LIDO A.DI SECURITY AUDIT REPORT

Jul 04, 2024

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e. Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

The Lido Delivery Infrastructure protocol is a fork of the Aave a.DI protocol with a modification that incorporates the usage of a well-known `CrossChainExecutor` contract for executing Lido DAO proposals on different chains. Currently, the protocol is to be used on BSC chain only. It is important to note that message passing is planned to be only in one direction from Ethereum to other networks. Still, the protocol architecture allows Lido to set up message passing to Ethereum in the future.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Lido
Project name	Delivery Infrastructure
Timeline	April 02 2024 - July 04 2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
02.04.2024	41c81975c2ce5b430b283e6f4aab922c3bde1555	Commit for the audit (Delivery Infrastructure)
02.04.2024	56720adb8b751c3742a5f9b8c5bcb8a9446ba941	Commit for the audit (solidity-utils)

Project Scope

The audit covered the following files:

File name	Link
src/contracts/BaseCrossChainController.sol	BaseCrossChainController.sol
src/contracts/CrossChainController.sol	CrossChainController.sol
src/contracts/CrossChainForwarder.sol	CrossChainForwarder.sol

File name	Link
src/contracts/CrossChainReceiver.sol	CrossChainReceiver.sol
src/contracts/adapters/BaseAdapter.sol	BaseAdapter.sol
src/contracts/adapters/ccip/CCIPAdapter.sol	CCIPAdapter.sol
src/contracts/adapters/ccip/ICCIPAdapter.sol	ICCIPAdapter.sol
src/contracts/adapters/ccip/lib/Client.sol	Client.sol
src/contracts/adapters/hyperLane/HyperLaneAdapter.sol	HyperLaneAdapter.sol
src/contracts/adapters/hyperLane/libs/StandardHookMetadata.sol	StandardHookMetadata.sol
src/contracts/adapters/hyperLane/libs/TypeCasts.sol	TypeCasts.sol
src/contracts/adapters/layerZero/LayerZeroAdapter.sol	LayerZeroAdapter.sol
src/contracts/adapters/layerZero/libs/BytesLib.sol	BytesLib.sol
src/contracts/adapters/layerZero/libs/ExecutorOptions.sol	ExecutorOptions.sol
src/contracts/adapters/layerZero/libs/OptionsBuilder.sol	OptionsBuilder.sol
src/contracts/adapters/polygon/PolygonAdapterBase.sol	PolygonAdapterBase.sol
src/contracts/adapters/polygon/PolygonAdapterEthereum.sol	PolygonAdapterEthereum.sol
src/contracts/adapters/polygon/PolygonAdapterPolygon.sol	PolygonAdapterPolygon.sol
src/contracts/adapters/polygon/tunnel/FxTunnelEthereum.sol	FxTunnelEthereum.sol
src/contracts/adapters/polygon/tunnel/FxTunnelPolygon.sol	FxTunnelPolygon.sol
src/contracts/adapters/wormhole/WormholeAdapter.sol	WormholeAdapter.sol
src/contracts/libs/ChainIds.sol	ChainIds.sol
src/contracts/libs/EncodingUtils.sol	EncodingUtils.sol

File name	Link
src/contracts/libs/Errors.sol	Errors.sol
src/Lido/contracts/BridgeExecutorBase.sol	BridgeExecutorBase.sol
src/Lido/contracts/CrossChainExecutor.sol	CrossChainExecutor.sol
src/contracts/access-control/OwnableWithGuardian.sol	OwnableWithGuardian.sol
src/contracts/utils/Rescuable.sol	Rescuable.sol

Deployments

Ethereum:mainnet

File name	Contract deployed on mainnet	Comment
CrossChainController.sol	0x5f456f...953f2c63	
TransparentUpgradeableProxy.sol	0x935598...3c369A7C	OpenZeppelin v4.9.0
ProxyAdmin.sol	0xADD673...31A6fDeE	OpenZeppelin v4.8.3
WormholeAdapter.sol	0xEDc0D2...7eAbDF17	
LayerZeroAdapter.sol	0x742650...1fB54411	
HyperLaneAdapter.sol	0x8d374D...09b3a7F3	
CCIPAdapter.sol	0x29D4fA...6F597d5d	

BSC:mainnet

File name	Contract deployed on mainnet	Comment
CrossChainController.sol	0xB7Ba81dd07885ae7BFD18452B36D3404d7EDD8Ee	
TransparentUpgradeableProxy.sol	0x40C4464fCa8caCd550C33B39d674fC257966022F	OpenZeppelin v4.9.0
CrossChainExecutor.sol	0x8E5175D17f74d1D512de59b2f5d5A5d8177A123d	
ProxyAdmin.sol	0x29E6817db339795766244B96aEf5Dc534a98518d	OpenZeppelin v4.8.3
WormholeAdapter.sol	0xBb1E43408BbF2C767Ff3Bd5bBC34E183CC1Ef119	
LayerZeroAdapter.sol	0xc934433f4c433Cf80DE6fB65fd70C7a650D8a408	
HyperLaneAdapter.sol	0xCd867B440c726461e5fAbe8d3a050b2f8701C230	
CCIPAdapter.sol	0x15AD245133568c2498c7dA0cf2204A03b0e9b98A	

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	2
Low	11

ID	Name	Severity	Status
M-1	Expired proposals can be executed	Medium	Acknowledged
M-2	The guardian in `BridgeExecutorBase` can cancel an `ActionsSet` that changes the guardian	Medium	Acknowledged
L-1	The implementation can be destroyed by the `owner`	Low	Acknowledged
L-2	Inconsistent zero checks	Low	Acknowledged
L-3	An unnecessary storage update	Low	Acknowledged
L-4	`_trustedRemotes` cannot be updated	Low	Acknowledged
L-5	Unrestricted `forwardMessage` and `setupPayments`	Low	Acknowledged
L-6	Unrestricted `sendMessage`	Low	Acknowledged
L-7	The missing check for `requiredConfirmation` when updating the `allowedBridgeAdapters` set	Low	Acknowledged
L-8	The refund address in `WormholeAdapter` can be different in different chains	Low	Acknowledged

L-9	`_disableBridgeAdapters()` and `_updateReceiverBridgeAdapters()` skip unfound bridge adapters	Low	Acknowledged
L-10	`_updateMessagesValidityTimestamp()` can set a validity timestamp for a wrong `chainId`	Low	Acknowledged
L-11	`CCIPAdapter.supportsInterface()` doesn't follow EIP-165	Low	Acknowledged

1.6 Conclusion

In this audit, we focused on assessing the system's robustness and reliability across various security and operational aspects, with a particular emphasis on accurately handling of edge cases that may arise during message transmission. Special attention was paid to the final execution of governance proposals in sidechains.

The following key points of the system, potential attack vectors, and points of failure were considered:

1. **Contract Proxies and Implementation Security**: Implementations of the contracts planned to be used from proxies (e.g., `CrossChainController`) cannot be exploited by malicious actors to steal users' funds or destroy the protocol. Initializing of implementations is impossible. However, a minor risk was identified where the contract owner could potentially destroy an implementation. A simple safeguard has been recommended to eliminate this problem.

2. **System Configuration and Parameter Management**: The system efficiently handles settings adjustments and parameter updates across chains. Our review confirmed that changes in one chain's parameters do not lead to protocol issues on other chains. Comprehensive checks are in place to prevent the setting of parameters that could render the protocol inoperative. Additional safeguards (like zero checks) have been suggested to further minimize the risk of operational errors.

3. **Message Handling Accuracy**: Messages are correctly and unambiguously encoded and decoded, effectively eliminating the risk of collisions. The protocol design prevents the execution of a proposal more than once, thus safeguarding against replay attacks. The use of the try/catch mechanism in `CrossChainReceiver` ensures proper handling of reverts, preventing malicious actors from exploiting the 63/64 gas attack.

4. **Cross-Chain Message Transmission**: It is a particularly sensitive part of the protocol, as it involves interaction with external bridge protocols. The system is resilient to the failure of part of the bridges. In case of a transmission failure, there is a possibility to resend the proposal. Adapters correctly handle the case when the same message from one of the bridges arrives several times. Almost all adapters in the scope correctly account for gas refunds, all adapters correctly calculate amounts of fees that should be used for message transferring. ChainIds are correctly converted for all the adapters. Our findings include a minor issue with the `WormholeAdapter` refund address, which, while not problematic in the current implementation, is recommended for correction to avoid future issues.

5. **Proposal Execution by CrossChainExecutor**: Received messages are queued and executed correctly with an execution delay and grace period applied. There is proper handling of calls and delegatecalls. Proposals related to changes in the parameters of the CrossChainExecutor itself are correctly executed. The `CrossChainExecutor` is also effectively safeguarded against reentrancy attacks. However, several issues have been noted. The first is related to the possibility of executing expired proposals when changing the grace period. The second is the lack of the ability to change the guardian in case of a compromise of the guardian. Optimal solutions have been proposed for these concerns.

6. **Storage Management**: All the adapters within the scope do not modify the storage of the controller. It is important because they are called via `delegatecall`. Whenever data is transferred from storage to memory for gas optimization, updates to storage are made in a timely and correct manner.

7. **Access Restrictions**: Functions across all stages of message transmission are properly restricted to prevent unauthorized use. Publicly accessible functions are designed to prevent disruption of system operations. `CrossChainReceiver.deliverEnvelope()` can only be called for an existing envelope for which the attempt at final delivery was unsuccessful and it's possible to make only one non-reverted retry. `BridgeExecutorBase.execute()` can be called only for an existing previously unexecuted `ActionSet` within proper time.

This project demonstrates a high degree of security and operational integrity. The audit has identified no critical or high vulnerabilities that would compromise the overall functionality or security of the system. With the adoption of the suggested improvements, the infrastructure will continue to provide robust cross-chain communication capabilities, vital for the seamless execution of governance proposals across the Lido ecosystem.

Besides conducting a security audit, the same team of auditors completed deployment verification. The following aspects were thoroughly checked:

1. The deployed contracts structure follows the designed and audited architecture.
2. Deployed contracts bytecode matches the audited contracts at the specific commit.
3. The selected design approach doesn't allow proxies initialization.
4. Proxy admins and owners are correctly configured to the trusted addresses.
5. Adapters inside the `CrossChainController` contract are correctly configured.
6. Adapters are initialized using the correct parameters.
7. `TrustedRemote`s on the receiving side are correctly set.

2. FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

M-1	Expired proposals can be executed
Severity	Medium
Status	Acknowledged

Description

An expired proposal can be executed in case of the `_gracePeriod` extension: BridgeExecutorBase.sol#L233.

Recommendation

We recommend saving `_gracePeriod` as a parameter of the action, so the parameter updates will not affect the protocol.

Client's commentary

> Acknowledged. The Lido governance token holders assume associated risks to verify the `_gracePeriod` input sanity during governance processes.

M-2	The guardian in `BridgeExecutorBase` can cancel an `ActionsSet` that changes the guardian
Severity	Medium
Status	Acknowledged

Description

The guardian in `BridgeExecutorBase` can cancel any `ActionsSet` immediately after an addition. They can also cancel an `ActionsSet` with a transaction to change the guardian. Thus, if the guardian's address is compromised, there is no way to change it, and redeployment of `CrossChainExecutor` will be required.
BridgeExecutorBase.sol#L112

Recommendation

We recommend accepting the rule that the guardian can be updated in the `actionSet` that contains only one action - guardian update. Such actions cannot be canceled by the guardian. In all other cases the guardian should be able to cancel the action.

Client's commentary

> Acknowledged. The risk is mitigated by setting the guardian address to ZERO_ADDRESS during the deployment. However, the DAO can opt-in to accept the risks involved by changing the address.

2.4 Low

L-1	The implementation can be destroyed by the `owner`
Severity	Low
Status	Acknowledged

Description

`owner` has rights to add any sender and bridge adapter on the implementation:
CrossChainController.sol#L12-L31 It can be used to call `selfdestruct` in some chains.
The implementation will be deployed by an EOA and the private key of the EOA can be stolen, or EOA can turns out a malicious actor.

Recommendation

We recommend transferring the ownership of the implementation to the `ZERO_ADDRESS`.

Client's commentary

> Acknowledged. During the a.DI deployment process, CrossChainController implementations on both networks will transfer ownership to the 0xDEAD address. Proxy owners will be transferred to the Lido DAO Agent and the CrossChainExecutor's addresses on Ethereum and BSC correspondingly.

L-2	Inconsistent zero checks
Severity	Low
Status	Acknowledged

Description

There are several places in the protocol where input parameters are not checked to be set:
`bridgeAdapterConfigInput.destinationChainId` CrossChainForwarder.sol#L332
`originConfig.originChainId` BaseAdapter.sol#L52
`FX_TUNNEL` PolygonAdapterBase.sol#L38
`REFUND_ADDRESS` WormholeAdapter.sol#L50
`owner` CrossChainController.sol#L14
`destination` CrossChainForwarder.sol#L119

Recommendation

We recommend adding zero checks for these parameters.

Client's commentary

> Acknowledged. The Lido governance token holders accept associated risks to verify the correctness of the values of destinationChainId, originChainId, FX_TUNNEL, REFUND_ADDRESS, and owner set upon initial a.DI deployment, and destination address upon future governance motions.

L-3	An unnecessary storage update
Severity	Low
Status	Acknowledged

Description

There is an unnecessary action in case `k == bridgeAdapterConfigs.length - 1`
CrossChainForwarder.sol#L407

Recommendation

We recommend adding a check that `k != bridgeAdapterConfigs.length - 1` and only in this case update the storage.

Client's commentary

> Acknowledged. Since the operation is straightforward and not repetitive, there is no need to complicate the code unnecessarily.

L-4	`_trustedRemotes` cannot be updated
Severity	Low
Status	Acknowledged

Description

It is impossible to update `_trustedRemotes` after the deployment BaseAdapter.sol#L52.

Recommendation

We recommend adding a function that allows to update `_trustedRemotes`.

Client's commentary

> Acknowledged. The current design assumes that adapters on each network can only communicate with the CrossChainControllers deployed during the same setup process. This is done to avoid any risks related to potential setup reconfiguration. In case of any further modification to the setup, new adapters may be deployed and a.DI setup can be re-configured, or a complete a.DI setup re-deployment will be conducted.

L-5	Unrestricted `forwardMessage` and `setupPayments`
Severity	Low
Status	Acknowledged

Description

The `forwardMessage` and `setupPayments` functions in all adapters can be called by anyone:
CCIPAdapter.sol#L67-L72
CCIPAdapter.sol#L122-L124
HyperLaneAdapter.sol#L46-L51
LayerZeroAdapter.sol#L75-L80
WormholeAdapter.sol#L54-L59
PolygonAdapterBase.sol#L42-L47

Recommendation

We recommend adding a check that these functions can be called only from the controller via `delegatecall`.

Client's commentary

> Acknowledged.
>
> - Unrestricted setupPayment is available to anyone to add funds to the adapter; there is no need to add restrictions here to easen the operations.
> - Unrestricted forwardMessage relies on the correct msg.sender transmission via the bridge and on msg.sender validation via trusted senders array on the target network adapter's side. This validation process helps filter out unrelated messages and ensures that only relevant messages are forwarded to the CrossChainReciever on the target network.

L-6	Unrestricted `sendMessage`
Severity	Low
Status	Acknowledged

Description

The `sendMessage` function in the Polygon adapter can be called by anyone:
FxTunnelEthereum.sol#L25-L29
FxTunnelPolygon.sol#L21-L24

Recommendation

We recommend restricting this function to only be called by the controller.

Client's commentary

> Acknowledged. The msg.sender will be validated on the target network adapter's side over _trustedRemotes array.

L-7	The missing check for `requiredConfirmation` when updating the `allowedBridgeAdapters` set
Severity	Low
Status	Acknowledged

Description

There is an issue in the function defined at line CrossChainReceiver.sol#L288. This function is used to update the `allowedBridgeAdapters` set. It is possible that some adapters could be removed, resulting in an insufficient number of remaining adapters to confirm a received envelope.

Recommendation

We recommend adding a check that after removing adapters from the `allowedBridgeAdapters` set, there are still enough adapters for the particular chain to pass the `requiredConfirmation` check.

Client's commentary

> Acknowledged. The Lido governance token holders accept associated risks to verify the input for the requiredConfirmation value correctness upon the a.DI deployment process and any potential future a.DI setup reconfigurations.

L-8	The refund address in `WormholeAdapter` can be different in different chains
Severity	Low
Status	Acknowledged

Description

`REFUND_ADDRESS` in `WormholeAdapter` is immutable. However, the address of the controller can be different in different chains. Using one `WormholeAdapter` for multiple chains can lead to the loss of gas refund.
WormholeAdapter.sol#L24
WormholeAdapter.sol#L69-L77

Recommendation

We recommend using different refund addresses for different chains.

Client's commentary

> Client: Not an issue. It's an intended design decision. One Wormhole adapter must be deployed for each path (remote network) because the refunding is done to the CrossChainController on the destination chain. (Deploy_Wormhole.s.sol#L17-L18)
> MixBytes: Considering that a unique Wormhole adapter will be deployed for each path, the issue will not affect the protocol in any way, but it is better to leave the finding in the report as a reminder for future development

L-9	`_disableBridgeAdapters()` and `_updateReceiverBridgeAdapters()` skip unfound bridge adapters
Severity	Low
Status	Acknowledged

Description

Unfound bridge adapters are just skipped if they were not found during removal in `CrossChainForwarder._disableBridgeAdapters()` and `CrossChainReceiver._updateReceiverBridgeAdapters()`. It makes it more difficult to identify a mistake in the modification of the allowed bridge adapters set.
CrossChainForwarder.sol#L391
CrossChainReceiver.sol#L288

Recommendation

We recommend reverting in case of an unfound bridge adapter.

Client's commentary

> Acknowledged. The Lido governance token holders accept associated risks to verify the bridgeAdapters values set upon the deployment and potential future a.DI reconfigurations.

L-10	`_updateMessagesValidityTimestamp()` can set a validity timestamp for a wrong `chainId`
Severity	Low
Status	Acknowledged

Description

There are no checks in `CrossChainReceiver._updateMessagesValidityTimestamp()` that a chain id is supported by the receiver. So, a validity timestamp can be set for any chain id. It makes more difficult to identify a mistake during the call of `CrossChainReceiver.updateMessagesValidityTimestamp()`.
CrossChainReceiver.sol#L246

Recommendation

We recommend adding a check to `CrossChainReceiver._updateMessagesValidityTimestamp()` that the `chainId` is in `_supportedChains`.

Client's commentary

> Acknowledged. The Lido governance token holders accept associated risks to verify the newValidityTimestampsInput values during the deployment and future a.DI reconfigurations.

L-11	`CCIPAdapter.supportsInterface()` doesn't follow EIP-165
Severity	Low
Status	Acknowledged

Description

CCIPAdapter implements IERC165, so it should follow EIP-165. EIP-165 requires the next:
https://eips.ethereum.org/EIPS/eip-165

```
Therefore, the implementing contract will
have a supportsInterface function that returns:

true when interfaceID is 0x01ffc9a7 (EIP165 interface)
false when interfaceID is 0xffffffff
true for any other interfaceID this contract implements
false for any other interfaceID
```

But `CCIPAdapter.supportsInterface()` returns false for `ICCIPAdapter`, which is implemented by `CCIPAdapter`.
CCIPAdapter.sol#L60

Recommendation

We recommend modifying `CCIPAdapter.supportsInterface()` to return true for the `ICCIPAdapter`'s identifier.

Client's commentary

> Acknowledged. The current codebase does not rely on this interface check, although adding the verification may be considered in the future.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes