STATE MIND

Insurance Fund

12-09-2022 - 15-09-2022

Table of contents

1. Project Brief

Title	Description
Client	Lido
Project name	Insurance Fund
Timeline	12-09-2022 - 15-09-2022
Number of auditors	7
Initial commit	e2aadf7b548a69860a3f535faaf7170712466463
Final commit	625d384f12c3df791085ecc2d15535e2121224d5

Short Overview

The Lido Insurance Fund is a contract that serves as a store for funds allocated for self-insurance purposes. This contract must securely store funds and allow the owner to have full access to the funds (transfer ERC20, ERC721, ERC1155 tokens, and ether).

Project Scope

The audit covered the following files:

InsuranceFund.sol

2. Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on its potential severity and has the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
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.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer 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 Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

3. Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	0
Informational	4

4. Conclusion

Commit with all fixes: `625d384f12c3df791085ecc2d15535e2121224d5`
No critical, high or medium severity issues were found, fixed 1 out of 4 issues, 3 acknowledged.

Deployment

File name	Contract deployed on mainnet
contracts/InsuranceFund.sol	0x8B3f33234ABD88493c0Cd28De33D583B70beDe35

5. Findings report

Informational

No zero-address check in the constructor	Fixed at PR#1

Description

Constructor allows to pass zero-address as the `_owner` parameter.

Recommendation

We recommend adding simple 'require' check if `_owner` is not zero-address.

Single point of failure in transferOwnership()	Acknowledged

Description

At the line InsuranceFund.sol#L14
Currently, `transferOwnership()` sets `_owner` if called with a non-zero address which is a potential risk if the address is not controlled by the Lido DAO and could lead to the loss of assets in the contract.
It is recommended to use a 2-step transfer of ownership where the pending owner has to accept ownership to become the owner.
For reference, OpenZeppelin's Ownable2Step.sol implementation.

Recommendation

Consider using a 2-step transfer of ownership.

Client's comments

> While we understand the benefit of the recommended ownership model, we will be proceeding with the original ownership contract. We do not plan to use the transfer ownership feature often (if ever), so the suggested model does not present any real benefits for us.

Directions of funds transfer from the Insurance Fund should be more restricted	Acknowledged

Description

In current implementation funds can be transfered to any address. But the logic of the contract implies that after the transfer funds should be used to cover slashing losses, which is a centralized proccess (for example, if they are all transferred to SelfOwnedStETHBurner.sol).

Recommendation

Based on the nature of contract, number of addresses which funds can be transferred to should be limited and monitored. That is why it makes sense to add a whitelist of addresses.

Client's comments

> We cannot predict the pool of recipients for transfers. Although the main recipient will be the DAO treasury, any assets that were sent to the vault by mistake will be transferred back directly to their appropriate owner whose address we obviously cannot know in advance. If we only whitelist the Treasury address, this will only increase operational overhead because now recovery will be a 2-step process. So we will proceed without having a whitelist.

Clone of OpenZeppelin repo

Acknowledged

Description

In your contracts there're two sources of OpenZeppelin contracts, your clone and brownie dependency.

Recommendation

In order to escape future out of sync and to make it more clear and safety, use only one trusted official version. Please remove your clone and make dependency of full version of OpenZeppelin lib. Change imports on trusted one. Delete dependency folder, cause it contains only OpenZeppelin contracts.

Client's comments

When working with packages there is always the threat of a supply chain attack, so we would rather bundle the dependencies into the repo and have one less point of failure.

6. Appendix A. Linter

Error/compiler-version

- contracts/InsuranceFund.sol:3 - Compiler version 0.8.10 does not satisfy the 0.8.13 semver requirement

Error/ordering

- contracts/InsuranceFund.sol:44 - Function order is incorrect, modifier definition can not go after constructor (line 37)

Error/max-line-length

- contracts/InsuranceFund.sol:70 - Line length must be no more than 100 but current length is 109.
- contracts/InsuranceFund.sol:84 - Line length must be no more than 100 but current length is 111.
- contracts/InsuranceFund.sol:105 - Line length must be no more than 100 but current length is 112.

7. Appendix B. Slither

Informational/High/low-level-calls

Low level call in InsuranceFund.transferEther(address,uint256): - (success) = _recipient.call{value: _amount}()

Low/High/shadowing-local

InsuranceFund.constructor(address)._owner shadows: - Ownable._owner (state variable)

8. Appendix C. Tests

Tests result

```
71 passed in 39.15s
```

Tests coverage

Function	Coverage
InsuranceFund.transferEther	100.0%
InsuranceFund.transferERC1155	50.0%

STATE
MIND