LIDO AAVE STETH SMART CONTRACT AUDIT

February 07, 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 . 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.The goal is to provide the ability to deposit stETH into AAVE and allow to use it as collateral and for variable rate borrowing.
aToken uses underliyng stETH shares to store balances and implement rebase ability.
Motivation behind this design is to encourage using
stETH as collateral rather than borrowing it. stETH is pegged steadily to ETH, so using it as collateral involves low liquidation risks.Contracts

- `IncentivizedERC20.sol` - basic ERC20 implementation. When transferring tokens, the `handleAction()` method is called from `_incentivesController`. As `IncentivesController` address should be provided on deploy and couldn't be upgraded, proxies addresses will be provided for both tokens. Lido DAO agent would be owner of both proxies to provide ability to upgrade it via the Lido DAO voting.
- `StableDebtToken.sol` - implements a stable debt token to track the borrowing positions of users at stable rate mode.
- `DebtTokenBase.sol` - base contract for different types of debt tokens, like `StableDebtToken` or `VariableDebtToken`.
- `AStETH.sol` - Rebaseable astETH token has an additional book-keeping layer on top of the existing aToken structure.
Generic aTokens have a private balance and public balance. The internal balance corresponds to the deposited balance without accrued interest.
The external balance corresponds to the deposited balance with interest.
- `StableDebtStETH.sol` - `StableDebtStETH` is inherited from `StableDebtToken`. There is only one difference between the prohibition of the `mint()` function.
- `VariableDebtStETH.sol` - implementation makes no functional changes to the generic implementation. When the debt tokens are mint and burn,
it performs additional operations to keep track of the amount of borrowed stETH in shares. This amount is stored as `_totalSharesBorrowed`.
The new `getBorrowData()` method returns the amount of borrowed shares and the total supply of the debtToken (is equal to the amount of borrowed stETH)
which are used for astETH math. The debtToken is non-rebasable token; the debt value is equal to the borrow + interest even after the rebasing of stETH.
AAVE protocol allows the use of incentives controllers in their `AToken`, `VariableDebtToken`, and `StableDebtToken` contracts
to distribute rewards on a token mint, burn or transfer. Lido's integration in AAVE uses a custom implementation of `AToken` - `AStETH`.
Repo contains two types of incentives controller implementation that can be used with `AStETH` - `AaveIncentivesControllerStub`
and `AaveAStETHIncentivesController`. `AStETH` token doesn't allow to change incentives controller after deployment.

To allow update implementation of incentives controller for `AStETH` both `AaveIncentivesControllerStub` and `AaveAStETHIncentivesController`
inherit `UUPSUpgradable` and `Ownable` contracts and would be deployed behind `ERC1967Proxy` contract, from the `OpenZeppelin` package.Contracts

- `UnstructuredStorageVersionised.sol` - Encapsulates the logic for initializing and upgrades proxied contracts on a versioned basis by the dedicated owner
- `AaveIncentivesControllerStub.sol` - Contains logic with empty implementation of IAaveIncentivesController's `handleAction()` method. Inherits from `UnstructuredStorageVersionised.sol` contract.
- `AaveAStETHIncentivesController.sol` - Contains logic to the linear distribution of reward tokens across holders of AStETH, proportional to the number of tokens the user hold. Contract inherits from `UnstructuredStorageVersionised.sol` contract and implements Unstructured Storage pattern to simplify future updates of incentivization logic.
- `RewardsUtils.sol` - Provides structs and a library for convenient work with staking rewards distributed in a time-based manner.

1.4 PROJECT DASHBOARD

Client	Lido
Audit name	Aave stETH
Initial version	f9096e3a66ef96a59147a40f7cd045eb7e90e133 12c9111990c9699e84a36f30ba849486ef8f2a84
Final version	2a42cb58d49c350d72c87614f0cf86819b29daa3 3f7fab329403553df5a39449735c1d8e2debe403
Date	November 15, 2021 - February 07, 2022
Auditors engaged	3 auditors

FILES LISTING

UInt256Lib.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/dependencies/uFragments/UInt256Lib.sol
IAToken.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/interfaces/IAToken.sol

ILendingPool.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/interfaces/ILendingPool.sol
MathUtils.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/libraries/math/MathUtils.sol
WadRayMath.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/libraries/math/WadRayMath.sol
DataTypes.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/libraries/types/DataTypes.sol
IncentivizedERC20.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/tokenization/IncentivizedERC20.sol
StableDebtToken.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/tokenization/StableDebtToken.sol
DebtTokenBase.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/tokenization/base/DebtTokenBase.sol
AStETH.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/tokenization/lido/AStETH.sol
StableDebtStETH.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/tokenization/lido/StableDebtStETH.sol
VariableDebtStETH.sol	https://github.com/lidofinance/aave-protocol-v2/blob/12c9111990c9699e84a36f30ba849486ef8f2a84/contracts/protocol/tokenization/lido/VariableDebtStETH.sol
AaveAStETHIncentivesController.sol	https://github.com/lidofinance/aave-asteth-incentives-controller/blob/f9096e3a66ef96a59147a40f7cd045eb7e90e133/contracts/incentives/AaveAStETHIncentivesController.sol
AaveIncentivesControllerStub.sol	https://github.com/lidofinance/aave-asteth-incentives-controller/blob/f9096e3a66ef96a59147a40f7cd045eb7e90e133/contracts/incentives/AaveIncentivesControllerStub.sol

RewardsUtils.sol	https://github.com/lidofinance/aave-asteth-incentives-controller/blob/f9096e3a66ef96a59147a40f7cd045eb7e90e133/contracts/utils/RewardsUtils.sol
UnstructuredStorageVersionised.sol	https://github.com/lidofinance/aave-asteth-incentives-controller/blob/f9096e3a66ef96a59147a40f7cd045eb7e90e133/contracts/versioning/UnstructuredStorageVersionised.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	1
Warning	5
Comment	5

CONCLUSION

Smart contracts have been audited and several suspicious places have been detected. During the audit, no critical problems were found, one major, several warnings, and comments were identified. After working on the reported results, they were all fixed or confirmed by the client.Final commit identifier with all fixes:
`2a42cb58d49c350d72c87614f0cf86819b29daa3` for https://github.com/lidofinance/aave-protocol-v2/,
`3f7fab329403553df5a39449735c1d8e2debe403` for https://github.com/lidofinance/aave-asteth-incentives-controller/

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

- AStETH.sol: 0xbd233D4ffdAA9B7d1d3E6b18CCcb8D091142893a
- StableDebtStETH.sol: 0x8180949ac41ef18e844ff8dafe604a195d86aea9
- VariableDebtStETH.sol: 0xde2c414b671d2db93617d1592f0490c13674de24

For all contracts, the `incentivesController` address is `0x0000000000000000000000000000000000000000`.

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Possible incorrect `scaledTotalSupply` calculation
File	AStETH.sol
Severity	Major
Status	Fixed at 23f16e68

DESCRIPTION

At the line
AStETH.sol#L595
if the shares are below zero then value may overflow and scaled total supply calculation will be wrong.

RECOMMENDATION

Before converting a negative number to the `uint256` type, you must make it positive.

CLIENT'S COMMENTARY

After disabling the borrowing of stETH, the method with this issue was removed from the contract.
Commit with fix is 23f16e68
Updated commit: 7cefeab3

2.3 WARNING

WRN-1	No validation of the address parameter value in contract constructor
File	AStETH.sol IncentivizedERC20.sol DebtTokenBase.sol DebtTokenBase.sol AaveAStETHIncentivesController.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The variable is assigned to the value of the constructor input parameter. But this parameter is not checked before this.
If the value turns out to be zero, then it will be necessary to redeploy the contract, since there is no other functionality to set this variable.

- At the line AStETH.sol#L81 the `POOL` variable is set to the value of the `pool` input parameter.

- At the line AStETH.sol#L82 the `UNDERLYING_ASSET_ADDRESS` variable is set to the value of the `underlyingAssetAddress` input parameter.

- At the line AStETH.sol#L83 the `RESERVE_TREASURY_ADDRESS` variable is set to the value of the `reserveTreasuryAddress` input parameter.

- At the line IncentivizedERC20.sol#L37 the `_incentivesController` variable is set to the value of the `incentivesController` input parameter.

- At the line DebtTokenBase.sol#L47 the `POOL` variable is set to the value of the `pool` input parameter.

- At the line DebtTokenBase.sol#L48 the `UNDERLYING_ASSET_ADDRESS` variable is set to the value of the `underlyingAssetAddress` input parameter.

- At the line AaveAStETHIncentivesController.sol#L62 the `REWARD_TOKEN` variable is set to the value of the `_rewardToken` input parameter.

- At the line AaveAStETHIncentivesController.sol#L63 the `STAKING_TOKEN` variable is set to the value of the `_stakingToken` input parameter.

RECOMMENDATION

In all the cases, it is necessary to add a check of the input parameter to zero before initializing the variables.

CLIENT'S COMMENTARY

No issue: All variables passed to the constructor will be double-checked before and after deployment. In case of wrong parameters passed, there is always an option to redeploy a contract with correct values.

WRN-2	Unlimited rights for the owner of the contract
File	AaveAStETHIncentivesController.sol
Severity	Warning
Status	Fixed at 9d4e96de

DESCRIPTION

There is nothing in the doc about the wallet of the contract owner.
At the line AaveAStETHIncentivesController.sol#L120 is set to the value of the variable using the `_setRewardsDuration ()` function.
The value of this variable is used in the `notifyRewardAmount ()` function on lines 153, 157.
If the value of the variable is equal to zero, then the work of the `notifyRewardAmount()` function will be blocked.
The `notifyRewardAmount ()` function can only be called by RewardsDistributor, but it can be blocked from another wallet.
The single wallet of the contract owner can be compromised. Better to use multisignature.

Another fact in favor of using multisignature is that the owner of the contract calls the `recoverERC20 ()` function on lines AaveAStETHIncentivesController.sol#L172-L175. The owner can always withdraw all tokens.

RECOMMENDATION

It is recommended to use multisignature to call functions from the contract owner.

CLIENT'S COMMENTARY

The owner of the contract will be set to Lido's Aragon Agent, and calls of admin methods of the contract might be possible only on behalf of the DAO. To restrict the rights of the Agent and exclude possible damage to AAVE's protocol, upgradability was removed from the`AaaveAStETHIncentivesController`, and updates of the `IncentivesController` might be done only by the AAVE governance via upgrading the implementation of the AStETH contract.
For rewards distributor will be used the standalone `RewardsManager` contract, used by Lido in many other integrations, which simplifies interaction with the incentives controller.
Fix was done in 9d4e96de

WRN-3	Extra condition
File	SignedSafeMath.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

If you update the compiler version from `0.6.12` to `0.8.x`. Then you can remove the extra code.
At the line
SignedSafeMath.sol#L29 an unnecessary check is performed…
If the value of the variable `a` is equal to `-1`, `-2`, `-3` and so on, the transaction will not be executed.
A similar superfluous condition on the line
SignedSafeMath.sol#L51.

RECOMMENDATION

You need to update the compiler version and remove unused code.

CLIENT'S COMMENTARY

No issue: Such change will require massive refactoring in the AAVE protocol contracts. It is out of the scope of the integration.

WRN-4	`claimReward()` may be external
File	AaveAStETHIncentivesController.sol
Severity	Warning
Status	**Fixed** at **5da77704**

DESCRIPTION

At the line
AaveAStETHIncentivesController.sol#L124
`claimReward()` function may be external, there is no internal using.

RECOMMENDATION

It is recommended to make it external.

CLIENT'S COMMENTARY

The visibility of the method was changed in 5da77704

WRN-5	`initialize()` should be private
File	AaveAStETHIncentivesController.sol
Severity	Warning
Status	Fixed at 9d4e96de

DESCRIPTION

At the line
AaveAStETHIncentivesController.sol#L74
`initalize()` function should be private because it is executed in the constructor once.

RECOMMENDATION

It is recommended to make it private.

CLIENT'S COMMENTARY

After removing the upgradability from the contract, the initialize() method will not be used in the constructor but later by the owner to set the address of the staking token, so it should be public. Fix was done in 9d4e96de

2.4 COMMENT

CMT-1	Unknown data in comments
File	VariableDebtStETH.sol
Severity	Comment
Status	**Fixed** at **23f16e68**

DESCRIPTION

- At the line VariableDebtStETH.sol#L19 has an unused `_totalGonsBorrowed` variable.
- At the line VariableDebtStETH.sol#L22 has an unused `fetchBorrowData()` variable.
- At the line VariableDebtStETH.sol#L23 has an unused `fetchTotalSupply()` variable.

RECOMMENDATION

It is recommended to delete the description of unused data.

CLIENT'S COMMENTARY

Comments were deleted in the commit 23f16e68
Updated commit: 7cefeab3

CMT-2	Reducing the source code
File	AStETH.sol
Severity	Comment
Status	**Fixed** at **23f16e68**

DESCRIPTION

At the line
AStETH.sol#L564 describes the `extData` variable.
But the description of this variable can be done on the line
AStETH.sol#L563 and remove line 564:

```
function _fetchExtData() internal view returns (ExtData memory extData) {
```

RECOMMENDATION

We recommend refactoring your source code.

CLIENT'S COMMENTARY

The code was refactored, and the whole method `_fetchExtData()` was removed.
The fix is in the commit 23f16e68
Updated commit:
7cefeab3

CMT-3	Internal function is not used anywhere
File	VariableDebtStETH.sol
Severity	Comment
Status	Fixed at 7cefeab3

DESCRIPTION

At the lines
VariableDebtStETH.sol#L174-L176 there's an internal function `fetchStETHTotalSupply ()`.
But in the current contract and in other contracts, it is not called anywhere.

RECOMMENDATION

Unused code must be removed.

CLIENT'S COMMENTARY

The unused method was removed in the commit 23f16e68
Updated commit: 7cefeab3

CMT-4	Code duplication
File	WadRayMath.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the library

- WadRayMath.sol
  there are four methods which can be reduced to two functions: `wadDiv()` and `rayDiv()` can be united to single function with addition param.
  And `wadMul()` and `rayMul()` functions can be united too.

Recomendation

It is recommended to combine these methods.

RECOMMENDATION

CLIENT'S COMMENTARY

No issue: The above methods are used widely across the AAVE protocol contracts, and such refactoring will be hard to implement.

CMT-5	Mistake in comment or in method
File	AaveAStETHIncentivesController.sol
Severity	Comment
Status	**Fixed** at **e47d3918**

DESCRIPTION

At `notifyRewardAmount()` method of
AaveAStETHIncentivesController.sol#L143
added comment `@param rewardHolder Address to retrieve reward tokens` that means that `rewardHolder` is the address which will receive `REWARD_TOKEN`.
But along method logic this address transfers `REWARD_TOKEN` to this contract at the line:
AaveAStETHIncentivesController.sol#L148.

RECOMMENDATION

The comment needs to be corrected.

CLIENT'S COMMENTARY

The comment was corrected in the commit e47d3918

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