LIDO FINANCE STETH PRICE FEED SMART CONTRACT AUDIT

June 10, 2021

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 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.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 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 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> 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

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

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> 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

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> 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
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.

1.4 EXECUTIVE SUMMARY

The LIDO stETH-price-feed project allows users and smart contracts to access the aggregated stETH price. The price is aggregated from two sources: the first source is the curve.fi stETH/ETH pool, and the second source is the LIDO stETH price oracle, which uses block hash parsing to get the stETH price from older blocks, and thus provides protection against attack through a flashloan. Additionally this implementation uses safe price range checks where the amplitude of the price change considered "safe" by the price feed, that potentially adds a layer of security for the price feed data customers.

1.5 PROJECT DASHBOARD

Client	Lido Finance
Audit name	stETH Price Feed
Initial version	459495f07c97d04f6e3839e7a3b32acfcade22ad 4a5db9ad4b0c8d815388d087a023f2b390af351a
Final version	4a5db9ad4b0c8d815388d087a023f2b390af351a
SLOC	276
Date	2021-05-20 - 2021-06-10
Auditors engaged	2 auditors

FILES LISTING

PriceFeedProxy.sol	PriceFeedProxy.sol
StEthPriceFeed.vy	StEthPriceFeed.vy

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	4
Comment	6

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical or major issues were found, several warnings and comments were spotted. After working on the reported findings all of them were fixed by the client or acknowledged (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `4a5db9ad4b0c8d815388d087a023f2b390af351a`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Possible assets loss
File	PriceFeedProxy.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

In the current version of protocol, user can lose money if accidentally sends tx to proxy with `msg.value>0`:
PriceFeedProxy.sol

RECOMMENDATION

We know that in the future the smart contract under proxy possibly would have some logic with receiving/withdrawing assets but as now it is not supported we recommend to override `_beforeFallback` like this:

```
function _beforeFallback() internal override {
    if (msg.value > 0) {
        require(IsReceivingAllowed, "Sending wei not allowed");
    }
}
```

CLIENT'S COMMENTARY

While the price oracle is permissionless, we expect it to be mostly called by "oracle operators" & pre-made oracle daemon apps. That means we don't expect any

funds sent to the contract. In case that would happen, we could upgrade the implementation to add "return funds" methods for ETH and ERC20 tokens.

WRN-2	Possible admin control loss
File	PriceFeedProxy.sol StEthPriceFeed.vy
Severity	Warning
Status	No issue

DESCRIPTION

In the current version of protocol, admin can set address of a new admin to zero, which means that nobody can call admin functions after that:
PriceFeedProxy.sol#L106
StEthPriceFeed.vy#L151

RECOMMENDATION

We recommend to add simple check:

```
require(newAdmin != 0, "Incorrect admin address");
```

CLIENT'S COMMENTARY

That's intended behavior, as we may want to have "immutable" price feed with admin set to the zero address. Check the test here: test_initialization.py#L50

WRN-3	Incorrect input parameters
File	StEthPriceFeed.vy
Severity	Warning
Status	No issue

DESCRIPTION

In the function for calculating price, relative difference `old` argument can be zero:
StEthPriceFeed.vy#L58

RECOMMENDATION

We recommend to add simple check:

```
assert old > 0, "oracle price not init"
```

CLIENT'S COMMENTARY

Price in the Curve pool - thus in the Oracle - can't happen to be zero in production environment.

WRN-4	Bad range for price change
File	StEthPriceFeed.vy
Severity	Warning
Status	Fixed at 4a5db9ad

DESCRIPTION

In the current version of feed smart contract, it is possible to set 100% (for assets with the same price it is very big range) allowed range for price change: StEthPriceFeed.vy#L162

RECOMMENDATION

We recommend to change max allowed difference to `1000`.

2.4 COMMENTS

CMT-1	Unnecessary payable modificator
File	PriceFeedProxy.sol
Severity	Comment
Status	No issue

DESCRIPTION

In the current version proxy constructor can receive assets, but does not use it anywhere:
PriceFeedProxy.sol#L57

RECOMMENDATION

We recommend to remove `payable` modificator.

CLIENT'S COMMENTARY

We've aimed to follow OpenZeppelin's ERC1967Proxy as close as possible here, leaving the modificator as-is.

CMT-2	Unnecessary check in constructor
File	PriceFeedProxy.sol
Severity	Comment
Status	No issue

DESCRIPTION

In the current version, proxy constructor contains unnecessary check:
PriceFeedProxy.sol#L69

RECOMMENDATION

We recommend to remove this check.

CLIENT'S COMMENTARY

That's a sanity check for the correctness of the slot initialization, which we plan to leave as-is

CMT-3	Possible incorrect initialization
File	StEthPriceFeed.vy
Severity	Comment
Status	No issue

DESCRIPTION

In the current version of feed contract, `initialize` function can be invoked only once and it can be done by anybody:
StEthPriceFeed.vy#L31

RECOMMENDATION

We recommend to deploy feed and proxy contracts in adjacent tx.

CLIENT'S COMMENTARY

Current deployment script deploys contracts one after another: deploy.py#L25-L26

CMT-4	Events not emitting
File	StEthPriceFeed.vy
Severity	Comment
Status	Fixed at 4a5db9ad

DESCRIPTION

In the current version of feed contract after change of some storage variables events not emitting:
StEthPriceFeed.vy#L117
StEthPriceFeed.vy#L144
StEthPriceFeed.vy#L155

RECOMMENDATION

We recommend to add events and emit them in functions `update_safe_price`, `set_admin` and `set_max_safe_price_difference`.

CLIENT'S COMMENTARY

`update_safe_price` creates an event; will update the other two methods to emit as well

CMT-5	No checks timestamp in `_update_safe_price`
File	StEthPriceFeed.vy
Severity	Comment
Status	No issue

DESCRIPTION

Currently `_update_safe_price` method (StEthPriceFeed.vy#L100) ignores information about `self.safe_price_timestamp`. So the intruder may manipulate with `safe_price_value` in one transaction.

RECOMMENDATION

We recommend to add some minimum diff for `safe_price_timestamp`.

CLIENT'S COMMENTARY

While it's possible to tamper with the pool state & safe price in single tx, the security guarantees price feed provides aren't breached there, as the safe price stays in the defined range.

CMT-6	Storage collisions between implementation versions
File	StEthPriceFeed.vy
Severity	Comment
Status	Fixed at 4a5db9ad

DESCRIPTION

In future, if a developer changes order or types of variables in a new logic contract, it may be broken. More information about Storage Collisions you can find https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies#unstructured-storage-proxie.

RECOMMENDATION

We recommend to add comment in the code of a logic contract about this collisions.

CLIENT'S COMMENTARY

Will add the comment about the possible collisions.

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