LIDO IN-PROTOCOL COVERAGE SMART CONTRACT AUDIT

February 28, 2022

MixBytes()

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 Lido Finance. 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 who check the provided source code independently of each other in accordance with the methodology described below:

01 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code
> Checking with static analyzers (i.e Slither, Mythril, etc.)

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the code for compliance with the desired security model:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> Cross-check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level and provide the client with an interim report.

05 Bug fixing & re-check:
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team 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 project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.3 PROJECT OVERVIEW

LIDO protocol is a project for stacking Ether to use it in Beacon chain. Users can deposit Ether to the Lido smart contract and receive stETH tokens in return. The stETH token balance corresponds to the amount of Beacon chain Ether that the holder could withdraw if state transitions were enabled right now in the Ethereum 2.0 network.
The Lido DAO is a Decentralized Autonomous Organization that manages the liquid staking protocol by deciding on key parameters (e.g., setting fees, assigning node operators and oracles, etc.) through the voting power of governance token (DPG) holders.
The Lido DAO is an Aragon organization. The protocol smart contracts extend AragonApp base contract and can be managed by the DAO.Currently, Lido has no adopted and well-defined mechanism of applying coverage for stakeholders' losses due to validators penalties, slashing and other conditions.
The researched smart contracts solve this problem. The contract enacts pending burning requests as a part of the next oracle report by burning all associated stETH tokens from its own balance.
The proposed contracts are non-upgradable and non-ownable.Contracts:

- `OrderedCallbacksArray` - is defining an ordered callbacks array supporting add/insert/remove ops.
- `CompositePostRebaseBeaconReceiver` - is defining a composite post-rebase beacon receiver for the Lido oracle.
- `SelfOwnedStETHBurner` - is dedicated contract for enacting stETH burning requests.

1.4 PROJECT DASHBOARD

Client	Lido Finance
Audit name	In-protocol Coverage
Initial version	ee1991b3bbea2a24b042b0a4433be04301992656
Final version	3d6a3f527e27a87e33c97726cce7de1ae7262d9f
Date	January 17, 2022 - February 28, 2022
Auditors engaged	4 auditors

FILES LISTING

CompositePostRebaseBeaconReceiver.sol	https://github.com/lidofinance/lido-dao/blob/ee1991b3bbea2a24b042b0a4433be04301992656/contracts/0.8.9/CompositePostRebaseBeaconReceiver.sol

OrderedCallbacksArray.sol	https://github.com/lidofinance/lido-dao/blob/ee1991b3bbea2a24b042b0a4433be04301992656/contracts/0.8.9/OrderedCallbacksArray.sol
SelfOwnedStETHBurner.sol	https://github.com/lidofinance/lido-dao/blob/ee1991b3bbea2a24b042b0a4433be04301992656/contracts/0.8.9/SelfOwnedStETHBurner.sol

FINDINGS SUMMARY

Level	Amount
Critical	1
Major	0
Warning	1
Comment	1

CONCLUSION

Smart contracts have been audited and several suspicious places have been found. The review found one critical issue, one warning, and one comment. After working on the reporting findings, all of them were confirmed and fixed by the client.

Final commit identifier with all fixes: `3d6a3f527e27a87e33c97726cce7de1ae7262d9f`

The following addresses contain deployed to the Ethereum mainnet and verified smart contracts code that matches audited scope:

- SelfOwnedStETHBurner.sol: 0x1e0C8542A59c286e73c30c45612d9C3a674A6cbC
- CompositePostRebaseBeaconReceiver.sol: 0xEdd972c22870726F30253efa88a08608F9748907
- OrderedCallbacksArray.sol: 0xEdd972c22870726F30253efa88a08608F9748907

2. FINDINGS REPORT

2.1 CRITICAL

CRT-1	Possibility of taking burned shares
File	SelfOwnedStETHBurner.sol
Severity	Critical
Status	Fixed at 3d6a3f52

DESCRIPTION

With attack it is possible to take burned shares profit even without taking shares before `processLidoOracleReport()` execution.
SelfOwnedStETHBurner.sol#L252
This exploit shows how the attack is done:
https://gist.github.com/georgiypetrov/22c0649058a97102e2fd97a1c619a3b3
If this is a front-run attack, then it will be the most convenient for the attacker.

RECOMMENDATION

It is necessary to add a limit on the amount of burned tokens.

CLIENT'S COMMENTARY

Recommendation implemented, commits:
PR-389
PR-389
PR-389
Updated spec: lip-6.md

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	There is no processing of the value returned by the function
File	SelfOwnedStETHBurner.sol
Severity	Warning
Status	Fixed at PR-389

DESCRIPTION

At the line
SelfOwnedStETHBurner.sol#L228
the `transfer()` function returns a boolean variable. But this variable is not processed in any way.
Similarly for the line:
SelfOwnedStETHBurner.sol#L203.

RECOMMENDATION

It is necessary to add processing of the values returned by the function.

CLIENT'S COMMENTARY

Implemented with the following commit:
PR-389

2.4 COMMENT

CMT-1	Extra operation
File	SelfOwnedStETHBurner.sol
Severity	Comment
Status	Fixed at PR-389

DESCRIPTION

At the line
SelfOwnedStETHBurner.sol#L223
checks the value of the `_token` variable. But if the value of the variable is zero, then the code on line 228 will not be executed.
Similarly for the line:
SelfOwnedStETHBurner.sol#L239

RECOMMENDATION

It is necessary to remove redundant operations.

CLIENT'S COMMENTARY

Implemented with the following commit:
PR-389

3. ABOUT MIXBYTES

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

BLOCKCHAINS

- Ethereum
- Cosmos
- EOS
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes