Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

December 19th 2020 — Quantstamp Verified

lido-dao

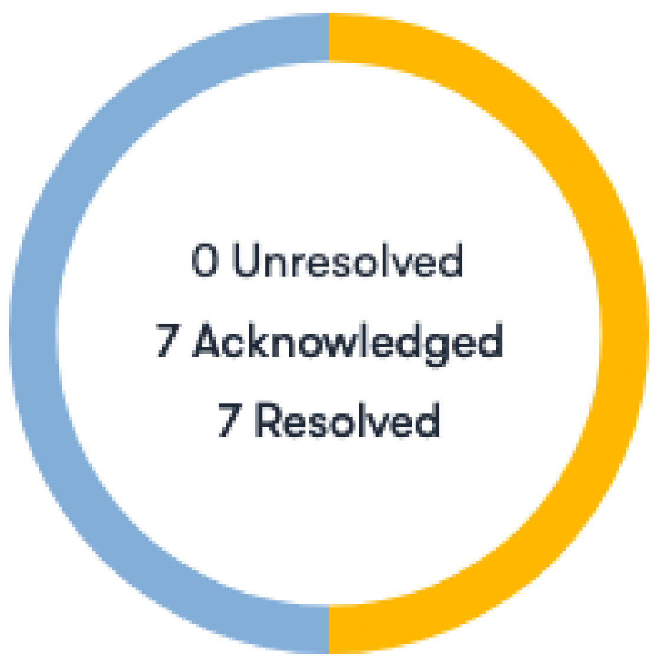
This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	Eth2 Staking Provider Aggregator
Auditors	Ed Zulkoski, Senior Security Engineer Luís Fernando Schultz Xavier da Silveira, Security Consultant Joseph Xu, Technical R&D Advisor
Timeline	2020-11-04 through 2020-12-11
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md The Lido Primer Document
Documentation Quality	High
Test Quality	High
Source Code	

Repository	Commit
depool-dao	v0.1.0-rc.1 (initial report)
lido-dao	v0.2.0 (revised report)
lido-dao	v0.2.1-rc.0 (revised report)

Total Issues	14 (7 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	1 (0 Resolved)
Low Risk Issues	4 (3 Resolved)
Informational Risk Issues	2 (2 Resolved)
Undetermined Risk Issues	7 (2 Resolved)



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

During the initial audit, a total of 13 issues were raised of varying severity. Several of these issues were related to TODOs in the code or unclear semantics of several functions.In addition, several suggestions for documentation and best practice improvements were made. An additional issue was raised during a re-audit, bringing the total number of issues to 14. We recommend addressing all issues before using the code in production.

The codebase has gone through major updates during the audit process. The developers made changes that were unrelated to the issues raised in the initial audit, which included addition of new features and changes to the contract architecture. While the Quantstamp team has audited these changes to the extent possible in subsequent re-audits (including new test suites), it should be noted that the majority of the audit effort was focused on the initial `depool-dao` repository.

Update: Most issues have been resolved as of commit `d4171a1`. One new issue was raised in QSP-14.
Update 2: One outstanding issue from the initial audit has been resolved as of commit `ad4b2f6`.
Update 3: QSP-14 is marked as Acknowledged, with detailed comments from the Lido team regarding the plan for implementing the fix and the procedure to handle the issue in the meantime.

ID	Description	Severity	Status
QSP-1	Users cannot withdraw until phase 2 of Eth2	^ Medium	Acknowledged
QSP-2	Unresolved TODOs in code	⌄ Low	Fixed
QSP-3	`depositIterationLimit` not enforced by `submit` function	⌄ Low	Fixed
QSP-4	`setQuorum` may cause the new epoch data to pre-emptively finalize	⌄ Low	Fixed
QSP-5	Potential gas-cost issues in `mode` function	⌄ Low	Acknowledged
QSP-6	Privileged Roles and Ownership	○ Informational	Fixed
QSP-7	Missing return values	○ Informational	Fixed
QSP-8	Unclear access control policy for `transferToVault`	? Undetermined	Acknowledged
QSP-9	`_ETH2Deposit` staking strategy may possibly lead to increased slashed funds	? Undetermined	Acknowledged
QSP-10	Potential duplicate reward addresses	? Undetermined	Acknowledged
QSP-11	`StETH` must be able to upgrade referenced `DePool` contract	? Undetermined	Fixed
QSP-12	Unclear mechanics of `burn` function	? Undetermined	Acknowledged
QSP-13	Unclear use of `safeTransferFrom`	? Undetermined	Fixed
QSP-14	`removeOracleMember()` does not clear the last update	? Undetermined	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 v0.6.13
- Mythril v0.22.13

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`
3. Installed the Mythril tool from Pypi: `pip3 install mythril`
4. Ran the Mythril tool on each contract: `myth -x path/to/contract`

Findings

QSP-1 Users cannot withdraw until phase 2 of Eth2

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `DePool.sol`

Description: Users will not be able to withdraw Ether after depositing to this contract as of now because the `withdrawal` function is explicitly marked as a WIP. Large withdrawals are only possible after Phase 2 of Eth2 launch.

Recommendation: Although a full resolution of this issue is dependent on the phase 2 launch of Eth2, users should be made aware that funds will be locked through documentation. Ensure that the users properly understand the risk in depositing to the pool, especially in terms of the execution risk for both Eth2 and DePool. In addition, it may be a good idea to prevent automatic deposit through the fallback function.
Update: Additional documentation has been added.

QSP-2 Unresolved TODOs in code

Severity: *Low Risk*

Status: Fixed

File(s) affected: `DePool.sol`

Description: On L351, there is the comment: `// TODO a separate vault`. As of now, the function `getInsuranceFund` returns the same address as `getTreasury`.
Further, the treasury address is not part of the initialization and there is currently no way to update it. A setter function should be introduced for the treasury address. Similarly, the insurance fund address is not part of the initialization and there is currently no way to update it. A setter function should be introduced for the insurance fund address.

Recommendation: Ensure all TODOs are resolved.

QSP-3 `depositIterationLimit` not enforced by `submit` function

Severity: *Low Risk*

Status: Fixed

File(s) affected: `DePool.sol`

Description: `_ETH2Deposit` imposes a limit on the maximum number of deposit calls but this condition is not checked in the `_submit` function making the call. The remaining funds would remain buffered until the next `_submit`. There may exist a large discrepancy between deposited ETH and actually staked ETH in such cases.

Recommendation: Limit the amount of deposit using `getDepositIterationLimit` within the `_submit` function.

QSP-4 `setQuorum` may cause the new epoch data to pre-emptively finalize

Severity: *Low Risk*

Status: Fixed

File(s) affected: `DePoolOracle.sol`

Description: `setQuorum` may cause the new epoch data to pre-emptively finalize with stale data. This would happen if `setQuorum` is called in a new epoch before receiving new data from a member (`contributionBitMask` only gets reset once a new data is received through `pushData`).

Recommendation: Ensure that `currentlyAggregatedReportInterval == _getCurrentReportInterval` if trying to finalize from `setQuorum`.

QSP-5 Potential gas-cost issues in `mode` function

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Algorithm.sol`

Description: In `Algorithm.sol`, the documentation says “low gas cost” but the algorithm takes quadratic time in the worst case. Because this worst case is not triggered when most oracle members agree, the impact is unclear. However, one can leverage Solidity having strong cryptographic hashes as a primitive to implement a version of this algorithm where the role of `dataValues` and `dataValuesCounts` is fulfilled by a `mapping`. The running time would become linear.

Recommendation: Consider revising the approach in `Algorithm.sol`.
Update from the Lido team: Postponing; we'll have a comparatively low amount of oracles that will usually converge on a single mode so we can't say it's a risk (even if low) anyway. Oracle submitting different values mean buggy or malicious oracles and gas issues are the least of the problems we can have with that.

QSP-6 Privileged Roles and Ownership

Severity: *Informational*

Status: Fixed

File(s) affected: `StETH.sol`, `DePool.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. The owner may invoke `stop`/`resume`, as well as change configurations through `setFee` and `setOracle`.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.
Update: The contracts have an Aragon-native ACL policy that makes the DAO voting app a privileged user which is described in the documentation.

QSP-7 Missing return values

Severity: *Informational*

Status: Fixed

File(s) affected: `DePool.sol`, `StakingProvidersRegistry.sol`

Description: The following functions are declared to return values but have no explicit return statement:

1. `StakingProvidersRegistry.addStakingProvider` does not return `uint256 id`.
2. `DePool.submit` is declared to return `uint256 StETH` but has no return statement.

Recommendation: Add return statements to these functions.

QSP-8 Unclear access control policy for `transferToVault`

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `DePool.sol`

Description: The function `transferToVault` can currently be called by anyone. It is not clear if this is desirable, or if it should be restricted to privileged addresses.

Recommendation: Clarify if the access-control policy of `transferToVault` is correct.
Update from the Lido team: Fixes planned (should be callable only by the DAO) but not a priority at the moment.

QSP-9 `_ETH2Deposit` staking strategy may possibly lead to increased slashed funds

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `DePool.sol`

Description: The function chooses to increase the stake of the staking provider with the least stake. If this provider has been slashed, their total allocated stake will increase. Thus, new funds may gravitate toward providers that are slashed more often and likely less reliable.

Recommendation: Clarify if this scenario is possible and adjust the code if needed.
Update from the Lido team: In the event of validators going offline, the DAO should take measures to limit staking limit of a node operator.

QSP-10 Potential duplicate reward addresses

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `StakingProvidersRegistry.sol`

Description: There is no check on duplicate reward addresses in `addStakingProvider` and `setStakingProviderRewardAddress`. It is unclear from the specification if it is acceptable to have multiple staking provider register with the same reward address.

Recommendation: Clarify if duplicate addresses in these scenarios are acceptable.
Updates from the Lido team: Duplicates are not acceptable and the DAO should control for them when adding node operators.

QSP-11 `StETH` must be able to upgrade referenced `DePool` contract

Severity: *Undetermined*

Status: Fixed

File(s) affected: `StETH.sol`, `DePool.sol`

Description: `DePool.sol` is incomplete and will likely require an update to enable the `withdraw` functionality. However, the `DePool` contract does not appear to be upgradeable once initialized in `StETH`.

Recommendation: Add a `setPool(address _pool)` function if needed.
Update from the Lido team: The `Lido` (formerly depool) contract address is not malleable but the `Lido` contract itself is upgradeable.

QSP-12 Unclear mechanics of `burn` function

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `StETH.sol`

Description: From the spec, it appears that `StETH` token are intended to be burned upon withdrawal ("Tokens are minted upon deposit and burned when redeemed."), but withdrawal functionality is not implemented. In the current implementation, it appears that `burn` achieves the following:

1. An address can forgo part of its claim on pooled ETH.
2. The total amount of pooled ETH remains the same.
3. The amount of "burned" claims on the pooled ETH gets redistributed evenly across the entire StETH token holder pool.
4. The invariant used for the sharesToBurn calculation is: (Shares address i - sharesToBurn) / (Total shares - sharesToBurn) = (pooledETH address i - amountToBurn) / Total pooledETH

Recommendation: Clarify the functionality of burn and its intended usage.
Update from the Lido team: Burn is used to offset slashings using insurance funds, e.g., if the protocol was slashed by 5 ETH then burn 5 stETH. Will be better documented by the release.

QSP-13 Unclear use of safeTransferFrom

Severity: *Undetermined*

Status: Fixed

Description: wrap and unwrap calls safeTransferFrom on StETH but StETH does not inherit SafeERC20.

Recommendation: Clarify the use of safeTransferFrom here.

QSP-14 removeOracleMember() does not clear the last update

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: LidoOracle.sol

Description: removeOracleMember() only disables the bitmask and does not update the reports stored in gatheredEpochData[lastReportedEpochId].reports.

Recommendation: Consider revising the function through updates to gatheredEpochData[lastReportedEpochId].reports.
Update from the Lido team: The team has developed plans to implement fixes and has a procedure to handle this issue in the meantime. The fix will be implemented as an upgrade after the deployment. The status of this issue can be tracked in the following GitHub page.

For additional information, this issue is exploitable only in case of several (at least quorum-1) removal of oracle members within the same epoch. In such instances, the DAO plans to pause the protocol entirely and apply fixes during the pause. This means that a slightly under quorum number of malfunctioning oracles can force a protocol-wide pause by the DAO.

The Lido team believes that this does not present a tangible risk during the first stage after the protocol deployment. During this stage, the protocol will use only DAO-picked trusted oracles. In the worst case, the DAO plans to pause and rotate oracles one-by-one to resolve the situation.

Automated Analyses

Slither

Slither warns of the following division-before-multiplication issues, which may lead to rounding inaccuracies:

- In DePool._submit(address): toUnbuffer = buffered.div(DEPOSIT_SIZE).mul(DEPOSIT_SIZE).
- In DePool.distributeRewards: tokens2mint = _totalRewards.mul(_getFee()).div(10000) as well as toTreasury = tokens2mint.mul(treasuryFeeBasisPoints).div(10000).
- In DePool.distributeRewards: tokens2mint = _totalRewards.mul(_getFee()).div(10000) as well as toInsuranceFund = tokens2mint.mul(insuranceFeeBasisPoints).div(10000).

Mythril

Mythril reported the following:

1. In DePool.sol, any user can invoke transferToVault.

Adherence to Specification

1. It is not clear why the stETH token is not used for governance instead of a separate token? Adding a rationale for this to the paper might be good.

Update: The Primer document explains the motivation and levers of DAO governance.

Code Documentation

1. There are several TODO links in the README. **Update: resolved.**
2. In Algorithm.sol, a docstring should be added to mode to explicitly describe the return values of the function (specifically the boolean related to unimodality).
3. In CstETH.sol, L65,66: stEth should be cstEth.
4. In setQuorum of DePoolOracle.sol: "QUORUM_WONT_BE_MADE" isn't a good error message for the situation in which quorum == 0.
5. In DePoolOracle.sol L219 should read: "...if quorum is reached **and data is unimodal." Update: resolved.**

Adherence to Best Practices

1. Pushing data in DePoolOracle.sol is the performance bottleneck of the oracle. For performance improvements, consider maintaining a mapping(address => uint256) keeping track of the members to speed up member search. **Update: Acknowledged. Performance improvement is coming but not part of this audit.**

2. In `StETH.sol`, the function `_transfer` should have `require(from != address(0))`.

3. In `CstETH.sol`, the constructor should have `require(_stEth != address(0))`.

4. In `StETH.sol`, the function `getSharesByPooledEth` has an unnecessary second external call to `getTotalControlledEther`, which could be stored in a local variable. **Update: resolved.**

5. In `StETH.sol`, the function parameter naming convention is not consistent -- a leading underscore is used for parameters in only some functions. Further, in docstrings on L192,217: `allowed_` should be `_allowed`; `_spender` should be `spender`.

6. In `Algorithm.sol` L15: `assert` should be `require`.

7. In `getLatestData` of `DePoolOracle.sol`: consider checking that `lastFinalizedReportInterval` is not zero. **Update: resolved.**

8. Why pass an argument to `_tryFinalize` in `DePoolOracle.sol` if it is always `currentlyAggregatedReportInterval`?

9. In `BitOps.sol`, why not have ocurrences of `_bitIndex` be of type `uint8` (a perfect fit)? If the change is made, don't forget to write `uint256(1) << _bitIndex` instead of `1 << _bitIndex`.

10. Regarding `popcnt` in `BitOps.sol`, there are faster algorithms for this, one of which is below (only makes log n computations rather than always 256):

```
function popcnt(uint256 _mask) internal pure returns(uint8 count) {
    uint256 mask = _mask;
    while (mask != 0) {
        count+= mask & 1;
        mask >>= 1;
    }
}
```

1. There should be a check that `_lido != address(0)` in the constructor of `LidoOracle.sol`.

Test Results

Test Suite Results

```
  Contract: Lido
Solidity stack traces only work with Solidity version 0.5.1 or higher.
     ✓ setFee works (201ms)
     ✓ setFeeDistribution works (300ms)
     ✓ setWithdrawalCredentials works (90ms)
     ✓ setWithdrawalCredentials resets unused keys (276ms)
     ✓ pad64 works (134ms)
     ✓ toLittleEndian64 works (43ms)
     ✓ deposit works (805ms)
     ✓ deposit uses the expected signing keys (547ms)
     ✓ deposit works when the first node operator is inactive (251ms)
     ✓ submits with zero and non-zero referrals work (44ms)
     ✓ key removal is taken into account during deposit (573ms)
     ✓ out of signing keys doesn't revert but buffers (405ms)
     ✓ withrawal method reverts (285ms)
     ✓ pushBeacon works (461ms)
     ✓ oracle data affects deposits (657ms)
     ✓ can stop and resume (570ms)
     ✓ rewards distribution works in a simple case (463ms)
     ✓ rewards distribution works (597ms)
     ✓ deposits accounted properly during rewards distribution (479ms)
     ✓ Node Operators filtering during deposit works when doing a huge deposit (1425ms)
     ✓ Node Operators filtering during deposit works when doing small deposits (1738ms)
     ✓ Deposit finds the right operator (694ms)
     ✓ burnShares works (186ms)
    treasury
       ✓ treasury adddress has been set after init
       ✓ treasury can't be set by an arbitary address (104ms)
       ✓ voting can set treasury
       ✓ reverts when treasury is zero address (69ms)
    insurance fund
       ✓ insurance fund adddress has been set after init
       ✓ insurance fund can't be set by an arbitary address (119ms)
       ✓ voting can set insurance fund (39ms)
       ✓ reverts when insurance fund is zero address (49ms)

  Contract: Algorithm
     ✓ mode function works

  Contract: LidoOracle
     ✓ beaconSpec is correct
    Test utility functions:
       ✓ addOracleMember works (421ms)
       ✓ removeOracleMember works (438ms)
       ✓ setQuorum works (297ms)
       ✓ getOracleMembers works (142ms)
       ✓ getCurrentEpochId works (79ms)
       ✓ getCurrentReportableEpochs works (79ms)
       ✓ getCurrentFrame works (108ms)
    When there is single-member setup
      current time: 1606824000 , current epoch: 0
         ✓ reverts when trying to report from non-member (112ms)
         ✓ reportBeacon works and emits event (56ms)
         ✓ reverts when trying to report this epoch again (82ms)
         ✓ reverts when trying to report future epoch
        current time: 1606825920, current epoch: 5
           ✓ reverts when trying to report stale epoch (147ms)
           ✓ reportBeacon works and emits event (56ms)
    When there is multi-member setup (4 members)
      current time: 1606824000 , current epoch: 0
         ✓ reverts when trying to report from non-member
         ✓ reportBeacon works and emits event (165ms)
         ✓ reportBeacon completes only if data is unimodal (611ms)
         ✓ reverts when trying to report this epoch again (265ms)
         ✓ reverts when trying to report this epoch again from the same user (86ms)
         ✓ reverts when trying to report future epoch
        current time: 1606825920, current epoch: 5
           ✓ members can reports to all reportable epochs, the earliest reportable epoch is the last completed, the latest is current (474ms)
           ✓ member removal dont affect other members' data in last reportable epoch, all other reportable epochs will be staled (575ms)
           ✓ member removal removes their data (576ms)
           ✓ tail member removal works (609ms)
           ✓ quorum change triggers finalization of last reported epoch, all other reportable epochs will be staled (387ms)

  Contract: Lido pushBeacon
     ✓ reportBeacon access control (48ms)
    with depositedVals=0, beaconVals=0, bcnBal=0, bufferedEth=0
       ✓ report BcnValidators:0 BcnBalance:0 = no rewards (87ms)
       ✓ report BcnValidators:1 = revert (92ms)
    with depositedVals=0, beaconVals=0, bcnBal=0, bufferedEth=12
       ✓ report BcnValidators:0 BcnBalance:0 = no rewards (78ms)
       ✓ report BcnValidators:1 = revert (83ms)
    with depositedVals=1, beaconVals=0, bcnBal=0, bufferedEth=3
       ✓ initial state before report
       ✓ report BcnValidators:0 BcnBalance:0 = no rewards (75ms)
       ✓ report BcnValidators:2 = revert (82ms)
       ✓ report BcnValidators:1 BcnBalance:31 = no rewards (74ms)
       ✓ report BcnValidators:1 BcnBalance:32 = no rewards (72ms)
    with depositedVals=2, beaconVals=1, bcnBal=30, bufferedEth=3
       ✓ initial state before report (40ms)
       ✓ report BcnValidators:1 BcnBalance:0 = no rewards (77ms)
       ✓ report BcnValidators:1 BcnBalance:1 = no rewards (73ms)
       ✓ report BcnValidators:2 BcnBalance:62 = no reward (79ms)
       ✓ report BcnValidators:1 BcnBalance:31 = reward:1 (75ms)
       ✓ report BcnValidators:2 BcnBalance:63 = reward:1 (77ms)
       ✓ report BcnValidators:3 = revert with REPORTED_MORE_DEPOSITED (92ms)
    with depositedVals=5, beaconVals=4, bcnBal=1, bufferedEth=0
       ✓ report decreased BcnValidators:3 = revert with REPORTED_LESS_VALIDATORS (160ms)

  Contract: NodeOperatorsRegistry
     ✓ addNodeOperator works (304ms)
     ✓ getNodeOperator works (178ms)
```

```
     ✓ setNodeOperatorActive works (662ms)
     ✓ setNodeOperatorName works (285ms)
     ✓ setNodeOperatorRewardAddress works (296ms)
     ✓ setNodeOperatorStakingLimit works (285ms)
     ✓ assignNextSigningKeys works (302ms)
     ✓ assignNextSigningKeys skips stopped operators (275ms)
     ✓ assignNextSigningKeys respects staking limit (282ms)
     ✓ reportStoppedValidators works (659ms)
     ✓ reportStoppedValidators decreases stake (274ms)
     ✓ trimUnusedKeys works (236ms)
     ✓ addSigningKeys works (947ms)
     ✓ rewardAddress can add & remove signing keys (435ms)
     ✓ can view keys (433ms)
     ✓ removeSigningKey works (871ms)
     ✓ distributeRewards works (429ms)

  Contract: StETH
    ERC20 methods
       ✓ info is correct
      zero supply
         ✓ initial total supply is correct
         ✓ initial balances are correct
         ✓ initial allowances are correct (55ms)
         ✓ approve works
         ✓ transfers works with no pooled ehter, balances aren't changed (50ms)
         ✓ balances aren't changed even if total pooled ether increased
      with non-zero supply
         ✓ total supply is correct
         ✓ balances are correct
        transfer
           ✓ reverts when recipient is the zero address
           ✓ reverts when the sender does not have enough balance (45ms)
           ✓ transfer all balance works and emits event
           ✓ transfer zero tokens works and emits event
        approve
           ✓ reverts when spender is zero address
           ✓ approve without any tokens works
          when the spender had no approved amount before
             ✓ approve requested amount works and emits event
            when the spender had an approved amount
               ✓ approve requested amount replaces old allowance and emits event
        transferFrom
           ✓ reverts when recipient is zero address
           ✓ reverts when sender is zero address
           ✓ reverts when amount exceeds allowance
           ✓ reverts if owner has not any tokens
           ✓ transferFrom works and emits events
        increase allowance
           ✓ reverts when spender is zero address
           ✓ increaseAllowance without any tokens works
          when the spender had no approved amount before
             ✓ increaseAllowance with requested amount works and emits event
          when the spender had an approved amount
             ✓ increaseAllowance with requested amount adds it to allowance and emits event
        decrease allowance
           ✓ reverts when spender is zero address
           ✓ reverts when requested amount exceeds allowance
           ✓ reverts when the spender had no approved amount
           ✓ decreaseAllowance without any tokens works
           ✓ decreaseAllowance with requested amount subs it from allowance and emits event
    with non-zero supply
       ✓ stop/resume works (245ms)
       ✓ allowance behavior is correct after slashing (260ms)
      mint
         ✓ mint works (115ms)
         ✓ reverts when mint to zero address
      burn
         ✓ reverts when burn from zero address (48ms)
         ✓ reverts when burn amount exceeds balance
         ✓ burning zero value works (54ms)
         ✓ burning works (redistributes tokens) (83ms)
         ✓ allowance behavior is correct after burning (103ms)
    share-related getters
      with zero totalPooledEther (supply)
         ✓ getTotalSupply
         ✓ getTotalShares
         ✓ getTotalPooledEther
         ✓ sharesOf
         ✓ getPooledEthByShares
         ✓ balanceOf
         ✓ getSharesByPooledEth
      with non-zero totalPooledEther (supply)
         ✓ getTotalSupply
         ✓ getTotalPooledEther
         ✓ getTotalShares
         ✓ sharesOf
         ✓ getPooledEthByShares
         ✓ balanceOf
         ✓ getSharesByPooledEth

  Contract: CstETH
     ✓ has no burn and burnFrom functions (discarded)
    Wrapping / Unwrapping
       ✓ initial balances are correct
       ✓ stETH is set correctly
       ✓ can't wrap zero amount
       ✓ can't wrap more than allowed
       ✓ cant wrap if sender hasn't any stETH (43ms)
      After successful wrap
         ✓ balances are correct
         ✓ can't unwrap zero amount
         ✓ user can't unwrap more than his csteth balance
         ✓ cant unwrap if sender hasn't any cstETH
        Before rewarding/slashing
           ✓ after partial unwrap balances are correct (101ms)
           ✓ after full unwrap balances are correct
           ✓ cstETH allowances isn't changed
          After user2 submission
             ✓ balances are correct
            After successful wrap
               ✓ balances are correct
        After rewarding
           ✓ after partial unwrap balances are correct (107ms)
           ✓ after full unwrap balances are correct
           ✓ cstETH allowances isn't changed
          After user2 submission
             ✓ balances are correct
             ✓ cstETH allowances isn't changed
            After user2 wrap
               ✓ balances are correct
               ✓ after partial unwrap balances are correct (192ms)
               ✓ after full unwrap balances are correct (65ms)
               ✓ cstETH allowances isn't changed
        After slashing
           ✓ after partial unwrap balances are correct (104ms)
           ✓ after full unwrap balances are correct (39ms)
           ✓ cstETH allowances isn't changed
    ERC20 part
       ✓ has a name
       ✓ has a symbol
       ✓ has 18 decimals
      total supply
         ✓ returns the total amount of tokens
      balanceOf
        when the requested account has no tokens
           ✓ returns zero
        when the requested account has some tokens
           ✓ returns the total amount of tokens
      transfer
        when the recipient is not the zero address
          when the sender does not have enough balance
             ✓ reverts
          when the sender transfers all balance
             ✓ transfers the requested amount
             ✓ emits a transfer event
(node:45282) DeprecationWarning: expectEvent.inLogs() is deprecated. Use expectEvent() instead.
          when the sender transfers zero tokens
             ✓ transfers the requested amount
             ✓ emits a transfer event
        when the recipient is the zero address
           ✓ reverts
      transfer from
        when the token owner is not the zero address
          when the recipient is not the zero address
            when the spender has enough approved balance
              when the token owner has enough balance
                 ✓ transfers the requested amount
                 ✓ decreases the spender allowance
                 ✓ emits a transfer event
                 ✓ emits an approval event
            when the token owner does not have enough balance
               ✓ reverts
          when the spender does not have enough approved balance
            when the token owner has enough balance
               ✓ reverts
            when the token owner does not have enough balance
               ✓ reverts
        when the recipient is the zero address
           ✓ reverts
      when the token owner is the zero address
         ✓ reverts
    approve
      when the spender is not the zero address
        when the sender has enough balance
           ✓ emits an approval event
          when there was no approved amount before
             ✓ approves the requested amount
          when the spender had an approved amount
             ✓ approves the requested amount and replaces the previous one
        when the sender does not have enough balance
           ✓ emits an approval event
          when there was no approved amount before
             ✓ approves the requested amount
          when the spender had an approved amount
             ✓ approves the requested amount and replaces the previous one
      when the spender is the zero address
         ✓ reverts
    decrease allowance
      when the spender is not the zero address
        when the sender has enough balance
          when there was no approved amount before
             ✓ reverts
          when the spender had an approved amount
             ✓ emits an approval event
             ✓ decreases the spender allowance subtracting the requested amount
             ✓ sets the allowance to zero when all allowance is removed
             ✓ reverts when more than the full allowance is removed
        when the sender does not have enough balance
          when there was no approved amount before
             ✓ reverts
          when the spender had an approved amount
             ✓ emits an approval event
             ✓ decreases the spender allowance subtracting the requested amount
             ✓ sets the allowance to zero when all allowance is removed
             ✓ reverts when more than the full allowance is removed
      when the spender is the zero address
         ✓ reverts
    increase allowance
      when the spender is not the zero address
        when the sender has enough balance
           ✓ emits an approval event
          when there was no approved amount before
             ✓ approves the requested amount
          when the spender had an approved amount
             ✓ increases the spender allowance adding the requested amount
        when the sender does not have enough balance
           ✓ emits an approval event
          when there was no approved amount before
             ✓ approves the requested amount
          when the spender had an approved amount
             ✓ increases the spender allowance adding the requested amount
      when the spender is the zero address
         ✓ reverts
    _mint
       ✓ rejects a null account
      for a non zero account
         ✓ increments totalSupply
         ✓ increments recipient balance
         ✓ emits Transfer event

Contract: Lido with official deposit contract
   ✓ deposit works (859ms)
   ✓ key removal is taken into account during deposit (581ms)
   ✓ Node Operators filtering during deposit works when doing a huge deposit (1540ms)
   ✓ Node Operators filtering during deposit works when doing small deposits (1910ms)
   ✓ Deposit finds the right operator (743ms)

Contract: Lido: deposit loop iteration limit
   ✓ DAO, node operators registry, token, and pool are deployed and initialized (1094ms)
   ✓ voting adds a node operator with 16 signing keys (192ms)
   ✓ a user submits 20 * 32 ETH (63ms)
   ✓ one can assign the buffered ether to validators by calling depositBufferedEther() and passing deposit iteration limit (179ms)
   ✓ one can advance the deposit loop further by calling depositBufferedEther() once again (307ms)
   ✓ the number of assigned validators is limited by the remaining ether (192ms)
   ✓ a user submits 2 * 32 ETH (49ms)
   ✓ the number of assigned validators is still limited by the number of available validator keys (93ms)
   ✓ depositBufferedEther is a nop if there are no signing keys available (47ms)

Contract: Lido: happy path
   ✓ DAO, node operators registry, token, and pool are deployed and initialized (998ms)
   ✓ voting sets fee and its distribution (88ms)
   ✓ voting sets withdrawal credentials (52ms)
   ✓ voting adds the first node operator (48ms)
   ✓ the first node operator registers one validator (49ms)
   ✓ the first user deposits 3 ETH to the pool (107ms)
   ✓ the second user deposits 30 ETH to the pool (172ms)
   ✓ at this point, the pool has ran out of signing keys
   ✓ voting adds the second node operator who registers one validator (91ms)
   ✓ the third user deposits 64 ETH to the pool (193ms)
   ✓ the oracle reports balance increase on Ethereum2 side (338ms)

Contract: Lido: penalties, slashing, operator stops
   ✓ DAO, node operators registry, token, and pool are deployed and initialized (1007ms)
   ✓ voting sets fee and its distribution (90ms)
   ✓ voting sets withdrawal credentials (52ms)
   ✓ voting adds the first node operator (44ms)
   ✓ the first node operator registers one validator (54ms)
   ✓ the user deposits 32 ETH to the pool (130ms)
   ✓ voting grants first operator right to have one validator
   ✓ new validator doesn't get buffered ether even if there's 32 ETH deposit in the pool (43ms)
   ✓ pushes pooled eth to the available validator (57ms)
   ✓ new validator gets the 32 ETH deposit from the pool
   ✓ first oracle report is taken as-is for Lido (167ms)
   ✓ the oracle reports balance loss on Ethereum2 side (162ms)
   ✓ voting adds the second node operator who registers one validator (93ms)
   ✓ the user deposits another 32 ETH to the pool (185ms)
   ✓ the oracle reports balance loss for the third time (174ms)
   ✓ the oracle can't report less validators than previosly
   ✓ oracle reports profit not making up for previous penalties (211ms)
   ✓ first operator adds a second validator (51ms)
   ✓ voting stops the first operator (51ms)
   ✓ user deposits another 32 ETH to the pool (126ms)
   ✓ oracle reports profit, stopped node operator doesn't get the fee (113ms)
   ✓ voting starts the first operator back (56ms)
   ✓ oracle reports profit, previously stopped node operator gets the fee (128ms)


271 passing (2m)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
0.6.**12/**	100	100	100	100	
CstETH.sol	100	100	100	100	
0.6.**12/interfaces/**	100	100	100	100	
IStETH.sol	100	100	100	100	
All files	100	100	100	100	
File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
0.4.**24/**	93.48	73.91	94.59	93.51	
Lido.sol	91.33	66.18	92.16	91.38	… 305,744,745
StETH.sol	100	95.83	100	100	
0.4.**24/interfaces/**	100	100	100	100	
ILido.sol	100	100	100	100	
ILidoOracle.sol	100	100	100	100	
INodeOperatorsRegistry.sol	100	100	100	100	
ISTETH.sol	100	100	100	100	
IValidatorRegistration.sol	100	100	100	100	
0.4.**24/lib/**	100	83.33	100	100	
MemUtils.sol	100	50	100	100	
Pausable.sol	100	100	100	100	
0.4.**24/nos/**	98.3	78.26	100	98.46	
NodeOperatorsRegistry.sol	98.3	78.26	100	98.46	290,380,597
0.4.**24/nos/test_helpers/**	85.71	100	83.33	85.71	
ERC20Mock.sol	50	100	50	50	15
PoolMock.sol	100	100	100	100	
0.4.**24/oracle/**	98.73	78.05	96.15	98.78	
Algorithm.sol	100	90	100	100	
BitOps.sol	100	83.33	100	100	
LidoOracle.sol	98.37	75.76	95.45	98.39	434,465
0.4.**24/oracle/test_helpers/**	100	100	100	100	
TestAlgorithm.sol	100	100	100	100	
0.4.**24/template/**	0	0	0	0	
Imports.sol	100	100	100	100	
LidoTemplate.sol	0	0	0	0	… 217,218,219
All files	**87.81**	**74.64**	**92.16**	**86.57**	

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

```
4a3179f8a2511171c47e3e0abf9ccf3b9285782f3f9b5a6e4d03ff2c59376677 ./contracts/0.6.12/CstETH.sol
c88be98471b066873c9b84c5e2ff1e894eb627cd96dd4de7ebcc39e98f8b3b3b ./contracts/0.6.12/mocks/CstETHMock.sol
94ac75e7d5dae639dbae1bd6cebfd6735f7a9595b29d00149a016e891cf9cb45 ./contracts/0.6.12/mocks/StETHMockERC20.sol
e7c019b2a86965a31a3159e610fa5ea33511dbdba86001b5d6842484d3b4824e ./contracts/0.6.12/interfaces/IStETH.sol
3737e162e929373397c548193d7a258ae59c510ad4ce78b8d80a1cd0a94f4183 ./contracts/0.6.11/deposit_contract.sol
840f53e43cb2a1abf1563842a04c65fde43bfaf6ecfd9a940f1d01c6d9f46c1b ./contracts/0.4.24/Lido.sol
c677caf7bea08d3934ceb8aeabf34809fc7b59ae3cc32aee8a4c4faa18372cd8 ./contracts/0.4.24/StETH.sol
bbe6f404b7c03dc34348829ad0e9aad3003f055ae8d4bddbb443c07c352c7edc ./contracts/0.4.24/test_helpers/LidoPushableMock.sol
a3d89415e788dea772be9dc4d17ea8932ff822829c94c72a43e152a36fac5764 ./contracts/0.4.24/test_helpers/OracleMock.sol
6c5558f88fde60a93e65c2ba83bb74e9b73e605b8e82e53be40505e884413082 ./contracts/0.4.24/test_helpers/StETHMock.sol
d53dab9d2659a251369c05d1591baf2d94720b4e0454ab39ef2c91593f049c7d ./contracts/0.4.24/test_helpers/TestLido.sol
600eb6b6974e74605ed08bf1ac753dca5011290965e4837ee0db1cc5db552bf4 ./contracts/0.4.24/test_helpers/TestLidoOracle.sol
55a9025343099d6855288a7a7dbcb01a4d350f6cfe85964dbd2038f342aa15fc ./contracts/0.4.24/test_helpers/ValidatorRegistrationMock.sol
11b31e426d812eb3d0b513973de9b0af04744cd94e81f5ae530502804110867d ./contracts/0.4.24/test_helpers/VaultMock.sol
afec6fed3681848d8a8fb29fbb3de31ce88aaa4c9f01e59a24df5a79461b29d4 ./contracts/0.4.24/template/Imports.sol
bc3760e8b6dabb64950573a211bfc344a5f0030ac103584144721c042949cca4 ./contracts/0.4.24/template/LidoTemplate.sol
dc77cbd528763832650c22d497e98106618cf1a786cee1163f0093add9c7feaf ./contracts/0.4.24/oracle/Algorithm.sol
5f5f39b27c3bf15e0196883b410c0798b82778620a20f803592514d7b52c4ec0 ./contracts/0.4.24/oracle/BitOps.sol
bf1ee48132a2a81f3b6025acba3c5eb411cb16f018ca6a035697cc8debd2f4b7 ./contracts/0.4.24/oracle/LidoOracle.sol
3fe5866f2a439038ebe2bab443b7a576d3e7895f9d9af9e949f8472a6cd5f0a1 ./contracts/0.4.24/oracle/test_helpers/TestAlgorithm.sol
9c8132d7eab82a39d2d4cc40139fcdef617f21cc242698e87be2ae8c6938ce3a ./contracts/0.4.24/nos/NodeOperatorsRegistry.sol
1acf565b253a65b6aa0a87a2a69cb8909096a1804ae7abebb2dd23a901e247fe ./contracts/0.4.24/nos/test_helpers/ERC20Mock.sol
f5e011c929af6f189b7aefa539e5eafb8a07a785dd9d52ab706c5d8721c3169c ./contracts/0.4.24/nos/test_helpers/PoolMock.sol
8012d1f3fd21ab8bb47515e554c9df31ac22dd822330fc0ecbc8650032489f13 ./contracts/0.4.24/lib/MemUtils.sol
7bb3cd46af81c9cece011fe84e1c8232de6a75649ea53c82bef5d792bc588fdc ./contracts/0.4.24/lib/Pausable.sol
ef53c70fb1e9f0150d892580295889ccdf9c69957877d0deee192eb2927d4dbd ./contracts/0.4.24/interfaces/ILido.sol
9f64120e9b5180dca4c0318c80603734b6ef35cfe5c5c3015cf6d89100db324a ./contracts/0.4.24/interfaces/ILidoOracle.sol
7d2a41e4c7e0694f198cb50e3a8526775776e890f4132702d0ed7399f0aaa2e8 ./contracts/0.4.24/interfaces/INodeOperatorsRegistry.sol
bd001de57aa581277395d2ed9c976d6f00d7517f5c435f9d7971815ebdadbd52 ./contracts/0.4.24/interfaces/ISTETH.sol
e779f53146096733f8e9d99089a846d37149a894d555c1f54fbdb1a4a4b8e921 ./contracts/0.4.24/interfaces/IValidatorRegistration.sol
```

Tests

```
2fc01fe7e4035fefa5bb9922122dc71ed5db648379ea39b91cea8ee52d3f271a ./test/deposit.test.js
6d6473ac8426032d077dca6754800d4932ad024a43620ef91c011dbf2b6200aa ./test/scenario/lido_deposit_iteration_limit.js
7f38847a642eb97f35068f5c4d7f0cf4e505934f216b85ff48822f021ed675fd ./test/scenario/lido_happy_path.js
40d2e0fee63c2bdaab899a76b7b9adb31a64e4e6ba8eea75f86fb0fc50ab9f46 ./test/scenario/lido_penalties_slashing.js
2a987112a102141675b3d9381d3b978e53cb2f1e1dd8865a6407c7c89f1f5abe ./test/scenario/helpers/deploy.js
15cb701a9497834d33921b851df26371daa81a53274bba3ed3f6d2c99421e032 ./test/helpers/utils.js
0aeb3fb964d0bcdf246dd5ba260de25d4242a0deb677eb7788b395cd193aa7ba ./test/0.6.12/csteth.test.js
b496f8bd829a074403e5f9ae641499308d5e69f0fa326f407d73eda67771b8a4 ./test/0.6.12/helpers/ERC20.behavior.js
38e5fbdb26fe9443bb631e28a328a29c5c891007502be77257797cae78d556af ./test/0.4.24/lido.test.js
321d1a2b1166f9037d319a5b6f4f960f32dde2e10682d1408f2620ab01cbf651 ./test/0.4.24/lidooracle.test.js
29225523a8ef3c305fedbc5caaf017c29846d2360262e20d128ea711576cbb3b ./test/0.4.24/lidoPushBeacon.test.js
7ed603b7df40f2f5fbc03df3e8049c8296279c62beb145c8abb831cf8ae5a954 ./test/0.4.24/node-operators-registry.test.js
cdc9e3db220345c57f45023947a7730f10fc8cbd60090df5a385c4b35d6d2044 ./test/0.4.24/steth.test.js
f1fb13f5b9667a0ff587eda8b451f265decd6c816c95d65adb17b165693c277a ./test/0.4.24/helpers/dao.js
22a1fe036e2f78278bf17ec988e27f667e9f769f842abe3951f9c653d0971b36 ./test/0.4.24/helpers/permissions.js
```

Changelog

- 2020-11-12 - Initial report
- 2020-12-04 - Updated report based on commit `d4171a1`
- 2020-12-11 - Updated report based on commit `ad4b2f6`

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.