcoinDepositor.sol#L173-L178](#)

#### Description

In the locations referenced above, zero address checks are missing for validating the correctness of those addresses, thereby preventing incorrectly set values.

#### Impact

Incorrectly set values in the referened location would lead to unintended behavior.

#### Recommendation

We recommend adding zero address checks for the aforementioned addresses.

ISSUE#6

### Use Custom Errors to Save Gas

⇩ None

☒ Not Fixed

#### Location

[L1BitcoinDepositor.sol#L163](#)

[L1BitcoinDepositor.sol#L209-L216](#)

[L1BitcoinDepositor.sol#L406-L409](#)

#### Description

The above-referenced statements use `require` and `revert` string messages for error handling. However, using a `revert` with a custom error, instead of a string error message, optimizes gas costs.

#### Impact

None – no security impact.

#### Recommendation

We recommend defining and using custom errors as described in the [Solidity Documentation](#).



## Verification Status

The tBTC team has stated their intention to assess the actual gas impact and possibly apply this suggestion in the next iteration.

ISSUE#7

## Unlocked Pragma Version

None

Fixed

## Location

[contracts/l2][<https://github.com/keep-network/tbtc-v2/blob/d11195be85b03b6c57fca935217aa635abd492c1/solidity/contracts/l2>]

## 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#8

## Use Latest Open Zeppelin Library Implementation

None

Not Fixed

## Location

[package.json#L37](#)

## 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 smart contracts mitigate potential security issues related to the known issue as the project evolves over time.

## Impact

None – no security impact.

## Recommendation

We recommend upgrading the OpenZeppelin library to version 5.0.2.



## Verification Status






The tBTC team has noted that implementing this suggestion immediately is non-trivial as it would require upgrading to OpenZeppelin in the whole solidity module. However, they have noted their intention to explore the possibility of doing so in the future.



# Appendix A

## Severity Rating Definitions

At Thesis Defense, we utilize the [Immunefi Vulnerability Severity Classification System - v2.3](#).

Severity	Definition
 Critical	<ul style="list-style-type: none"> <li>• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results</li> <li>• Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield</li> <li>• Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties</li> <li>• Permanent freezing of funds</li> <li>• Permanent freezing of NFTs</li> <li>• Unauthorized minting of NFTs</li> <li>• Predictable or manipulable RNG that results in abuse of the principal or NFT</li> <li>• Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content)</li> <li>• Protocol insolvency</li> </ul>
 High	<ul style="list-style-type: none"> <li>• Theft of unclaimed yield</li> <li>• Theft of unclaimed royalties</li> <li>• Permanent freezing of unclaimed yield</li> <li>• Permanent freezing of unclaimed royalties</li> <li>• Temporary freezing of funds</li> <li>• Temporary freezing NFTs</li> </ul>
 Medium	<ul style="list-style-type: none"> <li>• Smart contract unable to operate due to lack of token funds</li> <li>• Enabling/disabling notifications</li> <li>• Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol)</li> <li>• Theft of gas</li> <li>• Unbounded gas consumption</li> </ul>
 Low	<ul style="list-style-type: none"> <li>• Contract fails to deliver promised returns, but doesn't lose value</li> </ul>
 None	<ul style="list-style-type: none"> <li>• 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.</li> </ul>



# 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.

