Quantstamp Security Assessment Certificate

March 18th 2022 — Quantstamp Verified

Covalent Operational Staking Contract

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

Executive Summary

Type	Ethereum
Auditors	Ed Zulkoski, Senior Security Engineer Roman Rohleder, Research Engineer Hisham Galal, Research Engineer
Timeline	2022-03-14 through 2022-04-01
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
covalent-operational-staking-audit (initial report)	9513dc8
covalent-operational-staking-audit (final report)	55a9a3c

Total Issues	**6** (3 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**2** (1 Resolved)
Informational Risk Issues	**3** (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).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

We have reviewed Covalent's `OperationalStaking` contract. Only low or informational level issues were found, nonetheless, we suggest reviewing all findings before using the code in production. The test suite and specification were of high quality.
Update: All issues have been either resolved or acknowledged as of commit `55a9a3c`.

ID	Description	Severity	Status
QSP-1	Missing Input Validation	⌄ Low	Fixed
QSP-2	`rewardRedeemThreshold` can be avoided	⌄ Low	Acknowledged
QSP-3	Assumptions on external contracts	○ Informational	Acknowledged
QSP-4	Unlocked Pragma	○ Informational	Fixed
QSP-5	Privileged Roles and Ownership	○ Informational	Acknowledged
QSP-6	Unclear usage of `disabledAtBlock`	? 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.

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 v0.8.2

Steps taken to run the tools:

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

Findings

QSP-1 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `OperationalStaking.sol`

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following functions do not have a proper validation of input parameters:

1. The function `disableValidator()` should either check that `blockNumber != 0`, or possibly use `block.number`.
2. `setStakingManagerAddress()` does not check that `newAddress` is different from `address(0)`.
3. `takeOutRewardTokens()` does not check that `amount` is smaller or equal to `rewardPool`.
4. `addValidator()` does not check that `validator` is different from `address(0)` or that `commissionRate` is at least less than `DIVIDER`.
5. `recoverUnstaking()` does not check that `validatorId` is within `validatorsN` or that `unstakingId` is within `unstakings.length`.
6. `transferUnstakedOut()` does not check that `validatorId` is within `validatorsN`.
7. `redelegateUnstaked()` does not check that `unstakingId` is within `unstakings.length`.
8. `setValidatorMaxStake()` does not check that `maxStake` is non-zero.

Recommendation: Add the corresponding `require` statements to each function.

QSP-2 `rewardRedeemThreshold` can be avoided

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: In `_redeemRewards()`, when `redeemAll` is `true`, a `require` statement ensures that the total rewards exceeds `rewardRedeemThreshold`. However, a user can still specify an `amount` such that `0 < amount < rewardRedeemThreshold` to avoid this check.

Recommendation: It is not clear why `rewardRedeemThreshold` is needed, however if this is a requirement it should be enforced on both branches.

Update: From the Covalent team: It throws arithmetic overflow when reward is too small.

QSP-3 Assumptions on external contracts

Severity: *Informational*

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: As the repository only contains the single `OperationalStaking` contract under audit, we cannot verify correctness of external contract interactions. This includes the following:

1. In `rewardValidator`, we cannot check that the `stakingManager` handles the `rewardPool < amount` or `v.totalShares == 0` cases properly.
2. The `stakingManager` does not invoke `rewardValidator()` if `disabledAtBlock` has been set for a validator (it is unclear if this should be allowed).

Recommendation: Verify the correctness of external contract interactions.

QSP-4 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `OperationalStaking.sol`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

QSP-5 Privileged Roles and Ownership

Severity: *Informational*

Status: Acknowledged

File(s) affected: `OperationalStaking.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.
The `OperationalStaking.sol` contract contains the following privileged roles:

- `owner`, as initialized during `initialize()`:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceOwnership()`.

-
  . Assign a new owner (who must accept via acceptOwnership()), by calling `transferOwnership()`.
  . Set/Assign an arbitrary address to the `stakingManager` role, by calling `setStakingManagerAddress()`.
  . Change the commission rate of an arbitrary validator to an arbitrary value up to 10**18 (100%), by calling `setValidatorCommissionRate()`.
  . Deposit arbitrary amounts of CQT rewards token to the contract and accordingly increase the reward pool, by calling `depositRewardTokens()`.
  . Withdraw arbitrary amounts of CQT rewards token from the contract and accordingly decrease the reward pool, by calling `takeOutRewardTokens()`.
- `stakingManager`
  . Add new validators with arbitrary addresses and commission rates (above 10**18 or 100%), by calling `addValidator()`.
  . Enable or disable arbitrary validators, by calling `disableValidator()` with a zero or non-zero value respectively.
  . Reward validators with a smaller or equal amount of `rewardPool`, by calling `rewardValidator()`.

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.

QSP-6 Unclear usage of `disabledAtBlock`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `OperationalStaking.sol`

Description: Suppose the current block number is `2,000,000` and `disableValidator()` was invoked setting `v.disabledAtBlock = 100,000` for some validator `v`. Now suppose `v` wishes to unstake in the same block. We compute `cooldownEnd` on L268 as in: `uint128 coolDownEnd = uint128( v.disabledAtBlock != 0 ? block.number - v.disabledAtBlock : block.number);` (which will evaluate to 0), and then add `validatorCoolDown` on the following line (i.e., `180*6646 = 1,196,280`). Thus the validator will be able to unstake immediately.

Recommendation: It is likely that the `true` case of the ternary-statement should be `v.disabledAtBlock` instead of `block.number - v.disabledAtBlock`, however this is dependent on the `StakingManager` implementation.

Automated Analyses

Slither

Slither warns of potential reentrancy related to `_stake()`, however the external `CQT` contract is known and trusted.

Code Documentation

1. The word "commission" is misspelled throughout.

Adherence to Best Practices

1. Favor using `uint256` instead of `uint`. **Update: fixed.**
2. Avoid using `unchecked` throughout (except cases where explicit checks with error messages are used). **Update: fixed.**
3. To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the `delegator` address parameter in `OperationalStaking.Staked()`. **Update: fixed.**

Test Results

Test Suite Results

```
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.

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.

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.

  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

  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

  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 Rewards
    ✓ Should revert when requested amount 0
    ✓ Should revert when requested amount is too high

  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 not change contract balance
    ✓ Should revert when redelegating from enabled validator that was disabled

  Reward validator
    ✓ Should return false if reward pool is less than amount emitted
    ✓ Should return false when validator has nothing staked
    ✓ Should return true when successfull
    ✓ Should change reward pool
    ✓ Should commission available to redeem
    ✓ Should emit Rewarded event with correct validatorId, commission paid and amount emitted
    ✓ Should revert with invalid validator
    ✓ Should revert with 0 tokens emitted

  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.

  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.

  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

  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.

  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

  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

·-----------------------------------------------------|---------------------------|----------------|-----------------------------·
|                Solc version: 0.8.4                  ·  Optimizer enabled: true  ·  Runs: 1000000  ·  Block limit: 30000000 gas  |
·····················|·····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Methods                                                                                                                          |
·····················|·····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Contract           ·  Method                     ·  Min        ·  Max        ·  Avg            ·  # calls      ·  chf (avg)  |
```

```
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  addValidator               ·     105286  ·     122458  ·         118084  ·         122  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  depositRewardTokens        ·      59642  ·      76754  ·          75209  ·          78  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  disableValidator           ·      39914  ·      57026  ·          54447  ·          20  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  enableValidator            ·      34555  ·      36567  ·          34906  ·         120  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  recoverUnstaking           ·      58157  ·      92357  ·          64168  ·          11  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  redeemAllRewards           ·      69277  ·      71608  ·          69489  ·         219  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  redeemRewards              ·      67740  ·      71875  ·          68446  ·         500  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  redelegateUnstaked         ·      67287  ·      88999  ·          79435  ·          11  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  rewardValidator            ·      54393  ·      71517  ·          56109  ·         242  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  setMaxCapMultiplier        ·      37179  ·      37191  ·          37180  ·          24  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  setStakingManagerAddress   ·      37428  ·      54528  ·          52721  ·         123  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  setValidatorCommissionRate ·      39987  ·      40047  ·          40019  ·          11  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  setValidatorMaxStake       ·      37235  ·      37259  ·          37246  ·           9  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  stake                      ·      74092  ·     130204  ·          88391  ·         369  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  takeOutRewardTokens        ·      52045  ·      56857  ·          53652  ·           9  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  transferUnstakedOut        ·      56063  ·      60651  ·          56455  ·          24  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking  ·  unstake                    ·      77113  ·     101282  ·          94833  ·          49  ·          -  |
·····················|···························|·············|·············|···············|·············|·············
|  Deployments                                       ·                                           ·  % of limit   ·            |
·····················|···························|·············|·············|···············|·············|·············
|  OperationalStaking                                ·          -  ·          -  ·        4199872  ·        14 %  ·          -  |
·----------------------------------------------------|-------------|-------------|---------------|-------------|-------------·

  115 passing (2m)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	98.94	89.09	96.67	98.92	
OperationalStaking.sol	98.94	89.09	96.67	98.92	403,404
All files	98.94	89.09	96.67	98.92	
All files	**98.94**	**89.09**	**96.67**	**98.92**	

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

```
f264cdede367e74cbef9638dfcf3184a386cda46059a4081dfcab451469b6b95 ./contracts/OperationalStaking.sol
```

Tests

```
b428a0a348e7afa6539e69633f7a3fda76d5f2f330e88353e3a50a3a7c40a0cb ./test/fixtures.js
265a86d46a59636cb9a0b6e813ab655898fafa844993d682ac9b32470c5b1a81 ./test/integration-tests/all.js
bf9f8aa9376f4760845e55808d4baad60405b9ab5577f0f50fb8a6805fb5870f ./test/integration-tests/RewardsCalculator.js
f5dd88b0a23ded927b8969472defab79ace1051fabbb1c1d186158f328b81db7 ./test/unit-tests/setValidatorMaxStake.js
ebd81ab2048cb2114f299cfb953c849cb8753efb6b77f2bdd3cc14c910e5bc1b ./test/unit-tests/depositRewardTokens.js
1f0bc830147499a6561237dcc2aa49c6350bc0b819177dea01ed0a3f4938d647 ./test/unit-tests/access.js
173804cd24b0938d30373c81ef2798101c16ea3a657940e599608e8d2945b9b6 ./test/unit-tests/addValidator.js
11aa1986e8a1ff67236d86d04e9315979a0e6b12dc54d77d311f6c561921f441 ./test/unit-tests/setValidatorCommissionRate.js
a2155d71edc0e8b532ecbc9ea2573ca6cc6d1ed73f964b1c5e9c296fdff7965e ./test/unit-tests/redelegateUnstaked.js
fcdb3e6c43f7b13cf74dc2aac69886645ffd66eb7113735895814285a62453dd ./test/unit-tests/initialize.js
b5a6464921bfc7037800374378dd904c265fa7db8f7d243a482f724bc2b0b50a ./test/unit-tests/stake.js
506fbe6cbeb06a2f05df63ef71bdd5fb2025a840808776a2a33d9e27345ad214 ./test/unit-tests/getDelegatorMetadata.js
7f0dca0f3e89b13f9969a88e4e33db1137985d0f6b656f0b488635c3e3296113 ./test/unit-tests/redeemAllRewards.js
3a73a2c875fc3e40628069e2d3aae670ced76d0cedd886ffd90cc5975bd32908 ./test/unit-tests/disableValidator.js
461a4ef5cacc20789ccdafcf9545794358749c582d80102647baa43b163b4715 ./test/unit-tests/redeemRewards.js
7d3984ba7f55c07aaa50a4240eae10b064356b681e214e44143eddd56a5571fb ./test/unit-tests/takeOutRewardTokens.js
7fc93094ae8f7b17fb789a38be3bd874f3652dfafced6ddf02f648d4fb854651 ./test/unit-tests/unstake.js
3a0a56cf0d9908d2b8fe43578686be2e7c7c9dfe3ba320c0d86336f234e86f56 ./test/unit-tests/enableValidator.js
532bf583a78b33bccd577d04981fb39ee9698549eea48097ea9b862caa1c8b42 ./test/unit-tests/getValidatorCompoundedStakingData.js
895470035a4cd45ad4a1c7cc75822377c25a11bd9652a06e90d70013f19facac ./test/unit-tests/getMetadata.js
f67d5d67a562f35f4f1f1ff78f74c34bfefcec3d0bb3d428a45e5db1c96ea08e ./test/unit-tests/transferUnstakedOut.js
a5539fdfa89558e7d8f1d372fe4483eace9e2d73d333e20c1b617ed817edaabd ./test/unit-tests/rewardValidator.js
f1afb4d61f77e8b3a67c6239fd5820bdd415b4be01813fd1f3802b2d6ebc1362 ./test/unit-tests/setStakingManagerAddress.js
cf06aa4e2c0ac276c002bcb908a8411b79cd85a151586ad26974a1c69b3402d1 ./test/unit-tests/setMaxCapMultiplier.js
3429781312b9f20b6061522192a0e1a40a93ef3d748bbe8f54916ddbf91ad43c ./test/unit-tests/recoverUnstaking.js
411f5e13f7d3f7349dcc35c044c58fb08d43710d35e881b9738b9a80fe4ca56f ./test/unit-tests/getValidatorMetadata.js
```

Changelog

- 2022-03-18 - Initial report
- 2022-03-31 - Revised report based on commit `55a9a3c`

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.