SOLIDIFIED

Audit Report for Delphi Labs - May 26, 2021

Summary

Audit Report prepared by Solidified covering the Lido/Aave liquid staking pool smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code in several rounds. The debrief took place on 10 May 2021.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/delphidigital/lido-staked-aave

Commit number: `3dab1143a802e5ba2c78dc7c1f4995dd792e8280`

Update: The team has supplied fixes in commit number `c5b690520f6fcde234b0db36e62323dd5603997b`. These updates also introduce minor refactoring with changes in filenames.

The scope of the audit was limited to the following files:

```
contracts
├── FeeDistributor.sol
├── LdoAave.sol
├── interfaces
│   ├── IAaveGovernanceV2.sol
│   ├── ICustomProxyAdmin.sol
│   ├── IFeeDistributor.sol
│   ├── IGovernancePowerDelegationToken.sol
│   ├── ILSToken.sol
│   ├── ILdoAave.sol
│   └── IStakedAave.sol
├── libraries
│   ├── LSToken.sol
│   └── LSTokenGovPowerSnapshot.sol
└── proxies
    ├── CustomProxyAdmin.sol
    └── TransparentUpgradeableProxy.sol
```

Intended Behavior

The smart contracts implement an AAVE staking solution that simplifies reward distribution. Users do not need to claim their rewards. Instead, the users' balances are automatically increased to reflect their share.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for Delphi Labs - May 26, 2021

Test Coverage Report

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	90.22	54.55	76.92	90.22	
FeeDistributor.sol	90.48	63.64	84.62	90.48	97,98,102,103
LdoAave.sol	90	45.45	69.23	90	... 178,180,188
contracts/interfaces/	100	100	100	100	
IAaveGovernanceV2.sol	100	100	100	100	
ICustomProxyAdmin.sol	100	100	100	100	
IFeeDistributor.sol	100	100	100	100	
IGovernancePowerDelegationToken.sol	100	100	100	100	
ILSToken.sol	100	100	100	100	
ILdoAave.sol	100	100	100	100	
IStakedAave.sol	100	100	100	100	
contracts/libraries/	84.04	64.52	78.26	84.78	
LSToken.sol	75	50	68.75	75	... 118,120,178
LSTokenGovPowerSnapshot.sol	92	69.57	100	93.75	204,221,239
contracts/mocks/	100	81.25	100	100	
MockAaveGovernanceV2.sol	100	100	100	100	
MockChainlinkPriceFeed.sol	100	100	100	100	
MockERC20.sol	100	100	100	100	
MockLdoAaveUpgrade.sol	100	100	100	100	
MockStakedAaveV2.sol	100	81.25	100	100	
contracts/proxies/	89.36	65.38	95	90.38	
CustomProxyAdmin.sol	87.1	72.22	90	87.88	166,167,169,172
TransparentUpgradeableProxy.sol	93.75	50	100	94.74	47
All files	88.85	63.79	85.37	89.35	

SOLIDIFIED

Audit Report for Delphi Labs - May 26, 2021

Issues Found

Solidified found that the Lido contracts contain no critical issues, 1 major issue, 3 minor issues, in addition to 4 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	LSToken.sol: transferFrom() allows arbitrary tokens to be moved to an authorized receiver	Major	Resolved
2	CustomProxyAdmin.sol: Missing zero-checks for signer changes	Minor	Resolved
3	FeeDistributer.sol: Missing zero-checks	Minor	Resolved
4	Token susceptible to approve attack	Minor	Resolved
5	External calls to AAVE contracts	Note	-
6	CustomProxyAdmin.sol: propose() method accepts ETH in all cases	Note	-
7	FeeDistributor.sol: lidoAgent cannot be updated	Note	-
8	ICustomProxyAdmin.sol: Event SignerChanged() is not specific enough	Note	-

SOLIDIFIED

Audit Report for Delphi Labs - May 26, 2021

Critical Issues

No critical issues have been found.

Major Issues

1. LSToken.sol: transferFrom() allows arbitrary tokens to be moved to an authorized receiver

The function `transferFrom()` in `LSToken.sol` does not allow the approved address to send it to any address. Instead, it allows tokens to be moved only to an approved receiver. This non-standard behavior is due to checking the `to` parameter instead of `msg.sender` as the approved address.

Recommendation
It is recommended to use `msg.sender` instead of `to` address while validating the allowance during transfer.

Update
Resolved.

Minor Issues

2. CustomProxyAdmin.sol: Missing zero-checks for signer changes

Function `changeSigner()` does not perform checks for `address(0)` meaning that upgradability could accidentally be renounced.

Recommendation
Add the following precondition:

```
require(newSigner != address(0));
```

Update
Resolved.

3. FeeDistributer.sol: Missing zero-checks

Functions `setDelphiAgent()` and `setLidoAgent()` do not perform checks for `address(0)` meaning that fee receiver addresses could accidentally be renounced unrecoverably.

Recommendation
Add zero-checks

Update
Resolved.

4. Token susceptible to approve attack

Changing an allowance through the `approve()` method brings the risk that someone may use both the old and the new allowance by unfortunate transaction ordering. A detailed description of this vulnerability can be found here: https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM

Recommendation
One possible solution to mitigate this race condition is to implement `increaseAllowance()` and `decreaseAllowance()` functions.

Update
Resolved.

Informational Notes

5. External calls to AAVE contracts

The external calls to the AAVE protocols are not protected from reentrancy. This is fine with the current implementation of the official AAVE protocol. However, a number of AAVE forks and clones exist with a modified codebase. In these cases, caution is advised.

Recommendation
Ensure the codebase is only used with trusted AAVE versions or forks.

6. `CustomProxyAdmin.sol`: `propose()` method accepts ETH in all cases

The `propose()` method accepts funds even when it is not being used during proposal approval. This might be unintentional in the cases of `ProposalType.ADMIN` and `ProposalType.UPGRADE`.

Recommendation
Whilst this situation is recoverable, it is recommended to validate or return the extra funds sent when it is not used.

7. `FeeDistributor.sol`: `lidoAgent` cannot be updated

Consider implementing a `setLidoAgent()` function in case `lidoAgent` needs to be updated in the future.

8. ICustomProxyAdmin.sol: Event SignerChanged() is not specific enough

The SignerChanged() event is not specific to which particular signer has been changed.

Recommendation
Consider implementing separate `Signer1Changed()` and `Signer2Changed()` events.

Audit Report for Delphi Labs - May 26, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Delphi Labs or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.