LIDO EASYTRACK SMART CONTRACTS SECURITY AUDIT REPORT

0XORIO

OCTOBER 18, 2023

CONTENTS

1 INTRO

1.1 DISCLAIMER

The audit makes no assertions or warranties about the utility of the code, its security, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other statements about the fitness of the contracts for their intended purposes, or their bug-free status. The audit documentation is for discussion purposes only.

1.2 ABOUT OXORIO

Oxorio is a prominent audit and consulting firm in the blockchain industry, offering top-tier security audits and consulting to organizations worldwide. The company's expertise stems from its active involvement in designing and deploying multiple blockchain projects, wherein it developed and analyzed smart contracts.

With a team of more than six dedicated blockchain specialists, Oxorio maintains a strong commitment to excellence and client satisfaction. Its contributions to several blockchain projects reflect the company's innovation and influence in the industry. Oxorio's comprehensive approach and deep blockchain understanding make it a trusted partner for organizations in the sector.

Contact details:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

1.3 SECURITY ASSESSMENT METHODOLOGY

Several auditors work on this audit, each independently checking the provided source code according to the security assessment methodology described below:

1. Project architecture review

The source code is manually reviewed to find errors and bugs.

2. Code check against known vulnerabilities list

The code is verified against a constantly updated list of known vulnerabilities maintained by the company.

3. Security model architecture and structure check

The project documentation is reviewed and compared with the code, including examining the comments and other technical papers.

4. Cross-check of results by different auditors

The project is typically reviewed by more than two auditors. This is followed by a mutual cross-check process of the audit results.

5. Report consolidation

The audited report is consolidated from multiple auditors.

6. Re-audit of new editions

After the client has reviewed and fixed the issues, these are double-checked. The results are included in a new version of the audit.

7. Final audit report publication

The final audit version is provided to the client and also published on the company's official website.

1.4 FINDINGS CLASSIFICATION

1.4.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

- **CRITICAL**: A bug that could lead to asset theft, inaccessible locked funds, or any other fund loss due to unauthorized party transfers.
- **MAJOR**: A bug that could cause a contract failure, with recovery possible only through manual modification of the contract state or replacement.
- **WARNING**: A bug that could break the intended contract logic or expose it to DDoS attacks.
- **INFO**: A minor issue or recommendation reported to or acknowledged by the client's team.

1.4.2 Status Level Reference

Based on the client team's feedback regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

- **NEW**: Awaiting feedback from the project team.
- **FIXED**: The recommended fixes have been applied to the project code, and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Fixes for this finding are planned. This finding does not affect the overall security of the project.
- **NO ISSUE**: The finding does not affect the overall security of the project and does not violate its operational logic.

1.5 PROJECT OVERVIEW

Lido DAO governance currently relies on Aragon voting model. This means DAO approves or rejects proposals via direct governance token voting. Though transparent and reliable, it is not a convenient way to make decisions only affecting small groups of Lido DAO members. Besides, direct token voting doesn't exactly reflect all the decision making processes within the Lido DAO and is often used only to rubber-stamp an existing consensus. There are a few natural sub-governance groups within the DAO, e.g. validators committee, financial operations team and LEGO committee. Every day they need to take routine actions only related to their field of expertise. The decisions they make hardly ever spark any debate in the community, and votings on such decisions often struggle to attract wider DAO attention and thus, to pass.

Easy Track frictionless motions are solution to this problem. Easy Track motion is a lightweight voting considered to have passed if the minimum objections threshold hasn't been exceeded. As opposed to traditional Aragon votings, Easy Track motions are cheaper (no need to vote 'pro', token holders only have to vote 'contra' if they have objections) and easier to manage (no need to ask broad DAO community vote on proposals sparking no debate).

1.5.1 Documentation

For this audit, the following sources of truth about how the smart contracts should work were used:

- main GitHub repository of the project.
- documentation provided by the client

The sources were considered to be the specification. In the case of discrepancies with the actual code behavior, consultations were held directly with the client team.

1.6 AUDIT SCOPE

The scope of the audit includes the following smart contracts at:

- `AllowedTokensRegistry.sol`
- `TopUpAllowedRecipients.sol`
- `AllowedRecipientsBuilder.sol`
- `AllowedRecipientsFactory.sol`

The audited commit identifier is `425f4a254ceb2be389f669580b9dc76618e92756`

The bytecode of the deployed contracts
- `AllowedTokensRegistry`
- `TopUpAllowedRecipients`
- `TopUpAllowedRecipients`
- `TopUpAllowedRecipients`
- `AllowedRecipientsBuilder`
- `AllowedRecipientsFactory`

matches the final commit `52b1b1d99531a7aa46d8474bef56b157b83f318a` with accuracy up to the generated metadata.

2 FINDINGS REPORT

OXORIO

2.1 CRITICAL

No critical issues found.

2.2 MAJOR

No major issues found.

2.3 WARNING

W-01	Missing zero address validation for recepient leads to possible loss of fund in `TopUpAllowedRecipients`
Severity	WARNING
Status	• FIXED

Description

In the function `createEVMScript`, the function `_validateEVMScriptCallData` in the `TopUpAllowedRecipients` contract does not validate the recipients from the calldata array for zero address. While the funds can only be sent to allowed recipient addresses, which are validated through the `AllowedRecipientsRegistry` contract (out of scope), the `addRecipient` function of the contract allows adding a zero address as a recipient of the funds.

Recommendation

We recommend adding zero-address validation for the recipients' addresses. This will ensure that only valid addresses are allowed to receive funds, preventing potential issues or misuse of funds.

Update

Fixed in the commit `f4efe8e49aa23fe4f1f161386b750659100f39d7`.

W-02	Cross-token limit works correctly only with stablecoins in `TopUpAllowedRecipients`
Severity	WARNING
Status	• ACKNOWLEDGED

Description

In the function `_validateSpendableBalance` of the `TopUpAllowedRecipients` contract cross-token limit is checked. This limit applies to all allowed tokens and assumes that all tokens are stablecoins with no price difference. However, there is currently no on-chain check to ensure that only stablecoins are added as allowed tokens.

Recommendation

We recommend implementing an on-chain verification mechanism to validate that only stablecoins are added as allowed tokens. This will enhance the functionality of the contract and ensure that the cross-token limit is applied correctly.

Update

LIDO's response

We accept this operational risk and rely on verification before deployment.

2.4 INFO

I-01	Missing zero address validation in multiple contracts
Severity	INFO
Status	• ACKNOWLEDGED

Description

In the

- `constructor` of the `TopUpAllowedRecipients` contract
- `constructor` of the `AllowedRecipientsBuilder` contact
- `constructor` of the `AllowedTokensRegistry` contact,

in the functions

- `deployAllowedRecipientsRegistry`
- `deployAllowedTokensRegistry`
- `deployTopUpAllowedRecipients`
- `deployAddAllowedRecipient`
- `deployRemoveAllowedRecipient`

of the `AllowedRecipientsFactory` contract the parameters of the `address` type are not validated for zero address.

Recommendation

We recommend adding zero address validation for the mentioned address parameters. This will ensure that only valid addresses are accepted and prevent potential issues related to zero address usage.

Update

LIDO's response

These functions intended to be called from `AllowedRecipientsBuilder`. We accept operational risk and rely on verification before deployment directly from `AllowedRecipientsFactory`.

I-02	`Calldata` location can be used for function parameters in multiple contracts and interfaces
Severity	INFO
Status	• FIXED

Description

In the constructor of the `AllowedTokensRegistry` contract, in the functions

- `deployAllowedRecipientsRegistry`
- `deployAllowedTokensRegistry`

of the `IAllowedRecipientsFactory` interface,

- `deployAllowedRecipientsRegistry`
- `deployAllowedTokensRegistry`

of the `AllowedRecipientsFactory` contract,

- `deployAllowedRecipientsRegistry`
- `deployAllowedTokensRegistry`
- `deployFullSetup`
- `deploySingleRecipientTopUpOnlySetup`

of the `AllowedRecipientsBuilder` contract,

- `createEVMScript`,
- `decodeEVMScriptCallData`,
- `_validateEVMScriptCallData`,
- `_decodeEVMScriptCallData`

of the `TopUpAllowedRecipients` contract the function arguments are unnecessarily kept in memory, which can lead to inefficient gas usage.

Recommendation

We recommend changing the function arguments from `memory` to `calldata`, unless the code explicitly requires the argument to be in memory and modifies it. This will result in more efficient gas usage and improve the overall performance of the contract.

Update

LIDO's response

Fixed everything, except:
1) `constructor` in `AllowedTokensRegistry`. Data location must be `storage` or `memory` for constructor parameter in solidity 0.8.4. "Calldata is a non-modifiable, non-persistent area where function arguments are stored" (https://docs.soliditylang.org/en/v0.8.4/types.html#data-location)
2) `deployAllowedRecipientsRegistry` in `AllowedRecipientsBuilder`. This function is used from `deploySingleRecipientTopUpOnlySetup` with memory parameters what creates invalid implicit conversion.
3) `_validateEVMScriptCallData` in `TopUpAllowedRecipients`. This function is private and used only from `createEVMScript` with memory parameters.
Fixed in the commit `3b718ad094ed54da1d7e216c03b38b856dcac7a6`.

I-03	Floating pragma in the contracts of the scope
Severity	INFO
Status	• ACKNOWLEDGED

Description

All the contracts in the scope use the following pragma statement:

```
pragma solidity ^0.8.4;
```

This allows for the situation that the contracts get deployed with the compiler version different from the one they were tested with, which exposes the system to higher risks of undiscovered bugs.

Recommendation

We recommend locking the pragma statement to a specific compiler version intended by the developers. This will help ensure that the contracts are deployed with the intended compiler version and minimize the risks of undiscovered bugs.

I-04	Gas optimization of the `_recipients.length` in `TopUpAllowedRecipients`
Severity	INFO
Status	• ACKNOWLEDGED

Description

In the function `_validateEVMScriptCallData` of the `TopUpAllowedRecipients` contract, the length of the array from the function parameter `_recipients.length` is accessed three times. This can result in unnecessary gas consumption.

Recommendation

We recommend saving the value of the array length to a local variable. By doing so, the gas consumption can be reduced, as the value will only need to be accessed once.

I-05	Redundant initialization in `TopUpAllowedRecipients`
Severity	INFO
Status	• FIXED

Description

In the function `_validateEVMScriptCallData` of the `TopUpAllowedRecipients` contract the variable `totalAmount` is initialized with 0, which is redundant.

Recommendation

We recommend removing redundant initialization to keep the code base clean.

Update

Fixed in the commit `3b718ad094ed54da1d7e216c03b38b856dcac7a6`.

I-06	Redundant asserts in `AllowedRecipientsBuilder`
Severity	INFO
Status	• FIXED

Description

In the function `deployAllowedTokensRegistry` of the `AllowedRecipientsBuilder` contract the following assert statements are redundant:

```
assert(!registry.hasRole(REMOVE_TOKEN_FROM_ALLOWED_LIST_ROLE, address(this)));
assert(!registry.hasRole(DEFAULT_ADMIN_ROLE, address(this)));
```

This is because the `AllowedRecipientsBuilder` contract doesn't receive `DEFAULT_ADMIN_ROLE` and `REMOVE_TOKEN_FROM_ALLOWED_LIST_ROLE` roles in any previous steps.

Recommendation

We recommend removing the redundant assert statements to keep the codebase clean.

Update

Fixed in the commit `52b1b1d99531a7aa46d8474bef56b157b83f318a`.

I-07	Missing spec in `TopUpAllowedRecipients`
Severity	INFO
Status	• FIXED

Description

In the `constructor` of the `TopUpAllowedRecipients` contract the `_allowedTokensRegistry` parameter and its description are missing in the NatSpec comment of the constructor method.

Recommendation

We recommend adding the missing description of the variable to the NatSpec comment in the constructor method. This will improve the clarity and understanding of the code.

Update

Fixed in the commit `feb440dbe79031ca75d6956c08ce0c7d7f376ff7`.

3 CONCLUSION

The following table contains all the findings identified during the audit, grouped by statuses and severity levels:

Severity	FIXED	ACKNOWLEDGED	Total
CRITICAL	0	0	0
MAJOR	0	0	0
WARNING	1	1	2
INFO	4	3	7
Total	5	4	9

THANK YOU FOR CHOOSING

OXORIO