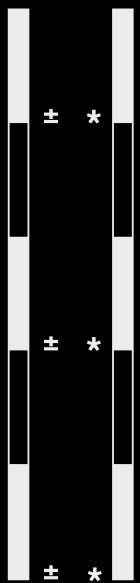
VERILOG

2023

MANTLE L2 ERC20 TOKEN BRIDGE:

SECURITY REVIEW REPORT



Sep 28, 2023

Contents

1 | Introduction

Figure 1.1: Mantle L2 Token Bridge Report Cover

This report presents our engineering engagement with the Mantle team on their development of L2 ERC20 Token Bridge.

Project Name	**Mantle L2 ERC20 Token Bridge**
Repository Link	https://github.com/mantlenetworkio/lido-l2/tree/main
First Commit Hash	First: 0ab4478;
Final Commit Hash	Final: cdd513c;
Language	Solidity
Chain	Mantle

2 | About Verilog Solutions

Founded by a group of cryptography researchers and smart contract engineers in North America, Verilog Solutions elevates the security standards for Web3 ecosystems by being a full-stack Web3 security firm covering smart contract security, consensus security, and operational security for Web3 projects.
Verilog Solutions team works closely with major ecosystems and Web3 projects and applies a quality above quantity approach with a continuous security model. Verilog Solutions onboards the best and most innovative projects and provides the best-in-class advisory services on security needs, including on-chain and off-chain components.

3 | Service Scope

3.1 | Service Stages

Our auditing service includes the following two stages:

- Smart Contract Auditing Service

3.1.1 | Smart Contract Auditing Service

The Verilog Solutions team analyzed the entire project using a detailed-oriented approach to capture the fundamental logic and suggested improvements to the existing code. Details can be found under Findings And Improvement Suggestions.

3.2 | Methodology

- **Code Assessment**
  - We evaluate the overall quality of the code and comments as well as the architecture of the repository.
  - We help the project dev team improve the overall quality of the repository by providing suggestions on refactorization to follow the best practices of Web3 software engineering.
- **Code Logic Analysis**
  - We dive into the data structures and algorithms in the repository and provide suggestions to improve the data structures and algorithms for the lower time and space complexities.
  - We analyze the hierarchy among multiple modules and the relations among the source code files in the repository and provide suggestions to improve the code architecture with better readability, reusability, and extensibility.
- **Business Logic Analysis**
  - We study the technical whitepaper and other documents of the project and compare its specifications with the functionality implemented in the code for any potential mismatch between them.
  - We analyze the risks and potential vulnerabilities in the business logic and make suggestions to improve the robustness of the project.
- **Access Control Analysis**
  - We perform a comprehensive assessment of the special roles of the project, including their authorities and privileges.
  - We provide suggestions regarding the best practice of privilege role management according to the standard operating procedures (SOP).
- **Off-Chain Components Analysis**
  - We analyze the off-chain modules that are interacting with the on-chain functionalities and provide suggestions according to the SOP.
  - We conduct a comprehensive investigation for potential risks and hacks that may happen on the off-chain components and provide suggestions for patches.

3.3 | Audit Scope

Our auditing for Mantle L2 ERC20 Token Bridge covered the Solidity smart contracts under the folder 'contracts' in the repository(https://github.com/mantlenetworkio/lido-l2) with commit hash **0ab4478**.

4 | Findings and Improvement Suggestions

Severity	Total	Acknowledged	Resolved
High	0	0	0
Medium	0	0	0
Low	2	2	0
Informational	3	3	3

4.1 | Low

4.1.1 | Missing modifiers

Severity	Low
Source	mantle/L2ERC20TokenBridge.sol#L53;
Commit	0ab4478;
Status	Acknowledged;

- **Description**
  The `withdrawTo()` function should validate that the `to_` address is not the zero address like the `depositERC20To()` function does by using the `onlyNonZeroAccount` modifier.
- **Exploit Scenario**
  N/A.
- **Recommendations**
  Add the `onlyNonZeroAccount` modifier to the `withdrawTo()` function.
- **Results**
  Acknowledged.
  Team's Response:
  'The findings were derived from the original Lido L2 implementation and will be documented to mitigate any issues.'

4.1.2 | Lack of zero address checks

Severity	Low
Source	BridgeableTokens.sol#L18-L19; L1ERC20TokenBridge.sol#L42; L2ERC20TokenBridge.sol#L39;
Commit	0ab4478;
Status	Acknowledged;

- **Description**
  The `constructor()` of the `BridgeableTokens`, `L1ERC20TokenBridge`, and `L2ERC20TokenBridge` contracts don't consider zero address checks when defining the address of the tokens and bridges. Adding these checks can prevent incorrect deployment.
- **Exploit Scenario**
  N/A.
- **Recommendations**
  Add zero address checks in the `constructor()` of the mentioned contracts.
- **Results**
  Acknowledged.
  Team's Response:
  'The findings were derived from the original Lido L2 implementation and will be documented to mitigate any issues.'

4.2 | Informational

4.2.1 | The domain separator can be cached instead of being constructed each time a permit is executed

Severity	Informational
Source	contracts/token/ERC20BridgedPermit.sol#L20;
Commit	0ab4478;
Status	Resolved;

- **Description**
  `ERC20BridgedPermit` contract can cache the `address(this)` address and `chainid` that are used in the domain separator. We can only build a new domain separator when the current `address(this)` address and `chainid` are different from cached ones.

- **Exploit Scenario**
  N/A.

- **Recommendations**
  Example from Openzeppelin contracts

```
function _domainSeparatorV4() internal view returns (bytes32) {
     if (address(this) == _cachedThis && block.chainid == _cachedChainId) {
         return _cachedDomainSeparator;
     } else {
         return _buildDomainSeparator();
     }
 }
```

- **Results**
  Resolved at commit hash d5d8cd5.

4.2.2 | Withdrawal functions in L2ERC20TokenBridge contract do not verify if the sender is an EOA

Severity	Informational
Source	contracts/mantle/L2ERC20TokenBridge.sol#L43; contracts/mantle/L2ERC20TokenBridge.sol#L53;
Commit	0ab4478;
Status	Acknowledged and partially resolved;

- **Description**
  The `L1ERC20TokenBridge`. `depositERC20()` function requires the sender to be an EOA address, whereas the `withdraw()` and `withdrawTo()` function in the `L2ERC20TokenBridge` contract don't.
- **Exploit Scenario**
  N/A.
- **Recommendations**
  We would like to confirm this with the team.
- **Results**
  Acknowledged and partially resolved.
  Team's Response:
  'withdraw() has been limited to EOAs, however, withdrawTo() was intended to allow both EOA and Smart Contract Wallets.

  commit hash: lidofinance/lido-l2@cdd513c'

4.2.3 | Suggest adding a helper function to increase nonce in `ERC20BridgedPermit.sol`

Severity	Informational
Source	contracts/token/ERC20BridgedPermit.sol#L12;
Commit	0ab4478;
Status	Resolved;

- **Description**
  We recommend adding a function that allows users to increment the nonce. This way, users can safely invalidate previously signed signatures by simply calling this function.

  For example, OpenZeppelin contracts' `ERC20Permit.sol`, includes a function for nonce consumption. Token contracts that inherit from `ERC20Permit.sol` can expose this function to users, enabling them to increment the nonce securely.

  Without such a function, users would need to invalidate their previously signed signatures by generating a new signature and executing it to increment the nonce.

- **Exploit Scenario**
  N/A.

- **Recommendations**
  We suggest adding a function that allows users to increase the nonce. This enables users to safely invalidate previously signed signatures by simply calling this function, without the need to create a new signature and execute it to invalidate the old ones.

- **Results**
  Resolved in commit c02d936.

5 | Notes and Additional Information

5.1 | Lido L2 Docs

Source: https://github.com/lidofinance/lido-l2;

Current implementation of Mantle L2 ERC20 Token Bridge is developed based on top of the Lido-l2 repo. Lido L2 repo contains the implementations of the L2 ERC20 token bridges for Arbitrum and Optimism chains. The current solution allows transferring ERC20 tokens between L1 and L2 chains.
To retrieve more detailed info about the bridging process, see the specifications for certain chains:

- **Lido's Arbitrum Gateway**:
https://github.com/lidofinance/lido-l2/blob/main/contracts/arbitrum/README.md
- **Lido's Optimism Bridge**:
https://github.com/lidofinance/lido-l2/blob/main/contracts/optimism/README.md

6 | Appendix

6.1 | Appendix I: Severity Categories

Severity	Description
High	Issues that are highly exploitable security vulnerabilities. It may cause direct loss of funds / permanent freezing of funds. All high severity issues should be resolved.
Medium	Issues that are only exploitable under some conditions or with some privileged access to the system. Users' yields/rewards/information is at risk. All medium severity issues should be resolved unless there is a clear reason not to.
Low	Issues that are low risk. Not fixing those issues will not result in the failure of the system. A fix on low severity issues is recommended but subject to the clients' decisions.
Information	Issues that pose no risk to the system and are related to the security best practices. Not fixing those issues will not result in the failure of the system. A fix on informational issues or adoption of those security best practices-related suggestions is recommended but subject to clients' decision.

6.2 | Appendix II: Status Categories

Severity	Description
Unresolved	The issue is not acknowledged and not resolved.
Partially Resolved	The issue has been partially resolved
Acknowledged	The Finding / Suggestion is acknowledged but not fixed / not implemented.
Resolved	The issue has been sufficiently resolved

7 | Disclaimer

Verilog Solutions receives compensation from one or more clients for performing the smart contract and auditing analysis contained in these reports. The report created is solely for Clients and published with their consent. As such, the scope of our audit is limited to a review of code, and only the code we note as being within the scope of our audit is detailed in this report. It is important to note that the Solidity code itself presents unique and unquantifiable risks since the Solidity language itself remains under current development and is subject to unknown risks and flaws. Our sole goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies. Thus, Verilog Solutions in no way claims any guarantee of security or functionality of the technology we agree to analyze.

In addition, Verilog Solutions reports do not provide any indication of the technology's proprietors, business, business model, or legal compliance. As such, reports do not provide investment advice and should not be used to make decisions about investment or involvement with any particular project. Verilog Solutions has the right to distribute the Report through other means, including via Verilog Solutions publications and other distributions. Verilog Solutions makes the reports available to parties other than the Clients (i.e., "third parties") – on its website in hopes that it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.