# Mantle Network (Bridge Contracts)

## Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

| | |
|---|---|
| Type | Bridge Contracts |
| Timeline | 2023-06-02 through 2023-06-16 |
| Language | Solidity |
| Methods | Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review |
| Specification | package/contracts/docs |
| Source Code | • mantlenetworkio/mantle #d627d24 |

| | |
|---|---|
| Documentation quality | Medium |
| Test quality | Undetermined |
| Total Findings | 5<br>Fixed: 1 Acknowledged: 4 |
| High severity findin | 1<br>Acknowledged: 1 |

| | | | |
|---|---|---|---|
| | | gs | |
| Auditors | • Guillermo Escobero Auditing Engineer<br>• Mustafa Hasan Senior Auditing Engineer<br>• Julio Aguliar Auditing Engineer<br>• Roman Rohleder Senior Auditing Engineer | Medium severity findings | 1<br>**Acknowledged: 1** |
| | | Low severity findings | 2<br>**Fixed: 1** **Acknowledged: 1** |
| | | Undetermineds | 0 |

| | |
|---|---|
| Informational findings | 1<br>**Acknowledged: 1** |

# Summary of Findings

Quantstamp performed an audit for the Mantle Network bridge contracts based on the code present in the listed repositories. The code is forked off from optimism at commit hash afdd020.

During the audit, we found some issues that pose a risk to users of the Mantle bridge when bridging non-compatible tokens. In particular, MANB-1 describes the possibility of locking funds if the users interact with the bridge contracts by sending the wrong parameters or non-compatible ERC-20 tokens.

Lower severity issues were found in the auditing process also. All of them are discussed in this report, including some best practices recommendations. We recommend addressing all of them.

Regarding testing, the reader is referred to MANB-2. No test suite was provided to perform unit testing of the contracts in scope. Although the Mantle team states that

functional testing is ongoing in testnet deployments, we recommend implementing unit tests as well to get code coverage metrics. This will help to cover all possible paths in the codebase.

The audited contracts integrate with Mantle Network cross-chain messaging system. This system is out of scope.

**After fix review:** The developers addressed all issues by either fixing or acknowledging them. Issue MANB-2 (Missing Test Suite) was acknowledged by the Mantle team. We still recommend implementing a proper test suite and code coverage metrics.

| ID | DESCRIPTION | SEVERITY | STATUS |
|---|---|---|---|
| MANB-1 | Not Compatible ERC-20 Tokens | High | Acknowledged |
| MANB-2 | Missing/Limited Test Suite | Medium | Acknowledged |
| MANB-3 | Missing Input Validation | Low | Fixed |
| MANB-4 | Implementation Contract Can Be Initialized | Low | Acknowledged |
| MANB-5 | Unlocked Pragma | Informational | Acknowledged |

# Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

### Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

> All cross-chain transport layer and future bridged L1/L2 tokens are out of the scope of this audit.

**Possible issues we looked for included (but are not limited to):**

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

**Methodology**

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
    1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

# Scope

All cross-chain transport layer and future bridged L1/L2 tokens are out of the scope of this audit.

**Files Included**

- `packages/contracts/contracts/L1/messaging/L1StandardBridge.sol`
- `packages/contracts/contracts/L2/messaging/L2StandardBridge.sol`

# Findings

## MANB-1
## Not Compatible ERC-20 Tokens

High Acknowledged

> **Update**
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > Token transfers through the official canonical bridge require registration through our token list.

**File(s) affected:** `L1StandardBridge.sol` , `L2StandardBridge.sol`

**Description:** The accounting approach for deposits in `L1StandardBridge.sol` and burning and minting operations in `L2StandardBridge.sol` are not compatible with ERC-20 tokens that (but not limited to):

- Have a transfer, minting, or burning fee
- Are deflationary/inflationary (i.e. the balance of an account can change "arbitrarily", without a transfer operation)
- Its L2 version unexpectedly reverts when minting, burning, or transferring
- Its L2 version does not implement minting and burning operations properly (e.g. mints or burns more or fewer tokens than expected, leading to duplicate funds or issues when withdrawing from L2)
- Implement blocklists

Currently, the system allows anyone to deploy a compatible token in L2 and link it to an L1 token. Users are responsible for ensuring that the ERC-20 L2 token is correct and that its behavior is not in the abovementioned list. **Interacting with a non-compatible token will block user funds in** `L1StandardBridge.sol` .

**Recommendation:**

1. Users should be informed of this risk when interacting with the smart contracts directly (Optimism Documentation). If a user interface is expected to be the main access to the system, the available tokens should be verified by the Mantle team. We strongly recommend showing an alert when interacting with non-verified (and potentially non-compatible) tokens.
2. Consider implementing a token allowlist and only including tokens validated by the Mantle team.

## MANB-2 Missing/Limited Test Suite

Medium Acknowledged

> **Update**
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > We will improve on it gradually.

**Description:** Although a test suite in the repository exists, it is from the original Optimism repository. The Mantle team stated that they do not have unit tests at this moment that cover the Mantle bridge contracts.

**Recommendation:** We strongly recommend adding unit tests (e.g. similar to the existing ones or adapting them).

## MANB-3 Missing Input Validation

Low Fixed

> **Update**
> Marked as "Fixed" by the client. Addressed in: `a53dd956c6a1330742c00f46f30aee881f76b958` .

**File(s) affected:** `L1StandardBridge.sol` , `L2StandardBridge.sol`

**Related Issue(s):** SWC-123

**Description:** It is important to validate input data even if they come from trusted sources to reduce human error:

1. `L1StandardBridge.initialize()` needs to validate `_l2TokenBridge` and `_l1BitAddress` against `address(0)` as well since there is no setter for them.

2. `L2StandardBridge.constructor()` needs to validate `_l2CrossDomainMessenger` and `_l1TokenBridge` against `address(0)` as well since there is no setter for them.

**Recommendation:** We recommend adding the mentioned validations.

## MANB-4
## Implementation Contract Can Be Initialized

Low Acknowledged

> **Update**
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > We have resolved it during contract deployment.

**File(s) affected:** `L1StandardBridge.sol`

**Description:** `L1StandardBridge.initialize()` is a public function with no access control. If `L1StandardBridge` is deployed behind a proxy, anyone can initialize the implementation contract by calling `initialize()` with arbitrary values.

Although this does not affect the functionality of the project, it may open possible phishing or social engineering attacks (e.g. an attacker can try to impersonate Mantle).

If the contract is not deployed using a proxy, there is a possibility of front-running. After contract deployment, an attacker can call `initialize()`, needing to deploy the contract again.

**Recommendation:** Once deployed, make sure that the implementation contract is initialized to a random value (e.g. zero addresses). If no proxy is used, we recommend deploying and initializing the contract in a single transaction. The use of `Initializable` from OpenZeppelin can mitigate this also.

## MANB-5
## Unlocked Pragma

Informational Acknowledged

> **Update**
> Marked as "Acknowledged" by the client. The client provided the following explanation:

> We plan to fix it in future releases. Implementing a new Solidity version immediately may introduce other unintended issues.

**File(s) affected:** `L1StandardBridge.sol` , `L2StandardBridge.sol`

**Related Issue(s):** SWC-103

**Description:** Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*` . The caret ( `^` ) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

**Recommendation:** For consistency and to prevent unexpected behavior in the future, we recommend removing the caret to lock the file onto a specific Solidity version. Consider using `pragma solidity 0.8.18` , the recommended compiler version at this moment.

# Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

# Adherence to Best Practices

1. Instead of hardcoding constant values (commonly known as "magic numbers"), declare them as constants:
    - `L1StandardBridge._initiateERC20Deposit()` uses the hardcoded address of the `BIT` token. Add it as a constant to either the `L1StandardBridge` contract or to the `Lib_PredeployAddresses`.
    - `L1StandardBridge.finalizeDeposit()` uses the hardcoded interface ID for the `l2Token` contract. We recommend using the corresponding `IL2StandardERC20.interfaceId()`.

# Appendix

### File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

### Contracts

- `e1f...ac0 ./L2StandardBridge.sol`
- `f7a...79d ./L1StandardBridge.sol`

# Toolset

The notes below outline the setup and steps performed in the process of this audit.

### Setup

Tool Setup:
- Slither v0.9.3

Steps taken to run the tools:
1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

# Automated Analysis

### Slither

Slither was used to get a static analysis of the repository. All the issues and recommendations are discussed in this report or classified as false positives.

# Changelog

- 2023-06-16 - Initial report
- 2023-07-06 - Final report
- 2023-07-13 - Updated clarifications from Mantle team

# About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

**Timeliness of content**

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and

assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

### Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

### Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

### Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or

sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.


 Mantle Network (Bridge Contracts)