Quantstamp Security Assessment Certificate

October 27th 2021 — Quantstamp Verified

Saddle Token

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

Executive Summary

Type	Governance Token
Auditors	Jose Ignacio Orlicki, Senior Engineer Sebastian Banescu, Senior Research Engineer Ed Zulkoski, Senior Security Engineer
Timeline	2021-09-20 through 2021-10-27
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
saddle-token	96295e8
saddle-token	617d17f

Total Issues	12 (11 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	4 (4 Resolved)
Informational Risk Issues	2 (2 Resolved)
Undetermined Risk Issues	6 (5 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 the code, documentation, and test suite and found several issues of various severities. Overall, we consider the code to be well-written but with insufficient documentation and a good test suite that can be improved given suggested changes from this report. We have outlined suggestions to better follow best practices, and recommend addressing all the findings to tighten the contracts for future deployments or contract updates. We recommend addressing all the **12** findings to harden the contracts for future deployments or contract updates. We recommend against deploying the code as-is.
Update: Quantstamp has audited the changes based on the commit for the `saddle-token` repository (617d17f). Of the original 12 issues, all 12 have been either fixed, acknowledged, or mitigated.

ID	Description	Severity	Status
QSP-1	Unchecked Function Arguments	Low	Fixed
QSP-2	Gas Concerns With Constructor and Vesting	Low	Fixed
QSP-3	Integer overflow	Low	Fixed
QSP-4	Vesting Release May Be Blocked By Integer Underflow	Low	Fixed
QSP-5	Missing Check for Zero ETH Transfers	Informational	Fixed
QSP-6	Privileged Roles And Ownership	Informational	Mitigated
QSP-7	Unclear Vesting Revocation Policy	Undetermined	Fixed
QSP-8	Unclear Vesting Policy When Beneficiary Is Changed	Undetermined	Fixed
QSP-9	Constant May Violate Requirements	Undetermined	Fixed
QSP-10	External Dependencies	Undetermined	Acknowledged
QSP-11	Unclear Logic for Paused Transfers	Undetermined	Fixed
QSP-12	Unclear Separation Of Duties	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 v0.8.0

- Mythril v0.22.23

Steps taken to run the tools:

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

Findings

QSP-1 Unchecked Function Arguments

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Vesting.sol`, `RetroactiveVesting.sol`, `SDL.sol`

Description:

1. `Vesting.initialize` should ensure that `_token` is non-zero and `_durationInSeconds` is non-zero (the latter is to avoid a possible divide-by-zero in `vestedAmount`).
2. `RetroactiveVesting.constructor` should ensure that all three address arguments are non-zero.
3. `SDL.constructor` should check that `_pausePeriod > 0` and also `_pausePeriod <= 52 weeks`. If the value were set to zero, it would allow immediate transfers since `_pause` would not be invoked. The constructor should also check that `_governance`, `_vestingContractTarget`, and each subfield of `_recipients` is non-zero (except `cliffPeriod`).
4. `RetroactiveVesting.verifyAndClaimReward()` does not check if `totalAmount` is greater than zero and if it fits into `uint120`. Due to the cast on L69, this could lead to an integer overflow in case `totalAmount` is greater than the largest integer in `uint120`.
5. `Vesting.changeBeneficiary()` does not check if `newBeneficiary != beneficiary`.
6. `SimpleGovernance.changeGovernance()` does not check that `newGovernance != governance`.

Recommendation: Add input validation to all the functions and parameters indicated above.

Update: Fixed on this pull request. Subitem number 3 (from the 6 subitems) is considered only mitigated as only checks for `_pausePeriod` and `_governance`were added.

QSP-2 Gas Concerns With Constructor and Vesting

Severity: *Low Risk*

Status: Fixed

File(s) affected: `SDL.sol`

Description: Given that the constructor is using a loop to distribute the initial amount of the token inside `Vesting()` contracts, there is the concern that given an enough long list of recipients the constructor will run out of gas. Also, there is no other way to include further recipients afterward if the list of recipients list has to be shortened.

Recommendation: Consider adding each single or batch segment of recipients and creating the vesting in a separate function from the constructor, without a loop.

Update: Fixed on this pull request.

QSP-3 Integer overflow

Severity: *Low Risk*

Status: Fixed

File(s) affected: `RetroactiveVesting.sol`

Description: Since Solidity 0.8.x does not perform safe casting by default, there is a potential integer overflow on L96 in `RetroactiveVesting.sol` where a summation of 2 `uint256` variables is cast to `uint120`.

Recommendation: Check that the sum is less than the maximum integer in `uint120` or use OZ's `SafeCast` library for casting.

Update: Fixed on this pull request.

QSP-4 Vesting Release May Be Blocked By Integer Underflow

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Vesting.sol`

Description: The `Vesting.vestedAmount()` function returns the minimum between `currentBalance` and `unreleased` on L129, and the comment right above on L128 indicates: "`currentBalance` can be 0 in case of vesting being revoked earlier." However, in the case when `currentBalance == 0`, the subtraction on L126 would underflow and the function would throw. This would also seemingly block the beneficiary from calling `release()`. However, since the intended vested amount would be anyway `0` it would effectively not make a difference to the beneficiary's balance. Nevertheless, the intended error message: "No tokens to release" would not be displayed to the beneficiary.

Recommendation: Check if the `currentBalance == 0` right at the beginning of the `else`-branch, after L122 inside `vestedAmount()`. If so, `return 0`.

Update: Fixed on this pull request.

QSP-5 Missing Check for Zero ETH Transfers

Severity: *Informational*

Status: Fixed

Description: On `rescueTokens()` there is a check to avoid transferring 0 tokens, but there is a check missing to transfer 0 ETH.

Recommendation: Add a revert check for ETH transfers too in `rescueTokens()`.

Update: Fixed on this pull request.

QSP-6 Privileged Roles And Ownership

Severity: *Informational*

Status: Mitigated

File(s) affected: `GenericERC20WithGovernance.sol`

Description: The owner (not `governance` address) of `GenericERC20WithGovernance` can mint any amount of tokens at any time.
The `governance` address of the `SDL` contract can:

1. Add/remove targets to the list of allowed addresses that may transfer SDL tokens even during the pause period.
2. May transfer any available ETH or any other ERC20 token from the `SDL` contract to any destination address.

Recommendation: Clarify the roles and privileged actions they can perform in publicly available end-user documentation.

Update: Mitigated with documentation added on the code on this pull request.

QSP-7 Unclear Vesting Revocation Policy

Severity: *Undetermined*

Status: Fixed

File(s) affected: `Vesting.sol`

Description: The contract mentions that vesting is "optionally revocable by the owner" and later mentions that `currentBalance` can be 0 in case of vesting being revoked earlier." However, it does not appear that any revocation logic exists.

Recommendation: Clarify if there is missing functionality, or update the comments.

Update: Fixed on this pull request.

QSP-8 Unclear Vesting Policy When Beneficiary Is Changed

Severity: *Undetermined*

Status: Fixed

File(s) affected: `Vesting.sol`

Description: The function `changeBeneficiary` allows governance to set a new beneficiary address for any `Vesting` contract. Suppose the `Vesting` contract owns 100 total tokens and they are 50% vested. If the old beneficiary address has recently been claimed, they would have received up to 50 tokens and the new beneficiary would only receive around 50 as well (after full vesting). However, if the previous beneficiary has not claimed any tokens yet, that address will not be entitled to any tokens after the change, and therefore the new address will be allotted all 100 tokens after the full vesting period. It is not clear which case is desirable.

Recommendation: Specify the intended use-case of `changeBeneficiary`.

Update: Fixed on this pull request.

QSP-9 Constant May Violate Requirements

Severity: *Undetermined*

Status: Fixed

File(s) affected: `RetroactiveVesting.sol`

Description: The constant `DURATION = 2 * (52 weeks)` is not documented. Since the provided requirement states that "transfer is blocked until 52 weeks pass...", it is not clear if this is defined correctly.

Recommendation: Add documentation and ensure that the constant is defined properly.

Update: Fixed on this pull request.

QSP-10 External Dependencies

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `RetroactiveVesting.sol`

Description: According to *natspec* in `L33` the Merkle root to check for eligibility is generated off-chain and registered in `constructor()`. It is not clear if this critical piece of software is secure or if the design is sound for the participants generating proofs that will be sent with function `verifyAndClaimReward()`.

Recommendation: Document the implementation of the Merkle root generator and how are the users going to be able to generate appropriate proofs for this mechanism.

Update: Acknowledged and added a link to docs on this pull request.

QSP-11 Unclear Logic for Paused Transfers

Severity: *Undetermined*

Status: Fixed

File(s) affected: `SDL.sol`

Description: According to *natspec* the addresses in `allowedTransferee` are allowed to transfer tokens even if the contract is paused. Is understood that these addresses can be `from` addresses when transferring. But in the logic at `_beforeTokenTransfer()` if the `to` address is in allowed then the transfer is allowed (`allowedTransferee[from] || allowedTransferee[to]`). This logic seems unclear and could allow users to partner with an allowed accomplice account to drain funds out of the contract, even during a paused contract scenario.

Recommendation: Consider limiting allowed addresses only for `from` addresses when paused.

Update: Fixed on this pull request.

QSP-12 Unclear Separation Of Duties

Severity: *Undetermined*

Status: Fixed

File(s) affected: `GenericERC20WithGovernance.sol`

Description: The `GenericERC20WithGovernance` contract extends `SimpleGovernance` and its description in the code comments indicates: "This contract simulates a generic ERC20 token that is mintable and burnable."
First, it is important to note that there is no publicly accessible burn function. Therefore, there is a deviation from the specification at this point.
Second, and more importantly, the `owner` and not the `governance` address is the one which is only allowed to call the `mint()` function. This is confusing as one would expect the decision to mint tokens would be given to the `governance` address, not to the `owner`.

Recommendation: Since `GenericERC20WithGovernance` extends both `Ownable` and `SimpleGovernance`, clarify what the duties of each of these roles are in the code comments or publicly available documentation.

Update: Fixed on this pull request.

Automated Analyses

Slither

Slither has detected many results out of which the majority have been filtered out as false positives and the rest have been integrated into the findings from this report.

Mythril

Mythril has detected many results out of which the majority have been filtered out as false positives and the rest have been integrated into the findings from this report.

Code Documentation

1. Unclear purpose of `_pausePeriod` in `SDL.sol`. The comment contains an extra word: "time in seconds until since deployment this token can be unpaused by the governance".
2. Typo on L105 in `RetroactiveVesting.sol`: "Address to calculated the vested amount for" -> "Address to calculate the vested amount for".

Adherence to Best Practices

1. In `Vesting.sol`, since `startTimestamp` is always initialized to `block.timestamp`, it is not clear that the check `blockTimestamp < startTimestamp` could ever fail, and therefore may be removable.
2. Event parameters of type `address` should be `indexed` to facilitate monitoring. None of the event parameters in the contracts from this repository are `indexed`:
   - `event Claimed(address account, uint256 amount);` from `RetroactiveVesting.sol`
   - `event Allowed(address target);` and `event Disallowed(address target);` from `SDL.sol`
   - `event SetGovernance(address governance);` from `SimpleGovernance.sol`
   - `event VestingInitialized(address beneficiary, uint256 cliff, uint256 duration);` and `event SetBeneficiary(address beneficiary);` from `Vesting.sol`.

Test Results

Test Suite Results

No test failed from a total of 49 tests. We recommend adding further tests based on the issues presented in this report if needed.

```
$ npm run test

> saddle-token@0.0.1 test
> hardhat test

No need to generate any newer typings.

  Retroactive Vesting
    verifyAndClaimReward
       ✓ Reverts when giving invalid proof
       ✓ Reverts when giving invalid amount
       ✓ Successfully claims when giving correct proof and amount
       ✓ Successfully claims for someone else when giving correct proof and amount
       ✓ Successfully claims after verifying once
    claimReward
       ✓ Successfully claims reward by themselves
       ✓ Successfully claims reward by themselves when providing zero address
       ✓ Successfully claims reward for someone else
    vestedAmount
       ✓ Reverts when account is not yet verified
       ✓ Successfully outputs correct vested amounts
       ✓ Successfully claims reward for someone else

  Token
    minting
Gas used to deploy token: 2368963
       ✓ Successfully mints to appropriate addresses and vestings
    totalSupply
```

```
    ✓ Successfully mints max supply (1e9 with 1e18 decimals) on deployment
  govCanUnpauseAfter
    ✓ Successfully sets govCanUnpauseAfter to be in the future on deployment
  anyoneCanUnpauseAfter
    ✓ Successfully sets anyoneCanUnpauseAfter to be 1 year after the deployment
  governance
    ✓ Successfully sets the governance address on deployment
  transfer
    ✓ Successfully transfers from an allowed address when paused
    ✓ Reverts when transfers from a not-allowed address to an allowed address when paused
    ✓ Reverts when transfers between not-allowed addresses when paused
  enableTransfer
    ✓ Reverts when governance attempts to unpause before govCanUnpauseAfter
    ✓ Reverts when non-governance attempts to unpause after govCanUnpauseAfter
    ✓ Succeeds when governance attempts to unpause after govCanUnpauseAfter
    ✓ Succeeds when non-governance attempts to unpause after anyoneCanUnpauseAfter
    ✓ Reverts when attempting to call enableTransfer after it is already unpaused
    transfer after enableTransfer
      ✓ Succeeds when transferring from an allowed address
      ✓ Succeeds when transferring from a not-allowed address to an allowed address
      ✓ Succeeds when transferring between not-allowed addresses
  addToAllowedList
    ✓ Succeeds to add an address to the allowed list
    ✓ Reverts when called by non-governance
  removeFromAllowedList
    ✓ Succeeds to remove an address from the allowed list
    ✓ Reverts when called by non-governance
  changeGovernance & acceptGovernance
    ✓ Reverts when called by other than the governance
    ✓ Succeeds to change governance
    ✓ Reverts when accepting governance when changeGovernance is not called before
    ✓ Reverts when accepting governance when called by other than pendingGovernance
  rescueToken
    ✓ Successfully rescues ETH
    ✓ Successfully rescues ERC20
    ✓ Reverts when called by non-governance

Vesting
  initialize
    ✓ Fails to initialize the logic contract
    ✓ Fails to initialize a clone with empty beneficiary
    ✓ Fails to initialize a clone with longer cliff than duration
    ✓ Successfully initializes a clone
  vestedAmount
    contract is initialized but NOT filled with tokens
      ✓ Successfully returns 0 when contract is empty
    contract is initialized and filled with some tokens
      ✓ Successfully calculates the vested amounts
      ✓ Successfully returns 0 when there are no more tokens left in the contract
  release
    ✓ Fails when there are no tokens to claim
    ✓ Successfully releases the vested amounts
  changeBeneficiary
    ✓ Fails when called by other than the governance
    ✓ Successfully changes beneficiary

·-------------------------------------------------------------|---------------------------|--------------|-----------------------------·
|                  Solc version: 0.8.6                        ·  Optimizer enabled: true  ·  Runs: 10000  ·  Block limit: 30000000 gas  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Methods                                                                                                                              |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Contract    ·  Method      ·  Min  ·  Max  ·  Avg  ·  # calls  ·  usd (avg)  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Cloner      ·  clone       ·     -  ·     -  ·  62897  ·       1  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC20       ·  approve     ·     -  ·     -  ·  46603  ·       1  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC20       ·  transfer    ·  31898  ·  55996  ·  52055  ·      12  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  GenericERC20WithGovernance  ·  mint  ·  70476  ·  70536  ·  70516  ·       3  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  GenericERC20WithGovernance  ·  transfer  ·     -  ·     -  ·  46704  ·       6  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  RetroactiveVesting  ·  claimReward  ·  45560  ·  63038  ·  51543  ·      15  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  RetroactiveVesting  ·  verifyAndClaimReward  ·  39183  ·  83339  ·  62362  ·      12  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  acceptGovernance  ·     -  ·     -  ·  27838  ·       1  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  addToAllowedList  ·  48423  ·  72626  ·  60525  ·       2  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  changeGovernance  ·     -  ·     -  ·  46339  ·       2  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  deployNewVestingContract  ·  249180  ·  249192  ·  249188  ·       3  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  enableTransfer  ·  27830  ·  27840  ·  27833  ·       6  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  removeFromAllowedList  ·     -  ·     -  ·  26523  ·       1  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  SDL         ·  rescueTokens  ·  33932  ·  54912  ·  44422  ·       2  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Vesting     ·  changeBeneficiary  ·     -  ·     -  ·  38440  ·       1  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Vesting     ·  initialize  ·     -  ·     -  ·  138653  ·       8  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Vesting     ·  release     ·  53893  ·  93550  ·  73722  ·       4  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Deployments                                    ·                    ·  % of limit   ·             |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Cloner                                         ·     -  ·     -  ·  154081  ·     0.5 %  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  GenericERC20WithGovernance                     ·     -  ·     -  ·  1232991  ·     4.1 %  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  RetroactiveVesting                             ·     -  ·     -  ·  893246  ·       3 %  ·          -  |
··············|··············|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Vesting                                        ·     -  ·     -  ·  970707  ·     3.2 %  ·          -  |
·-------------------------------------------------------------|-------------|-------------|--------------|--------------|-------------·

  49 passing (5s)
```

Code Coverage

The code coverage is very high but we strongly recommend increasing branch coverage to 100%.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	99.21	73.26	100	99.22	
RetroactiveVesting.sol	97.3	70.83	100	97.3	150
SDL.sol	100	64.29	100	100	
SimpleGovernance.sol	100	80	100	100	
Vesting.sol	100	83.33	100	100	
All files	**99.21**	**73.26**	**100**	**99.22**	

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

```
c96ed576d981da062ea2e065153afc02eb5e98882c52ef7a9910e17ed87fb8fa ./contracts/SDL.sol
494f0d91bdbae6790ffcff84e9f38aa894d09a9442dd49395193cdd8a79f937f ./contracts/RetroactiveVesting.sol
94e18f865a3c4afa74f0b85a4bea837b4c8d033e59f7036fffea6139c9d53439 ./contracts/Vesting.sol
2f70e260e093e235daf0e4e78cb37cf867df77db0bc0ee14eaafd3f45cf2ab48 ./contracts/SimpleGovernance.sol
499d3362fdb074b28d5616fce64e2592f1885768ad0491840322fa86de09ec37 ./contracts/helper/GenericERC20WithGovernance.sol
eff87b1f368f2c7dfe9fd0c4377d532b7fc9d3331c387072424af7d192501033 ./contracts/helper/Cloner.sol
```

Tests

```
433bab22a8cac52049af950b97483182b18be7ce039ed065ac31cb88d58129d3 ./test/vesting.ts
6ca13559b4fdb9edf0e20d7e6b94597d0969f6e47231c0b69396fe20bd0e3ab7 ./test/testUtils.ts
8a6605f2fc43cd488b137ab4a96940dcf2b257f67e9195f51c492ced206fba83 ./test/retroactiveVesting.ts
b815ff85b8155dae1a871ab42b6588401c3a9ed5e916880984a237b8d074a153 ./test/token.ts
```

Changelog

- 2021-09-24 - Initial report
- 2021-10-27 - Reaudit 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.