Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

December 23rd 2021 — Quantstamp Verified

Lido stETH Aave Integration

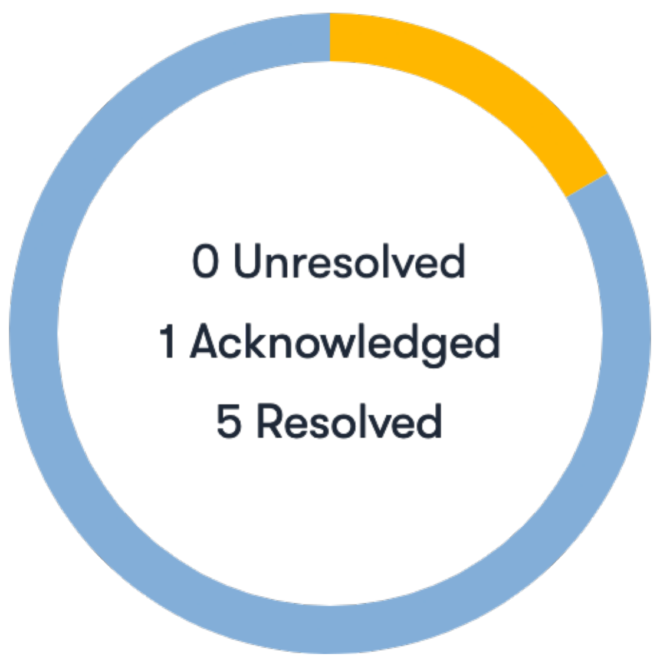
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Decentralized Lending
Auditors	Jake Goh Si Yuan, Senior Security Researcher Poming Lee, Research Engineer Mohsen Ahmadvand, Senior Research Engineer
Timeline	2021-12-02 through 2021-12-23
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	HackMD Audit Documentation New Documentation
Documentation Quality	High
Test Quality	High
Source Code	

Repository	Commit
aave-protocol-v2	12c9111
None	7cefeab

Total Issues	**6** (5 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**3** (3 Resolved)
Low Risk Issues	**2** (2 Resolved)
Informational Risk Issues	**1** (0 Resolved)
Undetermined Risk Issues	0 (0 Resolved)



FAST RESPONSE TIMES

ALL ISSUES ADDRESSED

Documentation Preparedness

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
? Undetermined	The impact of the issue is uncertain.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

We have performed an audit for the proposed Lido stETH Aave integration, and have uncovered 6 issues ranging from Medium to Informational. There were some opaqueness to this audit, such as the correctness and safety of the mint and burn formulas provided, the stand-in Incentives Controller, and external contract logic, which we have noted in the issues that follows. We have also made several documentation and best practices recommendations that would go a long way in improving the codebase.
Overall, we found the code well-documented and adequately written. That being said, we urge the team to consider our findings and suggestions, and to either fix them or write an official acknowledgement about it.

Update(2021-12-23): The reaudit results and fixes came back promptly with some well documented changes as noted in "New Documentation" above. There was a major change to the underlying features of the AStEth token. Borrowing was removed entirely, which rendered many of the issues fixed as the code was no longer existent. The change simplified the logic substantially, and we have also performed a best-effort review of it and found no major errors to report.

ID	Description	Severity	Status
QSP-1	Debt token contracts initialization is not protected, and can be done again	⌃ Medium	Fixed
QSP-2	Formulas in `_mintScaled` and `_burnScaled` cannot be verified	⌃ Medium	Fixed
QSP-3	Possible integer overflow due to naked arithmetic	⌃ Medium	Fixed
QSP-4	Unsafe sanity checks	⌄ Low	Fixed
QSP-5	Violation of principle of least privilege in `Incentives Controller`	⌄ Low	Mitigated
QSP-6	Reliance on external contract for critical accounting logic	○ Informational	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.0

Steps taken to run the tools:

Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .`

Findings

QSP-1 Debt token contracts initialization is not protected, and can be done again

Severity: Medium Risk

Status: Fixed

File(s) affected: `AStETH.sol`

Description: Currently, the `initializeDebtTokens` function does not check for whether the contract state variables `_stableDebtStETH` and `variableDebtStETH` are already set. This means that there is a possibility of these critical state variables are initialized and set again through another call.
The function is also not access-controlled, which exacerbates the probability of it taking place. This violates the principle of least privilege.

Recommendation: The `initializeDebtTokens` function should provide a validation that checks for whether the two variables are already set. It can come in the form of `require(_variableDebtStETH == address(0) && _stableDebtStETH == address(0));.`

Update: The relevant logic and state were removed as part of a large change in the `AStETH.sol` contract.

QSP-2 Formulas in `_mintScaled` and `_burnScaled` cannot be verified

Severity: Medium Risk

Status: Fixed

File(s) affected: `AStETH.sol`

Description: The formula used in `_mintScaled` and `_burnScaled` for minting and burning appears to be correct in its unit and its underlying rationale of manipulating the numbers, although the implied derivation of the final equation is not immediately clear.
What's more, based on the currently given information (i.e., the technical specification, code comments, and the code itself), we were unable to prove the formulas correct and verify that it does not contain edge cases that would provide a malicious actor some advantages.

Recommendation: A more concrete and solid mathematical derivation through substitution would go a long way to proving correctness. If that cannot be done, then the team should perform simulation on it to ensure that the characteristics of the formulas are well understood, and that the edge cases found are tolerable by the system.

Update: The relevant logic and state were removed as part of a large change in the `AStETH.sol` contract.

QSP-3 Possible integer overflow due to naked arithmetic

Severity: Medium Risk

Status: Fixed

File(s) affected: `AStETH.sol`

Description: There is a usage of naked arithmetic on AStETH.sol::L594-596 `ISTETH(UNDERLYING_ASSET_ADDRESS).getPooledEthByShares(uint256(_totalShares - e.totalSharesBorrowed)) + e.totalPrincipalBorrowed;`. It is entirely possible to overflow with these operations, especially considering that the data input is likely to be coming from an external contract.

Recommendation: Use SafeMath for the two operations.

Update: The relevant logic and state were removed as part of a large change in the `AStETH.sol` contract.

QSP-4 Unsafe sanity checks

Severity: Low Risk

Status: Fixed

Description: The following condition checks should be set to `revert` instead of a gentler `if`, as these conditions should never be violated.

1. `L528 if (userBalanceScaledBefore <= scaledTotalSupplyBefore)`
2. `L533 if (burnAmountScaled <= scaledTotalSupplyBefore)`
3. `L538 if (burnAmountScaled <= userBalanceScaledBefore)`

Recommendation: Switch the `if` statements for a stronger `require`.

Update: The relevant logic and state were removed as part of a large change in the `AStETH.sol` contract.

QSP-5 Violation of principle of least privilege in `Incentives Controller`

Severity: Low Risk

Status: Mitigated

File(s) affected: `AStETH.sol`

Description: In the HackMD Audit Documentation, it is stated that "We realize that having IncentivesController owned externally may involve extra security risks for AAVE. Lido is open to other options, if there is a process to upgrade an AAVE owned contract in the future.".
It is unclear from the code itself what such a contract would be, as it is currently out of scope, and it is stated in the same documentation that "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."
Given that this is a component that affects both Aave and Lido, two separate actors, it would be much better if the proxy upgrade process is administered bilaterally instead of the proposed unilateral action here. Recommendation: It is suggested that the upgrade operation should be controlled by both AAVE side and Lideo side through a multisig wallet in order to apply the principle of least privilege to the system.

Recommendation: The upgrade operation should be controlled by both Aave and Lido through a bilateral action, such as a shared multi-sig or an equivalent simpler analogue.

Update: From the Lido team: "Upon discussion with the AAVE team, the usage of upgradable proxy for IncentivesController was revised. The current implementation of AStETH will not be using IncentivesController at all. If necessary, the whole implementation of AStETH will be upgraded later with passed IncentivesController address."

QSP-6 Reliance on external contract for critical accounting logic

Severity: Informational

Status: Acknowledged

File(s) affected: `AStETH.sol`, `VariableDebtStETH.sol`

Description: There are several important calculations that rely on an external contract for critical accounting logic. It appears in the following:

1. AStETH.sol. A subset of functions in the scope of the audit rely on `getSharesByPooledEth`. Since this function resides outside the scope of this audit, we assume this function is correctly implemented. That is, there may exists some pitfalls in the stated function.
2. VariableDebtStETH.sol. Upon minting `_totalSharesBorrowed` is incremented by `getSharesByPooledEth(_scaledAmount)`. Upon burning the same value is deducted from `_totalSharesBorrowed`. It is crucial that the `getSharesByPooledEth` function carries out the rebasing correctly.

Recommendation: Consider having the stated function reviewed to ensure its correctness and safety.

Update: From the Lido team: The method `getSharesByPooledEth` implemented in the StETH contract were reviewed and audited carefully. Detailed audit reports might be found here:
https://github.com/lidofinance/audits/blob/main/Sigma%20Prime%20-%20Lido%20Finance %20Security%20Assessment%20Report%20v2.1.pdf
https://github.com/lidofinance/audits/blob/main/QSP%20Lido%20Report %2012-2020.pdf

Automated Analyses

Slither

Slither detected 232 results, of which we have found that they were all false-positives.

Code Documentation

1. AStETH.sol::L193: `ccrued` -> ` accrued.
2. VariableDebtStETH.sol::L22-23 has irrelevant legacy comments from where the contract was probably copied and modified from.
3. AStETH.sol::L577-578 describes the input parameters as `_intBalanceOf` and `_intTotalSupply` but it is clear that the type is `uint256`. It should be more appropriately named `_uint...` instead.
4. AStETH.sol::L268 has a redundant `@return The scaled balance of the user` that is overwritten by the following line `@return The scaled balance and the scaled total supply`. It should be removed.
5. AStETH.sol::L287 has a redundant `@return The internal balance of the user` that is overwritten by the following line `return The internal balance and the scaled total supply`. It should be removed.
6. AStETH.sol::L285,L288 states that the return object contains the `scaled total supply`, when it is likely to be referring to `internal total supply`.

Adherence to Best Practices

1. AStETH.sol::L401-404 should only be done under the `if(validate)` context as `fromBalanceScaled` and `toBalanceScaled` are only ever used within it.
2. AStETH.sol::[L261, L277, L301] could use the function `scaledTotalSupply()` instead, which would be cleaner.
3. AStETH.sol::mint() should check for whether `amount > 0` at the top, instead of doing so at L166 after some operations have been performed.

Test Results

Test Suite Results

The test suite was executed succesfully, with the log appended as follows:

```
StETH aToken
  steth rebasing
    positive rebase
       ✓ should update total supply correctly
    negative rebase
       ✓ should update total supply correctly
  user Transfer
    when lenderA deposits 1000 StETH, transfers all to himself
       ✓ should update balances correctly
    the single deposit after rebase
       ✓ should update balances correctly
    the first deposit after rebase for lender
       ✓ should update balances correctly
    several sequintial deposits
       ✓ should mint aTokens correctly
    deposit->borrow->rebase->repay->deposit->rebase
       ✓ should mint aToken correctly
    when lenderA deposits 1000 StETH, transfers more than he has
       ✓ should update balances correctly
    when borrowed amount > 0
      when lenderA deposits 1000 StETH, transfers all to himself
         ✓ should update balances correctly
      when lenderA deposits 1000 StETH, transfers more than he has
         ✓ should update balances correctly
  user deposit
    first deposit
       ✓ should mint correct number of astETH tokens
       ✓ should update balances after positive rebase
       ✓ should update balances after negative rebase
       ✓ should update balances after neutral rebase
    lone user
       ✓ should mint correct number of astETH tokens
       ✓ should update balances after positive rebase
       ✓ should update balances after negative rebase
    many users
       ✓ should mint correct number of astETH tokens
       ✓ should update balances after positive rebase
       ✓ should update balances after negative rebase
    v large deposit
       ✓ should mint correct number of astETH tokens
    when borrow>0
       ✓ should mint correct number of astETH tokens
       ✓ should update balances on positive rebase
```

```
        ✓ should update balances on negative rebase
    user withdraw
      single deposit partial withdraw
        ✓ should burn correct number of astETH tokens
      single deposit full withdraw
        ✓ should burn correct number of astETH tokens
        ✓ should burn correct number of astETH positive rebase
        ✓ should burn correct number of astETH negative rebase
      lone user multiple withdraws
        ✓ should burn correct number of astETH tokens
      multiple withdraws
        ✓ should burn correct number of astETH tokens
      v large withdraw
        ✓ should burn correct number of astETH tokens
      when borrow>0
        ✓ should burn correct number of astETH tokens
        ✓ should update balances on positive rebase
        ✓ should update balances on negative rebase
    user borrow repay with interest
      ✓ should update accounting
    user borrow repay with positive rebase
      ✓ should update accounting
    user borrow repay with negative rebase
      ✓ should update accounting
    multi user borrow repay
      ✓ should update accounting
```

Code Coverage

The attempt to run code coverage failed.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
5ff9282a57f5a11506d84339146c9e73be6f20cc2fdb25706548ed5e8661e0ac ./lido/StableDebtStETH.sol
50b6c7cc6a59b9a04beff4b20e3b2719af041cb6524acf844c4d73ae0716a639 ./lido/VariableDebtStETH.sol
c40a2374ba159b4b7208cbd010b608289bc3dc9e555e618fd197acda6315248a ./lido/AStETH.sol
```

Tests

```
75661ea331d1855e8565e6cd5208019790f319595c9a20d8a6389abba709d006 ./test/astETH.spec.ts
```

Changelog

- 2021-12-02 - Initial report of 12c9111
- 2021-12-23 - Final report of 7cefeab

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.