Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

September 15th 2022 — Quantstamp Verified

Covalent Operational Staking Contract Part 2

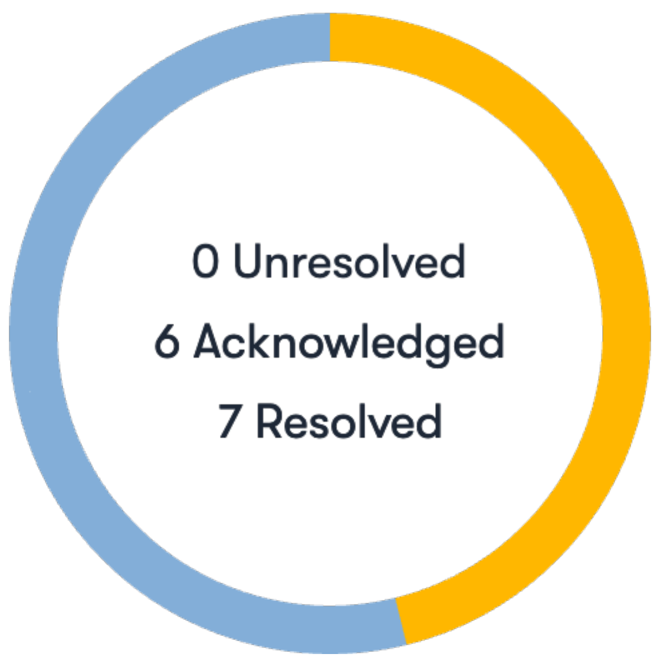
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Ethereum Staking Contract
Auditors	David Knott, Senior Research Engineer Fatemeh Heidari, Security Auditor Bohan Zhang, Auditing Engineer
Timeline	2022-07-12 through 2022-08-02
EVM	Grey Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Manual Review
Specification	None
Documentation Quality	High
Test Quality	High
Source Code	

Repository	Commit
covalenthq/bsp-staking	ab5b33b
covalenthq/bsp-staking	13d3821
covalenthq/bsp-staking	94e16e5

Total Issues	**13**	(7 Resolved)
High Risk Issues	**1**	(1 Resolved)
Medium Risk Issues	**2**	(2 Resolved)
Low Risk Issues	**5**	(2 Resolved)
Informational Risk Issues	**4**	(1 Resolved)
Undetermined Risk Issues	**1**	(1 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).
Fixed	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

After initial audit: Quantstamp has performed an audit of Covalents' `OperationalStaking` contract. Notably, we found a high severity issue in which a `validator` is able to overwrite a `delegate`'s staking information. The audit resulted in a total of `11` findings and an additional `11` best practice violations, described below. We recommend that all issues reported in this document be addressed.
After re-audit: Quantstamp has performed a re-audit of Covalents' `OperationalStaking` contract. We found two additional issues, the first allows `validator`'s to delete their own validator information and has already been fixed. The second issue is due to the copying of an unbounded array which could become unexecutable and is as of now unfixed. We recommend that the remaining unresolved issues be addressed.
After re-audit: Quantstamp has performed a re-audit of Covalents' `OperationalStaking` contract. During the re-audit the Quantstamp team discussed how to best prevent `validator` transfers from becoming unexecutable with the Covalent team. All issues have been resolved.

ID	Description	Severity	Status
QSP-1	Validator Can Overwrite Delegator's Staking Information	High	Fixed
QSP-2	Validator Information Deleted on Self Role Transfer	Medium	Fixed
QSP-3	Denial of Service From List Iteration and Copying	Medium	Fixed
QSP-4	Superuser Configuration Leads to Invariant Violations	Low	Acknowledged
QSP-5	Missing Input Validation	Low	Mitigated
QSP-6	Use of Block Numbers to Track Time	Low	Acknowledged
QSP-7	Redeems Under Reward Redeem Threshold Allowed	Low	Fixed
QSP-8	Privileged Roles and Ownership	Low	Acknowledged
QSP-9	Same Address Can Be Added as Validator Multiple Times	○ Informational	Acknowledged
QSP-10	Ownership Can Be Renounced	○ Informational	Fixed
QSP-11	Staked and Reward Amounts Under Threshold Are Locked	○ Informational	Acknowledged
QSP-12	Configurable Disable Blocknumbers Allow for Instant Withdrawals	○ Informational	Acknowledged
QSP-13	Unchecked Math Could Lead to Underflow	? Undetermined	Fixed

Quantstamp Audit Breakdown

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

DISCLAIMER:
If the final commit hash provided by the client contains features that are not in scope of the audit or a re-audit, those features are excluded from the consideration in this report.

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 v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither contracts/OperationalStaking.sol --solc-remaps @openzeppelin/=node_modules/@openzeppelin/`

Findings

QSP-1 Validator Can Overwrite Delegator's Staking Information

Severity: High Risk

Status: Fixed

File(s) affected: `OperationalStaking.sol`

Description: `OperationalStaking`'s `setValidatorAddress` function transfers the `msg.sender`'s `validator`'s `stakings` and `unstakings` fields to `newAddress`. However, `setValidatorAddress` does not check whether `newAddress` already has existing `stakings` and `unstakings` values. If a validator calls `setValidatorAddress` with `newAddress` equal to the `address` of an existing `delegate` then the `delegate`'s `stakings` and `unstakings` information will be overwritten.

Recommendation: Modify `setValidatorAddress` to check that `newAddress`'s `staking staked` amount is zero and `unstaking` array has a length of zero.

QSP-2 Validator Information Deleted on Self Role Transfer

Severity: Medium Risk

Status: Fixed

Description: `OperationalStaking`'s `setValidatorAddress` function transfers a `validator`'s address to `newAddress` by moving a `validator`'s `stakings` and `unstaking` fields to the `newAddress`. It then deletes `stakings` and `unstaking` from the old `validator`'s address. However, `setValidatorAddress` does not check whether the old `validator`'s address is the same as `newAddress`. If they are the same then the `stakings` and `unstaking` fields of the `validator` being transferred will be deleted, resulting in a loss of funds.

Recommendation: Modify `OperationalStaking`'s `setValidatorAddress` to check that `newAddress` is not the address of the `validator` being transferred.

QSP-3 Denial of Service From List Iteration and Copying

Severity: Medium Risk

Status: Fixed

Description: The gas costs of iterating or copying storage arrays is expensive as an `SSLOAD` (`800` gas) and an `SSTORE` (`20,000` gas) in the latter case is required for each element. By default dynamic lists have no maximum size and so the maximum cost of iterating through or copying an array is unpredictable. In the event that performing either of the aforementioned operations exceeds Ethereum's block gas limit, then calling said operations will result in a transaction reversion. It will also make code executed after said operations over the block gas limit to become unexecutable. Array iteration or copying are performed in the following places in `OperationalStaking`:

1. In `setValidatorAddress` where `v.unstakings` is iterated through to transfer `unstakings` to the new `validator` address.
2. In `getDelegatorMetadata` where `v.unstakings` is iterated through to collect a `delegator`'s metadata.

Recommendation: Modify `setValidatorAddress`'s copying of `unstakings` to `newAddress` to be performable over multiple Ethereum transactions so that though transferring a validator's address may become expensive it will never be impossible. Another approach to consider is adding an upper bound to the length of `unstakings` and to add a mechanism for `unstakings`'s owners to remove `unstakings` entries after they have been processed. Additionally, considering modifying `getDelegatorMetadata` to use pagination similar to `getValidatorsMetadata`.

Update: The Covalent team added an upper bound to the length of `unstakings` prior to it be copied, protecting against an out of gas error. They also added the ability for `validator`'s to opt-out of transferring `unstakings`, in which case a `validator`'s `unstakings` would associated with their old address instead of `newAddress`. They also stated that `getDelegatorMetadata` is only meant to be called off-chain and so its gas usage is not a concern.

QSP-4 Superuser Configuration Leads to Invariant Violations

Severity: Low Risk

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: `OperationalStaking` has an `owner` role that is able to configure `maxCapMultiplier` and `validatorMaxStake`. The decrease of `maxCapMultiplier` and `validatorMaxStake` can lead to situations where a `validator`'s `valueStaked` or `delegation` amounts are higher than the protocols upper bounds.

Recommendation: Track the the maximum amount a single `validator` has `staked` and `delegated`. Then modify `setMaxCapMultiplier` and `setValidatorMaxStake` to check whether the new values being set are less than the tracked maximums.

Update: The Covalent team stated that they decided not to add validation checks to `setMaxCapMultiplier` and `setValidatorMaxStake`.

QSP-5 Missing Input Validation

Severity: Low Risk

Status: Mitigated

File(s) affected: OperationalStaking.sol

Description: Many of OperationalStaking's function's inputs are either are missing sufficient validation. This can lead to the OperationalStaking ending up in unexpected states. The following function inputs are missing validation:

1. initialize does not check that:
   - cqt is a contract address.
   - dCoolDown is greater than zero.
   - vCoolDown is greater than zero.
   - maxCapM is greater than zero.
   - vMaxStake is greater than zero.
2. setStakingManagerAddress does not check that:
   - newAddress is a contract address.
3. setMaxCapMultiplier does not check that:
   - maxCapMultiplier is small enough that it will not cause an overflow when it is multiplied by validator's staked amounts on L303 and L332.
4. redeemCommission does not check that:
   - amount is greater than zero.
5. rewardValidators does not check that:
   - all ids are in _validators.
6. setValidatorAddress does not check that:
   - newAddress is not addresss(0).

Recommendation: Add the missing validation enumerated above or add documentation explaining why the lack of validation is intended.

Update: The Covalent team fixed issues 3.4, and 3.6 and acknowledged issues 3.1, 3.2, 3.3, and 3.5 with the following comments:
1. OperationalStaking is already deployed and init has already been called on it. Given that init is only intended to be called once during the initial deployment there is no need to add input validation to init as it will not be called again.
2. The stakingManager is intended to be able to be either a smart contract or an externally owned account (EOA).
3. Validation setMaxCapMultiplier validation checks were intentionally omitted.
5. ProofChain is responsible for ensuring that all ids are in _validators and adding the same check to rewardValidators would be redundant. Furthermore, given that rewardValidators is intended to be called multiple times per day, the gas increase for adding an ids inclusion check to rewardValidators was deemed not worth it.

QSP-6 Use of Block Numbers to Track Time

Severity: Low Risk

Status: Acknowledged

File(s) affected: OperationalStaking.sol

Description: OperationalStaking uses block numbers to track validatorCoolDown and delegatorCoolDown time periods. However, Ethereum block times are subject to change. Given that OperationalStaking has no functionality to update validatorCoolDown and delegatorCoolDown, cooldown times could drift substantially from the 6 month and 28 day time periods that were initially intended.

Recommendation: Modify validatorCoolDown and delegatorCoolDown to be block timestamps. Even though timestamps are manipulatable neither of the cooldowns require time accuracy of greater than 15 seconds which is the maximum amount of time manipulation possible without creating a hardfork. If block numbers are kept, allow them to be configured so that they can be updated based on Ethereum block time changes.

Update: The Covalent team states that if Ethereum's actual block times differ to greatly from OperationalStaking's expected block times then they will be replaced with timestamps. Not modifying cooldowns was an intentional business decision, as any modification to them could negatively impact user trust.

QSP-7 Redeems Under Reward Redeem Threshold Allowed

Severity: Low Risk

Status: Fixed

File(s) affected: OperationalStaking.sol

Description: OperationalStaking's _redeemRewards function checks whether the reward amount being redeemed is greater than or equal to REWARD_REDEEM_THRESHOLD on L437 inside the redeemAll conditional but does not perform an equivalent check when a staker specified amount is being redeemed. This leads to redeems being processed that are less than the REWARD_REDEEM_THRESHOLD.

Recommendation: Modify _redeemRewards to check whether user specified redeem amounts are greater than or equal to the REWARD_REDEEM_THRESHOLD. If user specified redeem amounts are not meant to be subject to the REWARD_REDEEM_THRESHOLD add technical documentation stating this.

QSP-8 Privileged Roles and Ownership

Severity: Low Risk

Status: Acknowledged

File(s) affected: OperationalStaking.sol

Description: Smart contracts will often have owner variables to designate an address with special privileges. The OperationalStaking contract has an owner who is able to:

1. Renounce his role and disable all the following listed actions by calling renounceOwnership.
2. Transfer the ownership to another address by calling transferOwnership.
3. Set or change the stakingManager by calling setStakingManagerAddress.

4. Transfer `CQT` from the owner to the contract for reward allocation by calling `depositRewardTokens`.
5. Transfer reward `CQT` from the contract to the owner by calling `takeOutRewardTokens`.
6. Updates `validator`'s `maxCapMultiplier` that determines how many tokens can be delegated by calling `setMaxCapMultiplier`.
7. Update `validator`'s `validatorMaxStake` by calling `setValidatorMaxStake`.
8. Update `validator`'s `comissionRate`s by calling `setValidatorCommissionRate`.

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 Covalent team added documentation to their `README.md` explaining the abilities of the `owner` role. However, no documentation or code changes were made that mitigate or remove the `owner` roles centralization.

QSP-9 Same Address Can Be Added as Validator Multiple Times

Severity: *Informational*

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: `OperationalStaking`'s `addValidator` function does not check whether the new `validator`'s `address` already exists in the list of validators. This makes it possible for a single `validator` to bypass `validatorMaxStake` and the `delegationMaxCap` constraints.

Recommendation: Modify `addValidator` to check whether a `validator address` has already been added. Potential solutions would be to use a mapping instead of an array or to use an extra mapping that tracks whether an address has previously been added as a validator.

Update: The Covalent team states that `addValidator` being able to add the same `address` multiple times was an intentional business decision and they added said decision to their `README.md`.

QSP-10 Ownership Can Be Renounced

Severity: *Informational*

Status: Fixed

File(s) affected: `OperationalStaking.sol`

Description: `OperationalStaking` inherits from OpenZeppelin's `Ownable` contract, which contains a `renounceOwnership` function. If the owner renounces their ownership, all `Ownable` contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

Recommendation: Double check if this is the intended behavior. If it is not, `override renounceOwnership` and disable it.

QSP-11 Staked and Reward Amounts Under Threshold Are Locked

Severity: *Informational*

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: Users may interact with the `OperationalStaking` with a relatively small amounts for a variety of reasons such as testing. In the `Readme.md` file, there is no mention made that `staking` and `unstaking` amounts must exceed the `REWARD_REDEEM_THRESHOLD`. The lack of documentation could cause users to accidentally lock amounts under `REWARD_REDEEM_THRESHOLD` in `OperationalStaking`.

Exploit Scenario:

1. User stakes 10**8 + 2 tokens.
2. User unstakes 10**8 + 1 tokens.
3. User is unable to unstake the 1 token remaining.

Recommendation: Add end-user documentation explaining that any staked funds and rewards less than `REWARD_REDEEM_THRESHOLD` will be locked.

Update: The Covalent team added documentation to their `README.md` stating that any staked funds and rewards less than `REWARD_REDEEM_THRESHOLD` will be locked. However, though the risk is better documented there's still a risk of users unexpectedly having funds locked.

QSP-12 Configurable Disable Blocknumbers Allow for Instant Withdrawals

Severity: *Informational*

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: The `Readme.md` mentions that `validator`s need to wait `180` days and delegates need to wait `28` to withdraw staked tokens. However, if the `stakingManager` sets a `validator`'s `disabledAtBlock` to `1` then the `validator` and its `delegates` can instantly `unstake` and transfer out all tokens without needing to wait the documented cooldown period.

Recommendation: Add end-user documentation stating why `disabledAtBlock` should be configured by the `stakingManager`. If it does not need to be modify `disableValidator` to set `disabledAtBlock` programmatically.

Update: The Covalent team states that allowing `disabledAtBlock` to be configured by the `stakingManager` contract was an intentional business decision and that it is intentionally set to `1` every time a new validator is added. They also stated that the ability for `stakingManager` to arbitrarily set `disabledAtBlock` has been intentionally left in to protect against future contract upgrades' edge cases.

QSP-13 Unchecked Math Could Lead to Underflow

Severity: *Undetermined*

Status: Fixed

File(s) affected: `OperationalStaking.sol`

Description: The `unchecked` keyword tells the Solidity compiler not add overflow and underflow checks. `OperationalStaking`'s `_redeemRewards` function subtracts `validatorSharesRemove` from `v.totalShares` and `s.shares` in unchecked blocks. However, due to division truncation converting between tokens and shares `validatorSharesRemove` could possibly be greater than `v.totalShares` or `s.shares` which would cause an underflow.

Recommendation: Either remove the `unchecked` blocks on `L448` and `L451` or add comments explaining why underflow protection is not necessary.

Automated Analyses

Slither

Slither reported 43 results on `OperationalStaking`. We filtered out the false positives and included the rest of the findings in the report.

Code Documentation

1. **Not Implemented:** Add Ethereum Natural Language Specification Format (NatSpec) to increase `OperationalStaking`'s readability.

2. **Acknowledged:** `REWARD_REDEEM_THRESHOLD` is used not only as a threshold for rewards but also for `staking` and `unstaking`. `REWARD_REDEEM_THRESHOLD`'s name should be changed to reflect that it is also a threshold for `staking` and `unstaking` to increase code readability.

   Update: The Covalent team states that the `OperationalStaking` contract is already deployed and so they cannot modify `constant`s.

3. **Fixed:** On `L341` `inconsistencies` is misspelled as `inconsisencies`.

Adherence to Best Practices

1. **Fixed:** In `OperationalStaking`'s `rewardValidators` function `rewardAmount` is defined but never used. Remove this variable, or use it in the next line instead of recalculating `amount - commissionPaid`.
2. **Acknowledged:** Change `DIVIDER`'s and `disabledAtBlock` type from `uint256` to `uint128` to save gas.
   Update: The Covalent team states that the `OperationalStaking` contract is already deployed and so they cannot add `constant`s and that they do not want to make any more modifications than absolutely necessary to `OperationalStaking`'s storage.
3. **Acknowledged:** Move `Validator`'s `_address` variable to the bottom of the declaration to save gas.
   Update: The Covalent team states that the `OperationalStaking` contract is already deployed and they do not want to make any more modifications than absolutely necessary to `OperationalStaking`'s storage.
4. **Acknowledged:**`OperationalStaking` is meant to be used as an implementation contract behind a proxy. To prevent the `OperationalStaking` implementation contract from being initialized with arbitrary values add the following code:

   ```
   contructor() initializer {}
   ```

   Update: The Covalent team states that the `OperationalStaking` contract is already deployed and so adding a `constructor` would have no impact on the execution of `OperationalStaking`.
5. **Fixed:** Instead of calling `_sharesToTokens(sharesAdd, v.exchangeRate)` on L299, use the token `amount` to save gas.
6. **Fixed:** The subtraction of `amount` from `us.amount` on `L373` can be `unchecked` as `us.amount` is already checked to be greater than or equal to `amount` in the `require` statement on `L371`.
7. **Fixed:** On `L442` `stakeRewardToRedeem` is unnecessarily initialized. `amountToRedeem` should be used directly in `stakeRewardToRedeem`'s place.
8. **Fixed:** `comissionRewardToRedeem` is declared on `L443` but is never set. It should either be set or removed to save gas and increase code readability.
9. **Fixed:** The subtraction of `amount` from `v.commissionAvailableToRedeem` on `L471` can be `unchecked` as the `require` statement on `L470` protects against integer underflow.
10. **Fixed** Avoid repeatedly accessing the same `struct` in mappings. This will increase `OperationalStaking`'s readability and save gas. For example in `addValidator`:

```
Validator storage v = _validators[N];
v._address = validator;
v.exchangeRate = uint128(DIVIDER); // make it 1:1 initially
v.commissionRate = commissionRate;
v.disabledAtBlock = 1; // set it to 1 to indicate that the validator is disabled
```

can be replaced by:

```
Validator storage v = _validators[N];
v._address = validator;
v.exchangeRate = uint128(DIVIDER); // make it 1:1 initially
v.commissionRate = commissionRate;
v.disabledAtBlock = 1; // set it to 1 to indicate that the validator is disabled
```

Similar optimization could be applied to `redelegateUnstaked`.

Test Results

Test Suite Results

Tests were run by adding a JSON-RPC endpoint url to `hardhat.config.js` and entering `npx hardhat test`. All 304 tests were passing.

```
All together
     ✓ Should redeem, stake, unstake and recover correct # of tokens.

  Ownership
     ✓ Should return owner address same as signer.
     ✓ Should access depositRewards, takeOutRewardTokens, setMaxCapMultiplier by owner.
     ✓ Should not access depositRewards, takeOutRewardTokens, addValidator by not owner.
     ✓ Should access rewardValidator, addValidator by proofChain.
     ✓ Should not access rewardValidator, addValidator by not proofChain.
     ✓ Should not access internal functions.

  Add Validator
     ✓ Should change validators number.
     ✓ Should emit event  with correct validator and commission rate.
     ✓ Should add validator with correct commission rate.
     ✓ Should add validator with correct address.
     ✓ Should revert when validator address is 0.
     ✓ Should revert when commission rate is 100%

  Deposit reward Tokens
     ✓ Should change balance of the contract and the owner.
     ✓ Should change rewardPool.
     ✓ Should emit RewardTokensDeposited event with correct amount.
     ✓ Should revert with wrong inputs.

  Disable validator
     ✓ Should not be able to call stake after validator got disabled.
     ✓ Should emit event with correct validator and disabled block.
     ✓ Should return correct disabled block.
     ✓ Should revert when trying to disable invalid validator.
     ✓ Should revert when disabled at block is 0 .

  Enable validator
     ✓ Should be able to call stake after validator got enabled after being disabled.
     ✓ Should emit event with correct validator and disabled block.
     ✓ Should return correct disabled block.
     ✓ Should revert when enabling invalid validator id.

  Get all validators metadata
     ✓ Should return correct validator addresses
     ✓ Should return correct # of tokens staked
     ✓ Should return correct # of tokens delegated
     ✓ Should return correct disabled at block number

  Get delegator metadata
     ✓ Should return correct # of tokens staked by validator
     ✓ Should return correct # of tokens staked by delegator
     ✓ Should return correct amounts of unstakings
     ✓ Should return correct end epochs of unstakings
     ✓ Should revert when validator id is invalid

  Get metadata
     ✓ Should return correct number of validators.
     ✓ Should return correct CQT address.
     ✓ Should return correct staking manager address.
     ✓ Should return correct reward pool.
     ✓ Should return correct delegator cool down .
     ✓ Should return correct validator cool down .
     ✓ Should return correct max cap multiplier.
     ✓ Should return correct validator max stake.

  Get validator staking data
     ✓ Should return correct # of tokens staked
     ✓ Should return correct # of tokens delegated

  Get validator metadata
     ✓ Should return correct validator address
     ✓ Should return correct validator commission rate
     ✓ Should return correct # of tokens staked
     ✓ Should return correct # of tokens delegated
     ✓ Should return correct disabled at block number
     ✓ Should revert when validator id is invalid

  Get validators from start id to end id metadata
     ✓ Should return correct validator addresses
     ✓ Should return correct # of tokens staked
     ✓ Should return correct # of tokens delegated
     ✓ Should return correct disabled at block number
     ✓ Should revert with invalid end id
     ✓ Should revert with invalid start and end ids

  Get validator staking data
     ✓ Should return correct # of tokens staked
     ✓ Should return correct # of tokens delegated
     ✓ Should revert when validator id is invalid

  Initialize contract
     ✓ Should emit Initialized event with correct args.

  Recover Unstaking
     ✓ Should revert when recover invalid unstaking
     ✓ Should revert when recover greater than staking
     ✓ Should emit event when recovered unstake successfully
     ✓ Should revert when try to recover the same unstake second time
     ✓ Should not change contract balance

  Redeem All Rewards
     ✓ Should emit redeem reward event with correct number of rewards when there are no delegators
     ✓ Should emit redeem reward event with correct number of rewards when there are delegators
     ✓ Should change balances of the contract and delegator
     ✓ Should revert with nothing to redeem
     ✓ Should revert with invalid beneficiary

  Redeem Commission
     ✓ Should CommissionRewardRedeemed event with correct number of rewards when there are no delegators
     ✓ Should emit CommissionRewardRedeemed event with correct number of rewards when there are delegators
     ✓ Should change balances of the contract and delegator
     ✓ Should revert with nothing to redeem
     ✓ Should revert with invalid beneficiary when trying to redeem some commmission
     ✓ Should revert with invalid beneficiary when trying to redeem all commmission
     ✓ Should revert with invalid validator when trying to redeem
     ✓ Should revert with invalid validator when trying to redeem
     ✓ Should revert when delegator trying to redeem some commission
     ✓ Should revert when delegator trying to redeem all commission

  Redeem Rewards
     ✓ Should revert when requested amount 0
     ✓ Should revert when requested amount is too high
     ✓ Should revert when trying to redeem from invalid validator
     ✓ Should revert when redeem amount is too small

  Redelegate Unstaked
     ✓ Should redelegate partially and emit Redelegated and Staked events
     ✓ Should redelegate fully and emit event
     ✓ Should not be able to redelegate the same unstake fully twice
     ✓ Should change number of staked tokens under new validator
     ✓ Should revert when redelegating with enabled validator
     ✓ Should revert when validators attempt to redelegate
     ✓ Should revert when redelegate greater than unstake
     ✓ Should revert when redelegating from invalid validator
     ✓ Should revert when redelegating invalid unstaking
     ✓ Should not change contract balance
     ✓ Should revert when redelegating from enabled validator that was disabled

  Set max cap multiplier
     ✓ Should not change the owner if owner is renounced.

  Reward validator
     ✓ Should change reward pool
     ✓ Should commission available to redeem
     ✓ Should emit Rewarded event with correct validatorId, commission paid and amount emitted
     ✓ Should emit Rewarded failed due to low pool event with correct validatorId and amount
     ✓ Should emit Rewarded failed due to zero stake  event with correct validatorId and amount
```

```
    ✓ Should revert when given uneven number of ids and reward amounts

  Set max cap multiplier
    ✓ Should change max cap multiplier.
    ✓ Should emit StakingManagerAddressChanged event with correct address.
    ✓ Should be able to delegate more if multiplier increases and should revert when attempted to delegate above max cap.
    ✓ Should revert if set to 0.

  Set staking manager address
    ✓ Should change staking manager address.
    ✓ Should emit StakingManagerAddressChanged event with correct address.
    ✓ Should revert when set to zero address.

  Set validator address
    ✓ Should change staking validator address.
    ✓ Should transfer rewards.
    ✓ Should transfer stakings.
    ✓ Should transfer unstakings.
    ✓ Should emit ValidatorAddressChanged event with correct address.
    ✓ Should not access setValidatorAddress by not validator.
    ✓ Should revert when the new address is 0.
    ✓ Should revert when the new address is the old one.
    ✓ Should revert when the validator id is invalid.
    ✓ Should transfer unstakings when the max amount is used.
    ✓ Should revert when more than the max amount of unstakings is used.
    ✓ Should transfer and merge rewards.
    ✓ Should transfer and merge stakings.
    ✓ Should transfer and merge unstakings.
    ✓ Should not transfer and merge unstakings.

  Set validator commission rate
    ✓ Should change validator commission rate.
    ✓ Should emit ValidatorCommissionRateChanged event with correct validator id and amount.
    ✓ Should emit correct amount of validator commission rewards.
    ✓ Should emit correct amount of delegator rewards.
    ✓ Should revert with invalid validator id.
    ✓ Should revert if set to >= 10^18.

  Set validator max stake
    ✓ Should change validator max stake amount.
    ✓ Should emit ValidatorMaxCapChanged event with correct amount.
    ✓ Should revert when max stake is set to 0.

  Staking
    ✓ Should stake when validator is disabled
    ✓ Should revert when transfer not approved
    ✓ Should stake 1 token and emit event with correct number
    ✓ Should return correct delegated #
    ✓ Should revert when stake by validator is more than stake max cap
    ✓ Should revert when stake to invalid validator
    ✓ Should change contract balance
    ✓ Should change delegator balance
    ✓ Should succeed when stake by validator is at max cap
    ✓ Should revert when stake amount is too small

  Take out reward Tokens
    ✓ Should change balance of the contract and the owner.
    ✓ Should revert with wrong inputs.
    ✓ Should change rewardPool.
    ✓ Should emit AllocatedTokensTaken event with correct amount.
    ✓ Should revert when reward pool is too small.

  Transfer Unstaked
    ✓ Should transfer out after cool down ends, delegator
    ✓ Should transfer out after cool down ends, validator
    ✓ Should transfer out partially
    ✓ Should change balance of the contract and the owner.
    ✓ Should transfer out after cool down ends, validator
    ✓ Should revert with wrong unstaking id
    ✓ Should revert when the transfer amount is higher than unstaked
    ✓ Should revert when trying to attempt transfer the same unstake twice
    ✓ Should revert when cool down did not end, delegator
    ✓ Should revert when cool down did not end, validator
    ✓ Should revert when given invalid validator id

  Unstaking
    ✓ Should revert when unstake is more than staked
    ✓ Should revert when unstake is too small
    ✓ Should revert when unstake beyond max cap
    ✓ Should unstake with safe max cap
    ✓ Should unstake beyond max cap when validator is disabled
    ✓ Should emit event when unstaked successfully
    ✓ Should not change balance of contract or delegator
    ✓ Should revert when validator is invalid

  Tests addAuditor()
    ✓ Lets governor add an auditor and emits OperatorAdded event with correct args
    ✓ Reverts when non-governance address adds an auditor
    ✓ Should set correct role
    ✓ Should be able to add multiple auditors
    ✓ Should revert when trying to add auditor who is an operator

  Tests addGovernor()
    ✓ Lets owner add a governor and emits OperatorAdded event with correct args
    ✓ Reverts when non-owner address adds a governor
    ✓ Should set correct role
    ✓ Should be able to add multiple governors
    ✓ Should revert when trying to add a governor who is an operator

  Tests addBSPOperator()
    ✓ Lets governance address add bsp operator
    ✓ Emits OperatorAdded event
    ✓ Reverts when non-governance address preapproves an address for a role type
    ✓ Reverts when adds the same operator twice
    ✓ Sets operators under correct validator
    ✓ Operator is disabled after being added
    ✓ Should set correct validator id
    ✓ Should set correct operator role
    ✓ Should set correct validator id

  Tests addValidator()
    ✓ Lets a governance role add a new validator to the staking contract
    ✓ Emits ValidatorAdded when a new validator is added to the staking contract
    ✓ Reverts when non-governance tries to add a validator

  Block Specimen Arbitration Tests
    ✓ Reverts if non AUDITOR_ROLE to call the function
    ✓ Reverts if the block specimen session has not started
    ✓ Reverts if the deadline has not been reached
    ✓ Reverts when arbitration happens before finalize
    ✓ Allows arbitration after finalize
    ✓ Emits BlockSpecimenSessionFinalized after deadline arbitration
    ✓ Should emit BlockSpecimenRewardAwarded with correct args when quorum not reached
    ✓ Should emit BlockSpecimenRewardAwarded with correct args when correct hash was not submitted by anyone

  Tests disable operator
    ✓ Lets a validator disable their operator
    ✓ Does not let a non-validator to disable an operator
    ✓ Emits OperatorDisabled
    ✓ Does not let an operator disable an operator
    ✓ Does not let a validatorID be used by a different validator
    ✓ Does not let disable an operator that does not exist or performs a different role
    ✓ Does not let disable a disabled operator
    ✓ Should return false when called isEnabled
    ✓ Should disable validator with correct block number
    ✓ Should not disable a validator when there are other enabled operators
    ✓ Should remove operator from active operators

  Tests disableValidator()
    ✓ Lets a governance role disable a validator after they are added
    ✓ Emits ValidatorDisabled when a validator is disabled
    ✓ Reverts when non-governance tries to disable a validator
  Tests startOperatorRole()
    ✓ Lets a validator enable an operator if the validator is preapproved
    ✓ Reverts when an operator enables an operator if the validator is preapproved
    ✓ Reverts when a non-validator enables an operator
    ✓ Emits OperatorEnabled event
    ✓ Does not let a validatorID be used by a different validator
    ✓ Does not let enable an operator that does not exist or performs a different role
    ✓ Does not let enable an enabled operator
    ✓ Should return true when called isEnabled
    ✓ Should enable validator with correct block number
    ✓ Should not enable a validator when it is already enabled
    ✓ Should add operator to active operators

  Block Specimen Session finalization Tests
    ✓ Reverts if the block specimen session has not started
    ✓ Reverts if the deadline has not been reached
    ✓ Changes require audit to true when not enough participants submitted, emits event and reverts if called again
    ✓ Changes require audit to true when quorum was not reached, emits event and reverts if called again
    ✓ Emits specimen hash reward awarded event with the correct args when quorum is achieved
    ✓ Emits specimen hash reward awarded event with the correct args when quorum is achieved
    ✓ Emits specimen hash reward awarded event with the correct args when quorum is achieved

  Tests remove auditor
    ✓ Lets Governance remove an audior
    ✓ Emits OperatorRemoved
    ✓ Does not let a non-governance role call remove auditor
    ✓ Should revert when trying to remove an auditor that does not exist or has a different role

  Tests remove governor
    ✓ Lets Governance remove an audior
    ✓ Emits OperatorRemoved
    ✓ Does not let a non-owner role call remove governor
    ✓ Should revert when trying to remove an governor that does not exist or has a different role

  Tests Governance control: removeBSPOperator()
    ✓ Lets Governance remove an operator
    ✓ Emits OperatorRemoved
    ✓ Does not let a non-governance role call removeBSPOperator()
    ✓ Emits ValidatorDisabled on staking contract when count is 0
    ✓ Does not emit ValidatorDisabled on staking contract when count is > 0
    ✓ Removes bsp role
    ✓ Removes from bsps
    ✓ Removes operator from validator ids
    ✓ Should revert when trying to remove an operator that does not exist or has a different role

  Tests all setters
    ✓ Lets Governance change the blockSpecimenRewardAllocation
    ✓ Emits BlockSpecimenRewardChanged
    ✓ Does not let non-governance change the blockSpecimenRewardAllocation
    ✓ Tests the getter for blockSpecimenRewardAllocation
    ✓ Lets Governance change the blockSpecimenSessionDuration
    ✓ Emits SpecimenSessionDurationChanged
    ✓ Does not let non-governance change the blockSpecimenSessionDuration
    ✓ Tests the getter for blockSpecimenSessionDuration
    ✓ Lets Governance change the blockSpecimenQuorum
    ✓ Emits SpecimenSessionQuorumChanged
    ✓ Does not let non-governance change the blockSpecimenQuorum
    ✓ Tests the getter for blockSpecimenQuorum
    ✓ Sets the required stake for the roles
    ✓ Emits MinimumRequiredStakeChanged
    ✓ Does not let non-governance change minimum stake required
    ✓ Lets a governance role set the staking contract address to a new address
    ✓ Emits StakingInterfaceChanged and successfully executes when governance calls
    ✓ Changes staking interface
    ✓ Reverts when non-governance sets staking contract address to a new address
    ✓ Lets Governance change the minSubmissionsRequired
    ✓ Emits SpecimenSessionMinSubmissionChanged
    ✓ Does not let non-governance change the minSubmissionsRequired
    ✓ Tests the getter for minSubmissionsRequired
    ✓ Lets Governance change the minSubmissionsRequired
    ✓ Emits NthBlockChanged
    ✓ Does not let non-governance change the nthBlock
    ✓ Tests the getter for nthBlock
    ✓ Lets Governance change the setSecondsPerBlock
    ✓ Emits SecondsPerBlockChanged
    ✓ Does not let non-governance change the maxNumberOfHashesPer24H
    ✓ Tests the getter for maxNumberOfHashesPer24H
    ✓ Lets Governance change the maxSubmissionsPerBlockHeight
    ✓ Emits BlockSpecimenMaxNumberOfHashesPer24HChanged
    ✓ Does not let non-governance change the maxSubmissionsPerBlockHeight
    ✓ Tests the getter for maxSubmissionsPerBlockHeight
    ✓ Lets Governance change the chainSyncData and emits event with correct args
    ✓ Does not let non-governance change the chainSyncData
    ✓ Reverts when seconds per block is 0
    ✓ Tests the getter for maxSubmissionsPerBlockHeight
    ✓ Lets Governance change the allowedThreshold and emits event with correct args
    ✓ Does not let non-governance change the allowedThreshold
    ✓ Tests the getter for allowedThreshold

  Tests submitBlockSpecimenProof()
    ✓ Lets a BSP role submit a specimen proof
    ✓ Reverts when a non-BSP submits a specimen proof
    ✓ Reverts when invalid chain ID is provided
    ✓ Emits BlockSpecimenProductionProofSubmitted event with correct args
    ✓ Reverts when trying to submit out of bounds of live sync
    ✓ Should revert when attempt to submit after session has closed (reached its deadline)
    ✓ Should revert when attempt to submit after session has closed reached its deadline and being finalized
    ✓ Should revert when attempt to submit specimen hash for the same block height and block hash twice
    ✓ Should revert when attempt to submit when operator did not stake sufficiently when session has not been started
    ✓ Should revert when attempt to submit when operator did not stake sufficiently when session has already been started
    ✓ Does same block height on different chain IDs without collision
    ✓ Reverts when trying to submit for invalid block height
    ✓ Reverts when trying to submit more than max number of submissions allowed per block height
    ✓ Reverts when trying to submit for the same block hash per block height twice

  Initialize contract
ProofChain upgraded to: 0x0affCff6f052dAD0f12e56E4fbC105811220642a
      ✓ works before and after upgrading
```

Code Coverage

Coverage was run with `npx hardhat coverage`.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	98.59	96.46	95.95	99.08	
CovalentQueryTokenFaucet.sol	0	0	0	0	21,37,38,42
IOperationalStaking.sol	100	100	100	100	
OperationalStaking.sol	99.15	98.63	100	100	
ProofChain.sol	100	97.37	100	100	
contracts/ERC20Permit/	0	0	0	0	
ERC20Permit.sol	0	0	0	0	… 69,71,72,81
IERC2612Permit.sol	100	100	100	100	
All files	**96.33**	**94.78**	**92.21**	**96.62**	

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

```
0ff9167a39efd487d28570bfacb6076c615eddfafe8c7d6a74c5399f7029d191 ./contracts/OperationalStaking.sol
```

Tests

```
aa2bc6642f52635d986cf4c3a1d4d66d97cd5a36607c2148ea3129e2bc40ccdf ./test/operational-staking/integration-tests/all.js
c2705ef1250378aec1518c30c8ecadec939bd1f90f03e119d35deb82ee20ea94 ./test/operational-staking/integration-tests/RewardsCalculator.js
a7940ed0b7755c26788da02d5e7b04e6d6bad809fdf4c74743a9df2fac96b675 ./test/operational-staking/unit-tests/renounceOwnership.js
d7a4591fb3ec1542000a26e9fae2bdf983cb52e28faef5e577751fa6402b912a ./test/operational-staking/unit-tests/setValidatorMaxStake.js
0bef0ceff2df6c7e6007b4f6465f6b4bfa779287933a60eb68d158ddf9e487b5 ./test/operational-staking/unit-tests/depositRewardTokens.js
cf6a483843dd93d35608c4faceb0df4cb4b033413a6d28283f02b6759ab6e929 ./test/operational-staking/unit-tests/access.js
166df24fc3a673797b3345c777008d6aeabaf25bfc98ebb60382ce756ebafcf6 ./test/operational-staking/unit-tests/addValidator.js
54f5bccd5a1e9c235733445b1f02d48ea13741328f8fd0a937d3d0a03ec33d70 ./test/operational-staking/unit-tests/setValidatorCommissionRate.js
2f191a2fe546a867553c75b88bb775f88ded4d7a90ae49a91550e973c3ced47f ./test/operational-staking/unit-tests/redelegateUnstaked.js
06f3c2282e66ff1fbbce5ca6cc6583d12635c490c16bfa3292c6d81737452000 ./test/operational-staking/unit-tests/initialize.js
ecba2a2158623d1789b909d17005abb603c208e4ea7fcba19ab3ca55dceec2a4 ./test/operational-staking/unit-tests/stake.js
c7f3adc12cb311b0322df8f903f3d70294618cf5bc828e871e4c3d8df1f296a4 ./test/operational-staking/unit-tests/getDelegatorMetadata.js
ecc8ffa86699d741c84eb8d4b29e493759810cf015e5ed95444377becdb95669 ./test/operational-staking/unit-tests/redeemAllRewards.js
703197a152514961348aec45f96b74552a8b4e1e2c1dc49ca069a871afc0a614 ./test/operational-staking/unit-tests/disableValidator.js
98c5b0edeb0a6ed30c96891c3c8fdce5574f05c8eea70853c51a03bd65b13e15 ./test/operational-staking/unit-tests/redeemRewards.js
01ccf2f38388feff6da5678c2d164f17db9902f9f28191f69440c15a9a7edda1 ./test/operational-staking/unit-tests/takeOutRewardTokens.js
dd88e461cdadc75d556161801d610ca832e457ce1215e86346ddf91adebf87fe ./test/operational-staking/unit-tests/unstake.js
3d2526a173533cd3e335a95e11e0a5bd93478608db3268b088854caef4fd31ce ./test/operational-staking/unit-tests/enableValidator.js
9ae4372440939f79b6c95a36f5050c4e248e2e67fb717f74c626a201c3a130f9 ./test/operational-staking/unit-tests/getValidatorCompoundedStakingData.js
d19a4949739dff8e968d881194913cf1051e3d3761986b8db8c4516707b13ff6 ./test/operational-staking/unit-tests/getMetadata.js
5e1735493dcf569b1e53bce010a883dcc0d3b4f194cc508ddf98f2bfcae46dcc ./test/operational-staking/unit-tests/transferUnstakedOut.js
1e6628e1c15d82d0b37149a92b2be36a5d841183ca195d9c7ca28b3a6dd59df2 ./test/operational-staking/unit-tests/setValidatorAddress.js
b8eb0de542c3f1bcc23ea56872e3c7089078fd9bcd56bc262497ef7a36c98471 ./test/operational-staking/unit-tests/rewardValidators.js
8f8c6da2967b779890e8ea37364824e0195734796399005975573322 2a03a6f2 ./test/operational-staking/unit-tests/getAllValidatorsMetadata.js
da79d7a73fcd2bed813319b207161d79bf2f1896b4ef2a9e3d17cbb1f9bb216c ./test/operational-staking/unit-tests/getValidatorsMetadata.js
ed063fce05c26c2f82656626ba980d9e1e19813ef20e7b2388f344e61ed06473 ./test/operational-staking/unit-tests/setStakingManagerAddress.js
f226cf49847cb3d803399a4d4de5704cb3c99bdc8cef737200661336ea99c456 ./test/operational-staking/unit-tests/setMaxCapMultiplier.js
f778570aac390086b21843e31310b32eeae335b21547ab8d30dee41e92888834 ./test/operational-staking/unit-tests/gevValidatorStakingData.js
c65fd3eefd0ad0e891a768ba1a844ed016396811cece2fc3bfd9268bcae894ca ./test/operational-staking/unit-tests/recoverUnstaking.js
38c1efd38a9e679712c8e4079537dfa7970b2c55cf2cebdf2c2406d5734a3c9e ./test/operational-staking/unit-tests/redeemCommission.js
1d992f407644e5ae5d34a4a4c17390669b0014b8bb1ea9bc9b12d720ef004ae5 ./test/operational-staking/unit-tests/getValidatorMetadata.js
```

Changelog

- 2022-07-12 - Initial report
- 2022-08-02 - Re-audit report
- 2022-09-03 - Re-audit report

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.