Lido Insurance Fund Security Review

Prepared by: BugBeast15

Table of Contents

Disclaimer

The audit team has conducted a thorough review of the code within the specified timeframe, focusing solely on security aspects. This report does not constitute an endorsement of the protocol or its underlying business model.

Risk Classification

Severity	Description
Critical	Bugs leading to asset theft or permanent fund locking
High	Bugs causing contract failure requiring manual intervention
Medium	Bugs breaking intended logic without direct fund loss
Informational	Minor issues with low immediate impact

Protocol Summary

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)

Audit Details

Scope

`InsuranceFund.sol` - Core insurance vault contract

Roles

Owner:

- Single EOA with full fund transfer rights
- Cannot renounce ownership

Executive Summary

Severity	Findings
Critical	0
High	0
Medium	0
Informational	4

Findings

Informational

[I-1]: Missing Zero-Address Check in Constructor

Description:
At the Line InsuranceFund.sol#L14
Constructor permitted zero-address ownership assignment, risking permanent fund locking.

Impact:
Permanent loss of fund control if zero-address set accidentally

Code:

```
constructor(address _owner) {
    _transferOwnership(_owner); // No zero-check
}
```

Recommendation

```
require(_owner != address(0), "Invalid owner");
```

[I-2] Single-Step Ownership Transfer

Description:
At the Line InsuranceFund.sol#L05 The contract uses a single-step ownership transfer mechanism, where ownership is immediately transferred upon calling `transferOwnership()`

```
// Inherited from OpenZeppelin's Ownable:
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    _transferOwnership(newOwner);
}
```

Impact:
Permanent loss of control over the contract if ownership is transferred to an incorrect or inaccessible address.

Recommendation Use a two-step ownership transfer pattern `OpenZeppelin's Ownable2Step` where:

- Current owner nominates a pending owner
- Pending owner must claim ownership

[I-3] Unrestricted Transfer Recipients

Description:
All transfer functions `transferEther`, `transferERC20`,`transferERC721()`,`transferERC1155()` allow sending to any non-zero address without additional restrictions.

Impact:
funds can be transfered to any address

Recommendation:
Implement recipient whitelist with emergency bypass

[I-4] Duplicated OpenZeppelin Dependencies

Description:
Local clones of OpenZeppelin contracts create version sync risks.

Recommendation:
Use official @openzeppelin/contracts npm package