MixBytes() Camp

Lido V2 Contest Report

1. Introduction

1.1 Disclaimer

The contest audit report and MixBytes make no statements or warranties about the utility of the code, the 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 the fitness of the contracts to purpose, or their bug-free status. The contest documentation is for discussion and information purposes only.

Mixbytes services offered are limited to contest hosting and contest conduct assistance on the MixBytes Camp platform and within the MixBytes Camp community. Mixbytes does not provide audit services within this contest. MixBytes Camp community members are not employees, partners, agents, contractors, or subcontractors of Mixbytes. MixBytes in no way shall be held liable or responsible for MixBytes Camp community members. MixBytes in no way controls or distributes the bounty fund. Mixbytes does not conduct any KYC verification on its platform.

1.2 Objective

The audit contest takes place in order to identify possible security vulnerabilities and flaws by the independent security researchers who are team members of the MixBytes Camp community.

The result of the contest is a unified report containing a list of all triaged vulnerabilities found during the contest, ranked according to the approved severity level.

Within the scope of the audit contest, there is no stage of re-audit after bug fixing on the Project's side.

Finding Severity breakdown

All vulnerabilities discovered during the audit contest are classified based on their potential severity and ranked in the following way:

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.

1.3 Project Overview

The Lido Ethereum Liquid Staking Protocol built on Ethereum allowes their users to earn staking rewards on the Beacon chain without locking Ether or maintaining staking infrastructure.

The second version of the protocol (Lido V2) has a withdrawal feature. This is a complex task due to the difficulty of synchronizing access to data between the Consensus and Execution Layers, as well as the async nature of the Consensus Layer validator exit and slashing mechanics.

Lido V2 handles withdrawal requests in an asynchronous manner using a FIFO queue. The mechanics are hugely affected by the asynchronous nature of the Ethereum withdrawal operations.

Lido V2 enables various staking modules to be introduced with the Staking Route.

The Staking Router contract is responsible for appropriately balancing stake distribution, accrued rewards and operation of the plugged-in staking modules.

1.4 Project Dashboard

Project Summary

Title	Description
Sponsor	Lido
Project name	Lido contest
Timeline	February 14 2023 - March 21 2023

Project Log

Date	Commit Hash	Note
21.02.2023	e57517730c3e11a41e9cbc32ce018726722335b7	Commit for the contest
07.04.2023	29efd7f69df2ef983ca07c43e7e855d05c3ca8c3	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	57b675212a68db308e690e6daacc37ae99ca2f99	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	0a9dbcbb1c035fff0c97128aee3a0f8530ece586	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	7e2a32f8b60e400a091d7a119e7095bb04a6e14b	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	ddf756779a07076460272a24b4491ef5542cd5e5	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	da4ac3420f8dc7cd4ada08560926d0b183405751	Commit with fixes of findings discovered by MixBytes Camp

07.04.2023	25a53c093f530e6315a87d1b5c95911ee6d081fe	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	59b0476fef2c89a09e3b7cca264923eb6f9635b9	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	2032f0dc57521437457a934daebe16d98e40b2bc	Commit with fixes of findings discovered by MixBytes Camp
07.04.2023	d1facbd5f3f39cfffe31201f2142f1eea9e1e1ce	Commit with fixes of findings discovered by MixBytes Camp

Project Scope

The audit covered the following files:

File name	Link
Packed64x4.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/lib/Packed64x4.sol
SigningKeys.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/lib/SigningKeys.sol
StakeLimitUtils.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/lib/StakeLimitUtils.sol
NodeOperatorsRegistry.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol
LegacyOracle.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/oracle/LegacyOracle.sol
Pausable.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/utils/Pausable.sol
Versioned.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/utils/Versioned.sol
Lido.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/Lido.sol
StETH.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/StETH.sol
StETHPermit.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/StETHPermit.sol
WstETH.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.6.12/WstETH.sol
Math.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/lib/Math.sol
PositiveTokenRebaseLimiter.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/lib/PositiveTokenRebaseLimiter.sol
UnstructuredStorage.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/lib/UnstructuredStorage.sol
UnstructuredRefStorage.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/lib/UnstructuredRefStorage.sol
AccountingOracle.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/AccountingOracle.sol
BaseOracle.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/BaseOracle.sol
HashConsensus.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/HashConsensus.sol
ValidatorsExitBusOracle.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/ValidatorsExitBusOracle.sol
OssifiableProxy.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/proxy/OssifiableProxy.sol

OracleReportSanityChecker.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/sanity_checks/OracleReportSanityChecker.sol
AccessControl.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/access/AccessControl.sol
AccessControlEnumerable.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/access/AccessControlEnumerable.sol
PausableUntil.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/PausableUntil.sol
Versioned.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/Versioned.sol
BeaconChainDepositor.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/BeaconChainDepositor.sol
Burner.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/Burner.sol
DepositSecurityModule.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol
EIP712StETH.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/EIP712StETH.sol
LidoExecutionLayerRewardsVault.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/LidoExecutionLayerRewardsVault.sol
LidoLocator.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/LidoLocator.sol
OracleDaemonConfig.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/OracleDaemonConfig.sol
StakingRouter.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/StakingRouter.sol
WithdrawalQueueERC721.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueueERC721.sol
WithdrawalQueue.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueue.sol
WithdrawalQueueBase.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueueBase.sol
WithdrawalVault.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalVault.sol
ECDSA.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/common/lib/ECDSA.sol
Math256.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/common/lib/Math256.sol
MemUtils.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/common/lib/MemUtils.sol
MinFirstAllocationStrategy.sol	https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/common/lib/MinFirstAllocationStrategy.sol

1.5 Summary of findings

Severity	№ of Findings
Critical	0
High	1
Medium	3

Low	13

2. Findings Report

2.1 Critical

Not found

2.2 High

1. Another user can use the message signed by the guardian again

Description

In the `pauseDeposits` function a user pauses deposits for the staking module if the `sig` param is a valid signature by the guardian.
The `pauseIntentValidityPeriodBlocks` param is set in constructor or `setPauseIntentValidityPeriodBlocks` and is more than `0` but this param has no limits and can have a big value.

Thus, the `sig` message will be valid and can be used again during a pause intent.

A possible attack flow:
1. A user with the `sig` param which is a valid signature pauses deposits for the staking module.
2. The owner calls `unpauseDeposits` with `stakingModuleId`.
3. During a pause intent another user calls `pauseDeposits` with this sig and pauses deposits for the staking module again.

https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol#L336
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol#L356
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol#L361

Recommendation

It's recommended to add limits for `pauseIntentValidityPeriodBlocks` and add a message to reuse the check.

> **Sponsor's commentary**
>
> The mentioned method can be used on behalf of the `Lido DAO Agent` contract that acts as an owner entity which is a part of the whole Lido protocol ACL setup. Therefore, to execute the change an on-chain Aragon vote is required and the Lido governance token holders accept associated risks of changing `pauseIntentValidityPeriodBlocks` if support the vote.

2.3 Medium

1. An unexpected overflow in the math function

Status

Fixed in commit:
https://github.com/lidofinance/lido-dao/commit/29efd7f69df2ef983ca07c43e7e855d05c3ca8c3.

Description

This bug was found with the Echidna fuzzer.
We have revert for the `type(int256).min` value in the `abs` function because the range for `int256` is **$-2^{255}...2^{255}-1$**.
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/common/lib/Math256.sol#L13

Recommendation

It's recommended to add a check for this case.

2. A possible frontrun attack for `depositBufferedEther` calls

Status

Acknowledged

Description

The `depositBufferedEther()` function allows to pass some `depositCalldata` for some users.
A user can frontrun a transaction with valid input data and replace it with `depositCalldata`.

https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol#L413
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol#L419

Recommendation

It's recommended to add `depositCalldata` to `msgHash` in the `_verifySignatures` function.

Sponsor's commentary

In the current design, the `depositCalldata` argument is the data might be necessary to prepare the deposit data by the staking module. Depending on the staking module implementation, it might be empty (`NodeOperatorsRegistry`) or contain complete deposit data (hypothetical staking module implementation that uses offchain storage for the deposit data). But for any staking module, the following is **ALWAYS** true: passed `depositCalldata` **MUST NOT** affect the deposit data set of the staking module (This requirement was explicitly added in the comments of the `IStakingModule.obtainDepositData()` method in the commit https://github.com/lidofinance/lido-dao/pull/729/commits/4c80206e016967989bb948ce0c42b9a009edf88b). In other words, the call of `StakingModule.obtainDepositData()` with different valid `depositCalldata` for the same `StakingModule.nonce` leads to the same unambiguous set of deposited validators. For example, the offchain staking module might store a hash of the expected `depositCalldata` onchain to validate it and reject any call with unexpected `depositCalldata`.

3. The range for a variable is not set

Status

Acknowledged

Description

The range for variable `_delay` is not set:

https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L1303.

A possible flow:

1. A user with `MANAGE_NODE_OPERATOR_ROLE` can set a big value for `_delay` accidentally.
2. A user with `STAKING_ROUTER_ROLE` calls `updateRefundedValidatorsCount` or `updateStuckValidatorsCount` functions and sets penalty in `stuckPenaltyStats` if needed:
3. https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L622
4. https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L644.
5. A user with `MANAGE_NODE_OPERATOR_ROLE` undoes `_delay` to the normal value but cannot undo `stuckPenaltyStats`.

Recommendation

It's recommended to add an allowable range for variable `_delay`.

> #### Sponsor's commentary
>
> The mentioned method can be used on behalf of the `Lido DAO Agent` contract that has a granted role which is a part of the whole protocol ACL setup. Therefore, the change requires an on-chain Aragon vote to enact and the Lido governance token holders accept associated risks of changing `STUCK_PENALTY_DELAY_POSITION` if support the vote.

2.4 Low

1. Unused functions

Status

Fixed in commit:
https://github.com/lidofinance/lido-dao/commit/57b675212a68db308e690e6daacc37ae99ca2f99.

Description

Unused functions in library `UnstructuredRefStorage`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/lib/UnstructuredRefStorage.sol#L7
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/lib/UnstructuredRefStorage.sol#L13.

Recommendation

It's recommended to remove unused functions.

2. Implicit type conversion

Status

Fixed in commit:
https://github.com/lidofinance/lido-dao/commit/0a9dbcbb1c035fff0c97128aee3a0f8530ece586.

Description

There are lost type `bytes` in `constructor`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/EIP712StETH.sol#L56.

Recommendation

It's recommended to add a `bytes` type to exclude the implicit type conversion

> **Sponsor's commentary**
>
> Decided to apply a different fix (to use implicit type conversions intentionally for other fields with a static string content).

3. Potential hash collisions for constants

Status

Acknowledged

Description

In hashes:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/access/AccessControlEnumerable.sol#L22
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/access/AccessControl.sol#L60
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/Versioned.sol#L24
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/PausableUntil.sol#L12.

Recommendation

It's recommended to improve the security of the constants by decreasing it by `-1` just like it is described in reference implementation in EIP-1967.

> **Sponsor's commentary**
>
> At the moment there is no significant risks associated with the finding. Changing the approach across the protocol would have posed additional risks of breaking compatibility and consistency across the codebase, tests, verification and deployment scripts.

4. Check null input data

Status

Fixed in commits:
https://github.com/lidofinance/lido-dao/commit/7e2a32f8b60e400a091d7a119e7095bb04a6e14b,
https://github.com/lidofinance/lido-dao/commit/ddf756779a07076460272a24b4491ef5542cd5e5.

Description

`admin`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/sanity_checks/OracleReportSanityChecker.sol#L149
`lido`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/AccountingOracle.sol#L157

Recommendation

It's recommended to add null data checks.

5. Not checking input array length

Status

Fixed in commit:
https://github.com/lidofinance/lido-dao/commit/da4ac3420f8dc7cd4ada08560926d0b183405751.

Description

It is expected that two arrays would have the same length as they contain corresponding data. But their lengths aren't compared.
`_stakingModuleIds` == `_totalShares`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/StakingRouter.sol#L260
`_stakingModuleIds` == `_exitedValidatorsCounts`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/StakingRouter.sol#L271

Recommendation

It's recommended to check that the arrays lengths are the same.

6. Typos

Status

Fixed in commit:
https://github.com/lidofinance/lido-dao/commit/25a53c093f530e6315a87d1b5c95911ee6d081fe.

Description

Typos in texts below:

1. `panlty` => `penalty`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L111
2. `timastamp` => `timestamp`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L113
3. `validoator` => `validator`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L656
4. `reawards` => `reawards`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/nos/NodeOperatorsRegistry.sol#L115
5. `memebrs` => `members`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/HashConsensus.sol#L308

Recommendation

It's recommended to correct them.

7. Paused staking check

Status

Fixed in commit:
https://github.com/lidofinance/lido-dao/commit/59b0476fef2c89a09e3b7cca264923eb6f9635b9.

Description

We have a comment in code:
"if staking was paused or unlimited previously, or the new limit is lower than previous, then reset the prev stake limit to the new max stake limit"
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/lib/StakeLimitUtils.sol#L148.

But conditions don't check the case when staking was paused:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.4.24/lib/StakeLimitUtils.sol#L151
Thus, we can add a condition: `_data.prevStakeBlockNumber == 0`.

Recommendation

It's recommended to add a check.

8. Magic numbers

Status

Acknowledged

Description

Use magic number `1e9`:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/AccountingOracle.sol#L585
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/oracle/AccountingOracle.sol#L610.

Recommendation

It's recommended to replace magic numbers with the constant.

> **Sponsor's commentary**
>
> An additional constant is redundant because the gwei-wei conversion is widely used and recognizable in this domain.

9. Extra emit

Status

Acknowledged

Description

The `_setMaxDeposits` function can override the same value and emit an extra event: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/DepositSecurityModule.sol#L189.

Recommendation

It's recommended to add a check for the previous value.

10. Unchecked return values and balances after the `transfer()/transferFrom()` functions

Status

Acknowledged

Description

Return values and balances are not checked:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.6.12/WstETH.sol#L57
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/Burner.sol#L166
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/Burner.sol#L199
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueue.sol#L380
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueue.sol#L390.

Recommendation

It's recommended to add checks for return values and balances.

> **Sponsor's commentary**
>
> The mentioned transfers are about Lido in-house token (stETH), which always returns true but reverts on failed transfers.

11. Additional checks in functions

Status

Acknowledged

Description

1. In function `findLastFinalizableRequestIdByTimestamp`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueueBase.sol#L147 If a `timestamp` is is in queue with `startId` for more than `_maxTimestamp`, then `finalizableRequestId` will be not found.
2. In function `findLastFinalizableRequestIdByBudget`: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueueBase.sol#L186 If `requiredEth` is in queue with `startId` for more than `_ethBudget`, then `finalizableRequestId` will be not found.

Recommendation

It's recommended to add conditions for the input values:

1. `_maxTimestamp < getQueue()[startId].timestamp` to: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueueBase.sol#L157.
2. Calc `requiredEth` for `startId` and add a `_ethBudget < requiredEth` check to: https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueueBase.sol#L198.

Sponsor's commentary

The mentioned code was deeply changed since the assessment's start.

12. Code improvements

Status

Fixed in commits:
https://github.com/lidofinance/lido-dao/commit/2032f0dc57521437457a934daebe16d98e40b2bc,
https://github.com/lidofinance/lido-dao/commit/d1facbd5f3f39cfffe31201f2142f1eea9e1e1ce.

Description

1. Double check
Function `resume` has a `whenPaused` modifier:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/WithdrawalQueue.sol#L101
and function `_resume` has a `whenPaused` modifier:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/utils/PausableUntil.sol#L54
It's recommended to remove the double check.

2. An unnecessary slot in the staking module name
In contract `StakingRouter` it's possible to create a staking module name with `length == 32`,
in this case it will be allocated in two slots: the first one will store only the size, the second one will store the name.

```
if (bytes(_name).length == 0 || bytes(_name).length > 32) revert StakingModuleWrongName();
```

You may restrict the name size with 31 bytes and the storage name in a single slot.
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/StakingRouter.sol#L179

Recommendation

It's recommended to fix them.

13. An uncontrollable event emit

Status

Acknowledged

Description

We have the `receiver` function in:
https://github.com/lidofinance/lido-dao/blob/e57517730c3e11a41e9cbc32ce018726722335b7/contracts/0.8.9/LidoExecutionLayerRewardsVault.sol#L75.
This accepts eth from anyone and emits an event but this method doesn't check that `msg.value > 0`. An attacker can call a contract with a null msg value in loop what will cause a large amount of `ETHReceived` events with a null argument what can DoS the off-chain logic.

Recommendation

It's recommended to add a check for the min value.

Sponsor's commentary

While the issue poses the risks for the off-chain tooling in some scenarios, the griefing attempts have intrinsic gas costs. Tools and widgets built for the Lido on Ethereum on behalf of the Lido DAO aren't relying on these events at all.

3. About MixBytes Camp

MixBytes Camp is a platform that connects blockchain projects and independent smart contracts auditors and security researchers.