Defense By Thesis

Security Audit Report

Brotocol

Brotocol BridgeEndPoint Smart Contract

Initial Report // August 21, 2025
Final Report // August 25, 2025

Team Members

Krisytiyan Maslarov // Senior Security Auditor

Table of Contents

About Thesis Defense

Defense is the security auditing arm of Thesis, Inc., the venture studio behind tBTC, Fold, Mezo, Acre, Taho, Etcher, and Embody. At Defense, we fight for the integrity and empowerment of the individual by strengthening the security of emerging technologies to promote a decentralized future and user freedom. Defense is the leading Bitcoin applied cryptography and security auditing firm. Our team of security auditors have carried out hundreds of security audits for decentralized systems across a number of ecosystems including Bitcoin, Ethereum + EVMs, Stacks, Cosmos SDK, NEAR and more. We offer our services within a variety of technologies including smart contracts, bridges, cryptography, node implementations, wallets and browser extensions, and dApps.

Defense will employ the 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/Brotocol-xyz/xlink
- **Audit Commit:** `84e3661dbc6cd4e0a16e37849112a6256d7c7bec`
- **Verification Commit:** `7cb7b5f766e3f208c62cefeac6d416e282ca45f0`
- **File in Scope:** BridgeEndPoint.sol

Section 2.0
Executive Summary

Schedule

This security audit was conducted from August 17, 2025 to August 20, 2025 by 1 senior security auditor for a total of 3 person days.

Overview

The BridgeEndPoint smart contract is part of the Brotocol smart contract suite which is deployed on the Mezo blockchain. All other deployed files had been audited previously, as a result, we have conducted an security audit of BridgeEndPoint smart contract to ensure full audit coverage of the Brotocol's Mezo blockchain deployment.

Section 3.0
Key Findings Table

Issues	Severity	Status
ISSUE #1 No Enforcement of 18 Decimal Input Amount	Medium	Acknowledged
ISSUE #2 `setUnwrapSent` Can Arbitrarily Mutate Orders With No Event Emitted	Low	Acknowledged
ISSUE #3 `transferToUnwrap` & `nonReentrant` Don't Adhere to Best Practice	None	Fixed
ISSUE #4 No Validation Against `minAmount` in `setMinFeePerToken` as in `setApprovedToken`	None	Acknowledged
ISSUE #5 `burn` and `mint` Unscaled `amount` in `_transfer` and `transferToUnwrap` Functions	None	Acknowledged

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

No Enforcement of 18 Decimal Input Amount

Medium Acknowledged

Location

BridgeEndpoint.sol#L227

BridgeEndpoint.sol#L167

Description

For tokens with `decimals() < 18` there are certain scenarios where amount can be less than the scaling factor and result in 0 after the calculation.

```
function transferFromFixed(
  ERC20  token.
  address  from,
  address  to.
  uint256  amount
) internal {
  if ( amount > 0)
     token.safeTransferFrom(
       from,
       to.
      _amount / (10 ** (18 - _token.decimals()))
    );
}
```

Impact

Zero transfer on non-zero amount.

Recommendation

We recommend performing the calculation before transferring, and reverting if 0.

Verification Status

Given that no fix has been implemented, the Brotocol team must make users aware that transferring the wrong decimal format could result in loss of funds.

ISSUE#2

`setUnwrapSent` Can Arbitrarily Mutate Orders With No Event Emitted

Low | Acknowledged

Location

BridgeEndpoint.sol#L319

Description

The Owner role can overwrite `recipient`, `token`, `amount`, and `sent` for any order with `registry.orderSent(orderHash) == true`. No event is emitted.

Impact

- The role can mark `sent=true` without an actual transfer, blocking `finalizeUnwrap`.
- The role can change `(token, amount)` so a liquidity provider calling `finalizeUnwrap` later may transfer unexpected assets (it pulls from caller).
- Off-chain indexers lack visibility (no event).

Recommendation

We recommend that an event be emitted(e.g., `SetUnwrapSentEvent(orderHash, recipient, token, amount, sent)`). We also recommend that role permissions be reduced to only allow `sent` to be updated.

Verification Status

The Brotocol temam stated that this issue would be resolved in later development.

ISSUE#3

`transferToUnwrap` & `nonReentrant` Don't Adhere to Best Practice

None Fixed

Location

BridgeEndpoint.sol#L236

Description

Modifiers execute in order. The Role/watchlist checks calls into the `registry` **before** the reentrancy guard is set. If `registry` were compromised, it could attempt reentry earlier.

Impact

Defense-in-depth gap; not an immediate exploit with a correct registry, but an avoidable risk.

Recommendation

We recommend placing `nonReentrant` first:

```
 external
nonReentrant
onlyApprovedRelayer
whenNotPaused
```

ISSUE#4

No Validation Against `minAmount` in `setMinFeePerToken` as in `setApprovedToken`

None Acknowledged

Location

BridgeRegistry.sol#L93 BridgeRegistry.sol#L127

Description

There is an inconsistency in the smart contract validation logic. The setApprovedToken function includes validation to check that:

```
_require(minFee <= minAmount, Errors.MIN_FEE_GREATER_THAN_MIN_AMT);
```

However, the `setMinFeePerToken` function lacks this same validation check:

```
function setMinFeePerToken(
  address  token.
  uint256  minFee
) external onlyOwner {
  minFeePerToken[ token] =  minFee:
  emit SetMinFeePerTokenEvent(_token, _minFee);
}
```

Impact

`minFeePerToken[_token]` could be mistakenly set to values below the intended minimum threshold.

Recommendation

We recommend adding the same `minAmount` validation check to `setMinFeePerToken` to ensure consistent security controls across both functions.

ISSUE#5

`burn` and `mint` Unscaled `amount` in `_transfer` and `transferToUnwrap` Functions

None Acknowledged

Location

BridgeEndpoint.sol#L360 BridgeEndpoint.sol#L258

Description

In the `_transfer` function, the function attempts to `burnFrom` the `msg.sender` an amount as received in the parameters - in 18 decimals:

```
if (registrv.burnable(token)) {
  IBurnable(token).burnFrom(msg.sender, amount.sub(feeDeducted)); // uses fixed
}
```

The same issue exists in `transferToUnwrap` function:

```
if (registrv.burnable(token)) {
    IBurnable(token).mint(address(this). amount):
    ERC20(token).transferFixed(recipient, amount);
 }
```

Impact

When trying to `burn` for example 100 USDC, the amount would incorrectly formed in 18 decimals - 100e18, which is a huge difference from 100e6 which is the 100 USDC in the token's real decimals. This would lead to the DoS of the function for such huge difference due to insufficient balance or allowance of the `msg.sender`.

On the other hand, when minting, the tokens are also received in 18 decimals, so instead of minting 100e6 USDC, you are going to mint 100e18.

In both cases, the amount sent in the `transferFixed` function would be scaled to the real token decimals and the the correct amount would be sent to the user or the `pegAddress`. However when minting, the difference between 100e18 and 100e6 will remain locked in the smart contract.

```
function transferFixed(ERC20 _token, address _to, uint256 _amount) internal {
  if ( amount > 0)
    _token.safeTransfer(_to, _amount / (10 ** (18 - _token.decimals())));
}
```

Recommendation

We recommend scaling to the correct decimals for every token before every burn or mint in order.

Verification Status

The Brotocol team stated that only tokens with the correct decimals will be added to the list of tokens that are approved for burning/

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.