Quantstamp

Zircuit / Lido / NEW Proposal

Executive Summary

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

Type	Configuration Review
Timeline	2024-09-23 through 2024-09-23
Language	Solidity
Methods	A combination of manual and automated review is used to determine the difference between the deployed and audited code. Manual checks are also performed to confirm that the field values of deployed contracts match the field values in the proposal.
Specification	wsteth-bridging-guide
Source Code	• https://github.com/lidofinance/lido-l2/ • https://github.com/aave/governance-crosschain-bridges
Auditors	• Mostafa Yassin Auditing Engineer • Cameron Biniamow Auditing Engineer

Documentation quality	Undetermined
Test quality	Undetermined
Total Findings	0
High severity findings	0
Medium severity findings	0
Low severity findings	0
Undetermined severity findings	0
Informational findings	0

Summary of Findings

This engagement is meant to verify the deployment code and smart contract field values of Zircuit's proposal to deploy `wstETH` on Zircuit L2. The engagement had the following steps:

Part 1

Identify any code changes made in the listed files between the audited commit hash and the deployment commit hash for the following repositories.

Repo: https://github.com/lidofinance/lido-l2

- Audit Report
- Audited Commit Hash: `082e7eb59de63bd376b30886568813408d04f00b`
- Deployment Commit Hash: `d8b68db14a98d49aeff20bfd4ccd581a02ed3f48`
- Files:
  - `contracts/optimism/L1ERC20TokenBridge.sol`
  - `contracts/proxy/OssifiableProxy.sol`
  - `contracts/token/ERC20Bridged.sol`
  - `contracts/optimism/L2ERC20TokenBridge.sol`

Changes

No code changes were identified.

Deployment Solidity Version

The `lido-l2` contracts were compiled and deployed with Solidity version `0.8.10+commit.fc410830`. Below is a list of known issues that exist in `0.8.10`:

- AbiReencodingHeadOverflowWithStaticArrayCleanup
    - Medium Severity
    - Introduced in `0.5.8`
    - Fixed in `0.8.16`
- AbiReencodingHeadOverflowWithStaticArrayCleanup
    - Low Severity
    - Introduced in `0.5.8`
    - Fixed in `0.8.16`
- DirtyBytesArrayToStorage
    - Low Severity
    - Introduced in `0.0.1`
    - Fixed in `0.8.15`
- DataLocationChangeInInternalOverride
    - Very Low Severity
    - Introduced in `0.6.9`
    - Fixed in `0.8.14`
- NestedCalldataArrayAbiReencodingSizeValidation
    - Very Low Severity
    - Introduced in `0.5.8`
    - Fixed in `0.8.14`

For more details about every issue, please check this link.

Repo: https://github.com/aave/governance-crosschain-bridges

- Audit Report
- Audited Commit Hash: `8fa25b0080dd3dcc2390313631aea6796a12c9d8`
- Deployment Commit Hash: `57dd43969a68eb076fdbeae2953b572534694986`
- Files:
    - `contracts/bridges/OptimismBridgeExecutor.sol`
    - `contracts/bridges/L2BridgeExecutor.sol`
    - `contracts/bridges/BridgeExecutorBase.sol`

Changes

The engagement concluded that the code at the audited commit hash matches the code at the deployment commit hash **with the only difference being in the pragma (solidity version)**.

The following are the pragma updates:

- `contracts/bridges/BridgeExecutorBase.sol` - `0.8.10` to `^0.8.10`
- `contracts/bridges/L2BridgeExecutor.sol` - `0.8.10` to `^0.8.10`
- `contracts/bridges/OptimismBridgeExecutor.sol` - `0.8.10` to `^0.8.10`

Deployment Solidity Version

The `governance-crosschain-bridges` contracts were compiled and deployed with Solidity version `0.8.20+commit.a1b79de6`, which does not impact the protocol's security. Below is a list of known issues that exist in `0.8.20`:

- VerbatimInvalidDeduplication
    - Low Severity
    - Introduced in `0.8.5`
    - Fixed in `0.8.23`
- FullInlinerNonExpressionSplitArgumentEvaluationOrder
    - Low Severity
    - Introduced in `0.6.7`
    - Fixed in `0.8.21`

For more details about every issue, please check this link.

Part 2

Verify that the on-chain contracts were deployed using the deployment commit hash for both Mainnets and Testnets.

Mainnet
Ethereum

- `L1ERC20TokenBridge.sol`
    - Implementation
        - Address: 0×6bc726C993103197C41d787dd72eCd4D2e1614E8

        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/optimism/L1ERC20TokenBridge.sol
    - OssifiableProxy
        - Address: 0×912C7271a6A3622dfb8B218eb46a6122aB046C79
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/proxy/OssifiableProxy.sol

Zircuit

- `ERC20Bridged`
    - Implementation
        - Address: 0×929569e10d9166f31c8284fE3FE5db1C1E56D6b4
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/token/ERC20Bridged.sol
    - OssifiableProxy:
        - Address: 0xf0e673Bc224A8Ca3ff67a61605814666b1234833
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/proxy/OssifiableProxy.sol
- `L2ERC20TokenBridge`
    - Implementation:
        - Address: 0×224F00AEDD7A9F10e571898662ad19CD5abd9F2c
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/optimism/L2ERC20TokenBridge.sol
    - OssifiableProxy:
        - Address: 0xF4DC271cA48446a5d2b97Ff41D39918DF8A4Eb0e
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/proxy/OssifiableProxy.sol

Testnet
Ethereum Sepolia

- `L1ERC20TokenBridge.sol`
    - Implementation
        - Address: 0×0b72F930bb0e378b19E93eBadf1c563D28A584ed
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/optimism/L1ERC20TokenBridge.sol
    - OssifiableProxy
        - Address: 0×130424c81a7d497Efa53bc71BB8B718202087726
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/proxy/OssifiableProxy.sol

Zircuit Testnet

- `ERC20Bridged`
    - Implementation
        - Address: 0×549aF13787A46eF63341c8C7e78691F4a2bFbE48
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/token/ERC20Bridged.sol
    - OssifiableProxy:
        - Address: 0×6b8116B41bFd7e1A976cB892acB79926080A6Ca1
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/proxy/OssifiableProxy.sol
- `L2ERC20TokenBridge`
    - Implementation:
        - Address: 0×247f56cFc9021aeC161a4366412636ea33101D2B
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/optimism/L2ERC20TokenBridge.sol
    - OssifiableProxy:
        - Address: 0×7721F53d153Ae3CF937605fF1Bbb7D51B14E7902
        - Code: https://github.com/lidofinance/lido-l2/blob/d8b68db14a98d49aeff20bfd4ccd581a02ed3f48/contracts/proxy/OssifiableProxy.sol

Part 3

Verify that the deployed contract fields match the values in the proposal. The engagement concluded that this is indeed the case.

Mainnet
Ethereum

Key addresses:

- Lido DAO Agent: `0x3e40D73EB977Dc6a537aF587D48316feE66E9C8c`
- Lido Emergency Brakes Multisig on L1: `0x73b047fe6337183A454c5217241D780a932777bD`

Levers Setup:

- `L1ERC20TokenBridge.sol`
  - Admin role of `DEFAULT_ADMIN_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `WITHDRAWALS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `WITHDRAWALS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `DEPOSITS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `DEPOSITS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - `proxy_getAdmin()` is the Lido DAO Agent.
  - Lido DAO Agent has the `DEFAULT_ADMIN_ROLE`.
  - Lido DAO Agent has the `WITHDRAWALS_DISABLER_ROLE`.
  - Lido DAO Agent has the `WITHDRAWALS_ENABLER_ROLE`.
  - Lido DAO Agent has the `DEPOSITS_DISABLER_ROLE`.
  - Lido DAO Agent has the `DEPOSITS_ENABLER_ROLE`.
  - Lido Emergency Brakes Multisig on L1 has the `WITHDRAWALS_DISABLER_ROLE`.
  - Lido Emergency Brakes Multisig on L1 has the `DEPOSITS_DISABLER_ROLE`.

Zircuit

Key Addresses:

- Lido DAO Agent: `0x3e40D73EB977Dc6a537aF587D48316feE66E9C8c`
- `OptimismBridgeExecutor`: `0x6Bf2cac3ed2481da30aD36Cd3D64325c31065Cc5`
- Lido Emergency Brakes Multisig on L2: `0x9Bff79BF7226cB5C16d0Cca9c1dc60450feE560d`

Levers Setup:

- `OptimismBridgeExecutor.sol`
  - `getEthereumGovernanceExecutor()` is the Lido DAO Agent.
- `L2ERC20TokenBridge.sol`
  - Admin role of `DEFAULT_ADMIN_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `WITHDRAWALS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `WITHDRAWALS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `DEPOSITS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `DEPOSITS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - `proxy_getAdmin()` is the `OptimismBridgeExecutor` contract.
  - `OptimismBridgeExecutor` contract has the `DEFAULT_ADMIN_ROLE`.
  - `OptimismBridgeExecutor` contract has the `WITHDRAWALS_DISABLER_ROLE`.
  - `OptimismBridgeExecutor` contract has the `WITHDRAWALS_ENABLER_ROLE`.
  - `OptimismBridgeExecutor` contract has the `DEPOSITS_DISABLER_ROLE`.
  - `OptimismBridgeExecutor` contract has the `DEPOSITS_ENABLER_ROLE`.
  - Lido Emergency Brakes Multisig on L2 has the `WITHDRAWALS_DISABLER_ROLE`.
  - Lido Emergency Brakes Multisig on L2 has the `DEPOSITS_DISABLER_ROLE`.

Testnet
Ethereum Sepolia

Key Addresses:

- Lido DAO Agent: `0x32A0E5828B62AAb932362a4816ae03b860b65e83`
- Lido Emergency Brakes Multisig on L1: `0xa5F1d7D49F581136Cf6e58B32cBE9a2039C48bA1`

Levers Setup:

- `L1ERC20TokenBridge.sol`
  - Admin role of `DEFAULT_ADMIN_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `WITHDRAWALS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `WITHDRAWALS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `DEPOSITS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - Admin role of `DEPOSITS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
  - `proxy_getAdmin()` is the Lido DAO Agent.
  - Lido DAO Agent has the `DEFAULT_ADMIN_ROLE`.
  - Lido DAO Agent has the `WITHDRAWALS_DISABLER_ROLE`.
  - Lido DAO Agent has the `WITHDRAWALS_ENABLER_ROLE`.
  - Lido DAO Agent has the `DEPOSITS_DISABLER_ROLE`.
  - Lido DAO Agent has the `DEPOSITS_ENABLER_ROLE`.
  - Lido Emergency Brakes Multisig on L1 has the `WITHDRAWALS_DISABLER_ROLE`.
  - Lido Emergency Brakes Multisig on L1 has the `DEPOSITS_DISABLER_ROLE`.

Zircuit Testnet

Key Addresses:

- Lido DAO Agent: `0x32A0E5828B62AAb932362a4816ae03b860b65e83`
- `OptimismBridgeExecutor`: `0x989CD486c02bfBe5c2D3C157cDCab099134e7697`
- Lido Emergency Brakes Multisig on L2: `0xa5F1d7D49F581136Cf6e58B32cBE9a2039C48bA1`

Levers Setup:

- `OptimismBridgeExecutor.sol`

    - `getEthereumGovernanceExecutor()` is the Lido DAO Agent.
- `L2ERC20TokenBridge.sol`
    - Admin role of `DEFAULT_ADMIN_ROLE` is `DEFAULT_ADMIN_ROLE`.
    - Admin role of `WITHDRAWALS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
    - Admin role of `WITHDRAWALS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
    - Admin role of `DEPOSITS_DISABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
    - Admin role of `DEPOSITS_ENABLER_ROLE` is `DEFAULT_ADMIN_ROLE`.
    - `proxy_getAdmin()` is the `OptimismBridgeExecutor` contract.
    - `OptimismBridgeExecutor` contract has the `DEFAULT_ADMIN_ROLE`.
    - `OptimismBridgeExecutor` contract has the `WITHDRAWALS_DISABLER_ROLE`.
    - `OptimismBridgeExecutor` contract has the `WITHDRAWALS_ENABLER_ROLE`.
    - `OptimismBridgeExecutor` contract has the `DEPOSITS_DISABLER_ROLE`.
    - `OptimismBridgeExecutor` contract has the `DEPOSITS_ENABLER_ROLE`.
    - Lido Emergency Brakes Multisig on L2 has the `WITHDRAWALS_DISABLER_ROLE`.
    - Lido Emergency Brakes Multisig on L2 has the `DEPOSITS_DISABLER_ROLE`.

Assessment Breakdown

Quantstamp's objective was to determine if the deployed code matches the specifications outlined in the proposal. Which is indeed the case.

> **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.

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

- Changes between the deployed code and the audited code; other than pragma (solidity version)
- Discrepancies between the deployed contract field values and the field values in the proposal.

Methodology

The engagement team compared the diff between the deployed code and the code located in the commits specified in the audit reports. Then, the team manually checked that the deployed contract fields' values matched those in the proposal document.

Findings

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).

Automated Analysis

N/A

Changelog

- 2024-09-24 - Initial report

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