LIDO DEPOSIT SECURITY MODULE 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.On Tuesday, Oct 5, the vulnerability allowing the malicious Node Operator to intercept the user funds on deposits to the Beacon chain in the Lido protocol was reported to Immunefi. The vulnerability could only be exploited by the Node Operator front-running the Lido.depositBufferedEther transaction with direct deposit to the DepositContract of no less than 1 ETH with the same validator public key & withdrawal credentials different from the Lido's ones, effectively getting control over 32 ETH from Lido.This scope contains contracts that are involved in mitigation after a vulnerability is found.
Lido propose to establish the Deposit Security Committee dedicated to ensuring the safety of deposits on the Beacon chain:

- monitoring the history of deposits and the set of Lido keys available for the deposit, signing and disseminating messages allowing deposits;
- signing the special message allowing anyone to pause deposits once the malicious Node Operator pre-deposits are detected.

Each member must generate an EOA address to sign messages with their private key. The addresses of the committee members will be added to the smart contract.
Main interacting contracts:
`Lido` - The contract contains basic logic for accepting deposits from Ethereum 1.0 and transferring them to Beacon chain.
`NodeOperatorsRegistry` - This contract stores the data on deposits of all validators that will be registered by the protocol. Each deposit data record contains, among other things, the public key of the validator.
`DepositSecurityModule` - This is a new contract created for committee members and the safe call of the `depositBufferedEther()` function in Lido's contact.
`deposit_contract` - This contract is required for Beacon chain deposits. Made according to the specification https://github.com/ethereum/eth2.0-specs. It hasn't been changed.
`ECDSA` - A library for getting an address from encoded data. Imported from OpenZeppelin.
`MemUtils`, `Pausable`, `BytesLib` - These contracts contain various helper libraries.

1.4 PROJECT DASHBOARD

Client	Lido Finance
Audit name	Deposit Security Module
Initial version	5b449b740cddfbef5c107505677e6a576e2c2b69

Final version	816bf1d0995ba5cfdfc264de4acda34a7fe93eba
Date	October 15, 2021 - February 28, 2022
Auditors engaged	4 auditors

FILES LISTING

Lido.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.4.24/Lido.sol
INodeOperatorsRegistry.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.4.24/interfaces/INodeOperatorsRegistry.sol
MemUtils.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.4.24/lib/MemUtils.sol
Pausable.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.4.24/lib/Pausable.sol
NodeOperatorsRegistry.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.4.24/nos/NodeOperatorsRegistry.sol
deposit_contract.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.6.11/deposit_contract.sol
DepositSecurityModule.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.8.9/DepositSecurityModule.sol
ECDSA.sol	https://github.com/lidofinance/lido-dao/blob/5b449b740cddfbef5c107505677e6a576e2c2b69/contracts/0.8.9/lib/ECDSA.sol
BytesLib.sol	https://github.com/GNSPS/solidity-bytes-utils/blob/5d251ad816e804d55ac39fa146b4622f55708579/contracts/BytesLib.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	2
Warning	13
Comment	7

CONCLUSION

Smart contracts have been audited and several suspicious places have been detected. During the audit no critical issues were found, two majors, several warnings and comments were spotted. After working on the reported findings all of them were acknowledged or fixed by the client.

Final commit identifier with all fixes: `816bf1d0995ba5cfdfc264de4acda34a7fe93eba`

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

- Lido.sol : 0xc7b5af82b05eb3b64f12241b04b2cf14469e39f7
- NodeOperatorsRegistry.sol: 0xec3567ae258639a0ff5a02f7eaf4e4ae4416c5fe
- deposit_contract.sol: 0x00000000219ab540356cbb839cbe05303d7705fa
- DepositSecurityModule.sol: 0xDb149235B6F40dC08810AA69869783Be101790e7

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Possible blocking of the contract
File	DepositSecurityModule.sol
Severity	Major
Status	**Fixed** at **dd992553**

DESCRIPTION

At the line
DepositSecurityModule.sol#L123
initializes the `owner` variable without checking the new value of the variable.
If the value of the variable is equal to zero, then the following functions will stop working:
`setOwner()`, `setNodeOperatorsRegistry()`, `setPauseIntentValidityPeriodBlocks()`, `setMaxDeposits()`, `setMinDepositBlockDistance()`, `setGuardianQuorum()`, `addGuardian()`, `addGuardians()`, `removeGuardian()`, `unpauseDeposits()`.

RECOMMENDATION

It is necessary to add a check for the value of the `newValue` variable to zero before initializing the `owner` variable.

CLIENT'S COMMENTARY

Resolved.
Sanity check for zero address was added in commit dd992553

MJR-2	Fix gas cost ETH transfer
File	Lido.sol
Severity	Major
Status	**Fixed** at **cb13efd1**

DESCRIPTION

ETH sending at this line
Lido.sol#L301
uses 2300 gas amount, but gas price can be vary and it is possible that in future it may exceed gas limit.

RECOMMENDATION

We recommend sending ETH via call.

CLIENT'S COMMENTARY

Resolved.
Transfer was replaced with call in commit cb13efd1

2.3 WARNING

WRN-1	It is required to redo the logic of working with quorum
File	DepositSecurityModule.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the following lines there is the value of the `quorum` variable changed:

- DepositSecurityModule.sol#L265 when adding one guardian
- DepositSecurityModule.sol#L278 when adding multiple guardians
- DepositSecurityModule.sol#L265 when removing one guardian

There may even be such a situation that the next such action will set the quorum value equal to `1`.
In this case, it is possible for the `depositBufferedEther()` function to work with the signature of only one guardian.
This disagrees with the documentation lip-5.md:
"To make a deposit, we propose to collect a quorum of 2/3 of the signatures of the committee members".
Currently, the value of the `quorum` variable is not associated with the value of the `guardians` variable.
We recommend doing the following:

- remove the ability to change the variable that affects the quorum when changing the number of guardians
- instead of the `quorum` variable, use the percentage of the number of elements in the `guardians` array
- add a lower bound for the percentage of quorum members and the minimum number of guardians

RECOMMENDATION

It is necessary to redo the logic of work with quorum counting.

CLIENT'S COMMENTARY

Acknowledged.
LIP 5 contains a proposal for setting the quorum value to 2/3, but it's not constant value and could be changed with DAO voting. All methods that could change quorum value are guarded by DAO voting.

WRN-2	Gas overflow during iteration (DoS)
File	NodeOperatorsRegistry.sol Lido.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Each iteration of the cycle requires a gas flow.
A moment may come when more gas is required than it is allocated to record one block.
In this case, all iterations of the loop will fail.
Affected lines:

- NodeOperatorsRegistry.sol#L214-L221
- Lido.sol#L616-L624

RECOMMENDATION

It is recommended to limit the number of loop iterations.

CLIENT'S COMMENTARY

Acknowledged.
Each iteration of the loop costs about 30k gas, so this method will fit in the gas limits of the block, even for a couple of hundreds of node operators. Such buffer of node operators amount will do for Lido for long terms.

WRN-3	Unused keys trimming doesn't remove keys and signatures from storage
File	NodeOperatorsRegistry.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

This function
NodeOperatorsRegistry.sol#L212
leave keys data in storage.
It may be dangerous.

RECOMMENDATION

We recommend to remove properly.

CLIENT'S COMMENTARY

Acknowledged.
`trimUnusedKeys()` sets the border of storage behind the last used key, so the new keys would rewrite the old ones, and reading not be performed for removed keys. Removing keys from storage is not necessary with the current approach. The actual removal of signing keys will require introducing an additional argument for the method, with the max amount of keys to remove, to fit within the gas limit of a block. It will complicate the process of removing all unused signing keys and will require multiple calls of this method to remove unused keys completely.

WRN-4	Possible blocking of work with buffered ETH
File	NodeOperatorsRegistry.sol DepositSecurityModule.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines
NodeOperatorsRegistry.sol#L252-L262 there's a function `addSigningKeysOperatorBH()`.
This function can be called from any address of operators `[_operator_id].rewardAddress`.
Calling this function increments the value of `KEYS_OP_INDEX_POSITION`. And the current value of `KEYS_OP_INDEX_POSITION` is used here:
DepositSecurityModule.sol#L413-L414.
Thus, any operator (and inactive too) can block the operation of `depositBufferedEther()`.
However, it is safe to do the same with the `addSigningKeys()` function, which can only be called by special users.

RECOMMENDATION

It is necessary to add a restriction on the call to the `addSigningKeysOperatorBH()` function or remove it from the source code altogether.

CLIENT'S COMMENTARY

Acknowledged.
Attack via `addSigningKeysOperatorBH()` will be really expensive for the malicious node operator, cause one transaction with a call of this method costs about 250k gas. In addition, we can change the reward address of the malicious node operator via the method `setNodeOperatorRewardAddress()` to make it impossible to call this method for the malicious node operator.

WRN-5	Superfluous actions when the function is called again
File	DepositSecurityModule.sol
Severity	Warning
Status	Fixed at ac96eb0a

DESCRIPTION

At the lines
DepositSecurityModule.sol#L336-L356 there's a call to the `pauseDeposits()` function to set the value of the `paused` variable to `true`.
Now this function will be executed if the value of the `paused` variable is already set to `true`.
Such actions will lead to unnecessary gas consumption.

RECOMMENDATION

It is necessary to roll back the transaction if the value of the `paused` variable is already set to `true`.

CLIENT'S COMMENTARY

Fixed in `ac96eb0a9c0c9b8f30ad3b49cf315ec4e9f8c6af`. Early return was added instead of reverting the transaction: otherwise, for consistency, all other mutating methods (e.g. setMaxDeposits) would need to be changed to use revert instead of doing nothing.

Also added a clarifying comment in the code. In the fix an attempt to pause already paused deposits is implemented as "no-op" instead of "revert" semantics as it was proposed.
In case of an emergency function pauseDeposits is supposed to be called by all guardians. Thus only the first call will do the actual change. But the other calls would be OK operations from the point of view of protocol's logic. Thus we have preferred not to use "error" semantics which is implied by require.

PR-393. Final commit containing all changes `bcae4249a0d9cf1e92858f92cf9fe5154dff8c21`.

WRN-6	Unnecessary actions
File	NodeOperatorsRegistry.sol
Severity	Warning
Status	Fixed at **b9bbc487**

DESCRIPTION

At the lines
NodeOperatorsRegistry.sol#L141-L157 there's a call to the `setNodeOperatorActive()` function to set the value of the `operators[_id].active` variable to `_active`.
However, if the value of the variable does not change, you do not need to make a call to the `_increaseKeysOpIndex()` function and record the `NodeOperatorActiveSet` event.
Before line 145, add the following check:

```
    require(operators[_id].active != _active, "SAME_VALUE");
```

And it will be necessary to remove the use of the `if` operator on line 146.

At the line
NodeOperatorsRegistry.sol#L166 you need to add a check like this:

```
    require(operators[_id].name != _name, "SAME_VALUE");
```

At the line
NodeOperatorsRegistry.sol#L178 you need to add a check like this:

```
    require(operators[_id].rewardAddress != _rewardAddress, "SAME_VALUE");
```

At the line
NodeOperatorsRegistry.sol#L189 you need to add a check like this:

```
    require(operators[_id].stakingLimit != _stakingLimit, "SAME_VALUE");
```

RECOMMENDATION

The source code needs to be changed.

CLIENT'S COMMENTARY

Fixed in b9bbc48702cf24475c3f1e8cb8259e7175e86783.

WRN-7	Guardian zero address check
File	DepositSecurityModule.sol
Severity	Warning
Status	Fixed at **071851dd**

DESCRIPTION

It is possible to add guardian with zero address
DepositSecurityModule.sol#L281.

RECOMMENDATION

It is recommended to add zero address check

```
require(addr != address(0), "guardian zero address");
```

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes bcae4249a0d9cf1e92858f92cf9fe5154dff8c21

WRN-8	Variable without visibility modifier
File	DepositSecurityModule.sol
Severity	Warning
Status	Fixed at ddb2420a

DESCRIPTION

Variable DepositSecurityModule.sol#L61. It is not clear which visibility modifier is default.

RECOMMENDATION

It is recommended to set visibility to `internal`

```
    mapping(address => uint256) internal guardianIndicesOneBased; // 1-based
```

CLIENT'S COMMENTARY

Fixed in ddb2420a1131ad57eaa73c4fb7476c09625707e5

WRN-9	Hardcoded `MAX_BPS`
File	Lido.sol
Severity	Warning
Status	**Fixed** at **6dea01fd**

DESCRIPTION

`MAX_BPS` is hardcoded here:
Lido.sol#L189,
Lido.sol#L582,
Lido.sol#L593,
Lido.sol#L599,
Lido.sol#L654,
Lido.sol#L681.
It is not safe if it may be changed.

RECOMMENDATION

It is recommended to store `MAX_BPS` as variable.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes bcae4249a0d9cf1e92858f92cf9fe5154dff8c21

WRN-10	Possible `uint256` overflow
File	NodeOperatorsRegistry.sol
Severity	Warning
Status	**Fixed** at **0b1e52e1**

DESCRIPTION

There may be `uint256` overflow

- NodeOperatorsRegistry.sol#L287,
- NodeOperatorsRegistry.sol#L311.

RECOMMENDATION

It is recommended to use SafeMath.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes bcae4249a0d9cf1e92858f92cf9fe5154dff8c21

WRN-11	Unnecessary staking limit decrease
File	NodeOperatorsRegistry.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Is not clear why staking limit should be decreased to index
NodeOperatorsRegistry.sol#L627
If we remove key at index 0 then we will have all keys of this node operator to be used.

RECOMMENDATION

It is recommended to not decrease staking limit.

CLIENT'S COMMENTARY

We replace the key to be deleted with the latter. And we cut it off just in case, because if we delete the keys, then either the protocol has some serious problem, or this is a large planned operation, and it should include raising the limit.
In theory, it could just be reduced by 1, but the code would be more complicated, and there is little difference in practice.

WRN-12	Using unchecked address
File	DepositSecurityModule.sol
Severity	Warning
Status	Fixed at bf3c2611

DESCRIPTION

At the lines:
DepositSecurityModule.sol#L77
DepositSecurityModule.sol#L78
address is not checked before using.

RECOMMENDATION

It is recommended to check address for zeros.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes bcae4249a0d9cf1e92858f92cf9fe5154dff8c21

WRN-13	An event is omitted
File	DepositSecurityModule.sol Lido.sol
Severity	Warning
Status	**Fixed** at **0ce2dc2c**

DESCRIPTION

There are omitted events at the lines:

- DepositSecurityModule.sol#L425
- Lido.sol#L283
- Lido.sol#L406
- Lido.sol#L415
- Lido.sol#L424
- Lido.sol#L429
- Lido.sol#L434
- Lido.sol#L546
- Lido.sol#L656

RECOMMENDATION

It is recommended to add events.

CLIENT'S COMMENTARY

The first event (lastDepositBlock change) is not needed since Lido already emits an event on deposit and the block can be trivially obtained from that event. However, an accessor was added for lastDepositBlock in 7bdd51eeebaf0617be9b5de8dd2789a561a1a04d to provide an easy way to obtain this value onchain.
Further the 9 issues within WRN-13 are numbered in order.

2.4 COMMENT

CMT-1	Can use the constant
File	NodeOperatorsRegistry.sol
Severity	Comment
Status	**Fixed** at **3b114262**

DESCRIPTION

At the line
NodeOperatorsRegistry.sol#L42 there's a constant `UINT64_MAX` defined.
At the line
NodeOperatorsRegistry.sol#L554 can use this constant.

RECOMMENDATION

It is recommended to refactor your source code.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes 3b11426286aa3605cb26acca1cb6e26e0f7e8fa6

CMT-2	Extra code
File	NodeOperatorsRegistry.sol
Severity	Comment
Status	**Fixed** at **a6d264ae**

DESCRIPTION

At the lines in regular methods:

- NodeOperatorsRegistry.sol#L541
- NodeOperatorsRegistry.sol#L566
- NodeOperatorsRegistry.sol#L646
  checks are made for the constant `PUBKEY_LENGTH`.

At the lines in regular methods:

- NodeOperatorsRegistry.sol#L567
- NodeOperatorsRegistry.sol#L647
- NodeOperatorsRegistry.sol#L646
  checks are made for the constant `SIGNATURE_LENGTH`.

It is correct to do these checks before initializing constants.

RECOMMENDATION

It is recommended to remove the extra code. This will save gas.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes a6d264aef7e2ee22441ec3ef0afb378babecfbf7

CMT-3	Extra initialization
File	DepositSecurityModule.sol
Severity	Comment
Status	**Fixed** at **308b8999**

DESCRIPTION

At the lines in the constructor
DepositSecurityModule.sol#L98-L99
initialization of variables is done.
But initially the variables already have these values.

RECOMMENDATION

It is recommended to remove the extra code.

CLIENT'S COMMENTARY

It's done purely for clarity to make reading the code easier.
Have reconsidered and fixed.
PR-391
Final commit containing all changes 308b8999d51467c5e146e19f62872ff466815f0e

CMT-4	Comment typo error
File	DepositSecurityModule.sol
Severity	Comment
Status	**Fixed** at **f9ab47ee**

DESCRIPTION

This comment is probably incorrect
DepositSecurityModule.sol#L390.

RECOMMENDATION

It is recommended to check for non-equal

```
     *   6. blockhash(blockNumber) != blockHash
```

CLIENT'S COMMENTARY

Fixed in f9ab47ee5dbade5d2aac98961df340e56c290ed2.

CMT-5	Storage assigning may be in if statement
File	NodeOperatorsRegistry.sol
Severity	Comment
Status	**Fixed** at **b9bbc487**

DESCRIPTION

This assigning can be placed in if statement above NodeOperatorsRegistry.sol#L154.

RECOMMENDATION

It is recommended to place in if statement above.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes b9bbc48702cf24475c3f1e8cb8259e7175e86783

CMT-6	Naming variable mistake
File	NodeOperatorsRegistry.sol
Severity	Comment
Status	**Fixed** at **bcae4249**

DESCRIPTION

Because of `_totalRewardShares` not divides for validators but divides for `effectiveStakeTotal` should to rename `perValidatorReward` to `perStakeReward` at `NodeOperatorsRegistry.sol` at line:
NodeOperatorsRegistry.sol#L441

RECOMMENDATION

It is recommended to rename `perValidatorReward` to `perStakeReward`.

CLIENT'S COMMENTARY

Fixed. Naming of neighbouring variables was incorrect. We've replaced effectiveStake with activeValidators thus making naming of perValidatorReward correct.
The fix will be delivered in the next protocol upgrade.
PR-393
Final commit containing all changes bcae4249a0d9cf1e92858f92cf9fe5154dff8c21

CMT-7	Omitted description
File	Lido.sol
Severity	Comment
Status	**Fixed** at **d17b86d1**

DESCRIPTION

There are no description at `Lido.sol` at `_setTreasury()` function at line:
Lido.sol#L427
At `_setInsuranceFund()` at line:
Lido.sol#L432
At `_distributeNodeOperatorsReward()` at line:
Lido.sol#L611

RECOMMENDATION

It is recommended to add description.

CLIENT'S COMMENTARY

Fixed PR-393
Final commit containing all changes d17b86d118206687bcf2a701ebc5a9f2ee85121e

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