Quantstamp Security Assessment Certificate

August 5th 2020 — Quantstamp Verified

CurveDAO (specific files only)

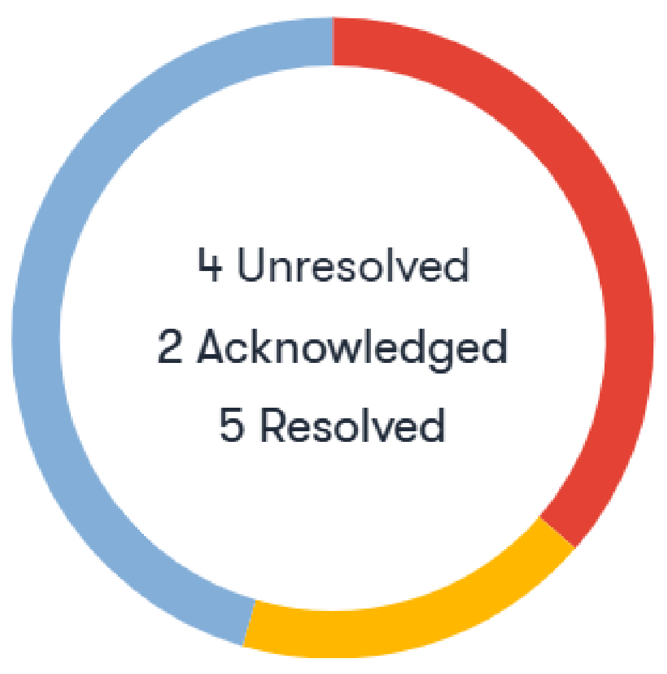
This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Audit
Auditors	Poming Lee, Research Engineer Ed Zulkoski, Senior Security Engineer Kevin Feng, Software Engineer
Timeline	2020-07-21 through 2020-08-05
EVM	Muir Glacier
Languages	Vyper
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Smart Contract - documentation
Source Code	

Repository	Commit
curve-dao-contracts	093138f
curve-dao-contracts	ebf4aec
curve-dao-contracts	afbe293

Total Issues	11	(5 Resolved)
High Risk Issues	1	(1 Resolved)
Medium Risk Issues	1	(0 Resolved)
Low Risk Issues	1	(1 Resolved)
Informational Risk Issues	7	(3 Resolved)
Undetermined Risk Issues	1	(0 Resolved)



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

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

Summary of Findings

During the audit, we found potential issues with varying levels of severity: one high-severity, two medium-severity, six low-severity issues, and seven informational-level findings. The code looks well-structured and concise, however, the pdf documentation may be slightly out-of-date. Additionally, more comments explaining large functions in the implementation are necessary for lowering the difficulty of future maintenance. Finally, we made 12 best practice recommendations which include naming, documentation, and other suggestions. We highly recommend addressing these findings before going live. Disclaimer: Please be aware that the scope of this audit as requested by the client was the two files contracts\LiquidityGauge.vy and contracts\GaugeController.vy and not the whole system.
** 2020-07-31 update **: Four findings are fixed, two findings are acknowledged, two false positive findings are clarified and removed from the report, and one finding remain unsolved.
** 2020-08-05 update **: QuantStamp was requested to audit additional materials; to be specific: 1) the diff between LiquidityGauge and LiquidityGaugeReward, and 2) VestingEscrow. During this audit, four informational-level findings were found and 6 best practice recommendations were made.

ID	Description	Severity	Status
QSP-1	Possibility of Missing Rewards	High	Fixed
QSP-2	Looping with high limit	Medium	Acknowledged
QSP-3	Business logic contradicting the specification	Low	Fixed
QSP-4	Unlocked Pragma	Informational	Fixed
QSP-5	Missing input checks	Informational	Fixed
QSP-6	Missing input checks	Informational	Fixed
QSP-7	Centralization of power	Informational	Acknowledged
QSP-8	Missing Input Check	Informational	Unresolved
QSP-9	Missing Input Check	Informational	Unresolved
QSP-10	Missing Input Check	Informational	Unresolved
QSP-11	Unspecified semantics for `claimable_reward()`	? Undetermined	Unresolved

Quantstamp Audit Breakdown

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

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:

- SmartCheck Released v2.0
- Mythril 0.22.8

Steps taken to run the tools:

1. Install SmartCheck globally To install SmartCheck globally to your system run (administrative rights required)

```
npm install @smartdec/smartcheck -g
```

3. (Optional) Add SmartCheck as development dependency To add and install SmartCheck as development dependency to your npm project run:

```
npm install --save-dev @smartdec/smartcheck
```

5. Start the analysis To start analysis simply run:

```
smartcheck -p .
```

7. Installed the Mythril tool from Pypi: `pip3 install mythril`

8. Ran the Mythril tool on each contract: `myth analyze FlattenedContract.sol`

Findings

QSP-1 Possibility of Missing Rewards

Severity: *High Risk*

Status: Fixed

File(s) affected: `contracts\LiquidityGauge.vy`

Description: Regarding the loop on `L182` with `range(500)`, if the user does not check in within 500 periods, the computation would finish with some fraction of the integral not accounted for. This would lead to a portion of the reward not given to the user.

Recommendation: Calculate the value forwardly instead of backwardly, and store the latest user_period when jumping out the for loop might solve this issue. However, it is also recommended to carefully examine the influence of this modification to the original design and modify the other parts related to this modification.

QSP-2 Looping with high limit

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `contracts\GaugeController.vy`

Description: Loop limits may be too high. There are several functions that have a limit of 500 on loop. In the worst case, these functions may reach the block gas limit and revert.
** 2020-07-31 update **: Curve team confirmed that up to at least a year missed (extremely unlikely unless the pool is completely abandoned) it should work fine.

Recommendation: It is recommended to estimate the gas costs carefully. The estimated gas cost can be obtained with e.g., `vyper --show-gas-estimates GaugeController.vy -f ir`.

QSP-3 Business logic contradicting the specification

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\LiquidityGauge.vy`

Description: In `_checkpoint()`, the block from `L141-167` intends to "Update integral of 1/supply". Suppose that many weeks have passed since the last checkpoint, and during that time, the rate has changed several times. This computation doesn't account for such a scenario. It appears that the rate can only take on the values from `rate` and `new_rate` as defined on `L130-134`. Note that within the pdf on p. 7, it claims that this value denoted `I_is` is updated every time the rate is changed (which would resolve this issue), however we couldn't find in the code where this call actually occurs.

Recommendation: State that risk in the documentation and let the users be aware of that.

QSP-4 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts\LiquidityGauge.vy`, `contracts\GaugeController.vy`

Description: Every Vyper file specifies in the header a version number of the format `@version ^0.2.*`. 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." For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Vyper version.

QSP-5 Missing input checks

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts\LiquidityGauge.vy`

Description: In `__init__()`, should check that the function arguments are non-zero.

QSP-6 Missing input checks

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts\GaugeController.vy`

Description: In `__init__()`, should check that the function arguments are non-zero. A similar issue exists for `add_gauge()`.

QSP-7 Centralization of power

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts\GaugeController.vy`

Description: The contract owner can add new types and arbitrarily high weights at any time. The admin can change the weight of an existing gauge by using the function `change_type_weight`to overwrite the weight of `next_time` at any moment as well.
** 2020-08-05 update **

1. the Curve team stated that the admin of the contract `GaugeController` will be the DAO instead of themselves.
2. For `contracts\VestingEscrow.vy`, the admin roles can disable the amount that is vested into associated accounts making them claim less, consider adding this to the document.

Recommendation: State that risk in the documentation and let the users be aware of that.

QSP-8 Missing Input Check

Severity: Informational

Status: Unresolved

File(s) affected: `Contracts\LiquidityGaugeReward.vy`

Description: Should check that `_rewarded_token` is non-zero in the constructor.

QSP-9 Missing Input Check

Severity: Informational

Status: Unresolved

File(s) affected: `contracts\VestingEscrow.vy`

Description: For Function `fund` should check if `_recipients` has called this function before. Alternatively, consider explicitly specifying that under the condition that `self.initial_locked[_recipients[i]]` is not zero, how should the contract react.

QSP-10 Missing Input Check

Severity: Informational

Status: Unresolved

File(s) affected: `contracts\VestingEscrow.vy`

Description: Should check that addresses in `__init__` are non-zero.

QSP-11 Unspecified semantics for `claimable_reward()`

Severity: Undetermined

Status: Unresolved

File(s) affected: `Contracts\LiquidityGaugeReward.vy`

Description: Should `claimable_reward()` take into account the `claimed_rewards_for` of `_addr`, such that it only returns the amount that has not already been claimed?

Automated Analyses

SmartCheck

SmartCheck reported findings in regard to the dependency of `block.timestamp`, and suggestions about visibility settings of variables. After checking, these findings are considered as false positives.

Mythril

Mythril reported findings in regard to the dependency of `block.timestamp`. After checking, these findings are considered as false positives.

Code Documentation

Added to the `Best Practices Review` section.

Adherence to Best Practices

- `def kick` in `contracts\LiquidityGauge.vy` did not do anything that seems to be "kicking" a user.
- On `L410` in `contracts\GaugeController.vy`, should check if `self.n_gauge_types` is smaller than `100` due to `L213` & `L225` 's for loop's limitation.
- In `contracts\LiquidityGauge.vy`, please rename the `TOKENLESS_PRODUCTION` so that it can reflect the `b_u` formula (in p.7), or simply add this variable name to the pdf file.
- In `contracts\LiquidityGauge.vy` a comment would be useful to describe why `TOKENLESS_PRODUCTION` and `BOOST_WARMUP` exist.
- In `contracts\LiquidityGauge.vy` the comment on `L213`(i.e., `# XXX explain`) should be updated.
- In `contracts/LiquidityGauge.vy` the contract has a two-part ownership transfer process: `commit_transfer_ownership() -> apply_transfer_ownership()`. Typically, this pattern is used to ensure that the future owner address is correct. However, `apply_transfer_ownership()` is required to be called by the existing owner, not the future one. It is recommended to change `apply_transfer_ownership()` so that it can only be called by the future admin.
- For `contracts/LiquidityGauge.vy`, the documentation doesn't seem to specify `BOOST_WARMUP` (`L107`) which only allows the user to receive an extra CRV bonus if and only if 2 weeks have passed after the pool has been initialized.
- On `L92` of `contracts/LiquidityGauge.vy`, the parameters `L` and `l` of the function `_update_liquidity_limit` are recommended to use more meaningful names

- For `contracts\GaugeController.vy`, the documentation (i.e., the pdf) seems to be relatively outdated, also does not highlight how the voting process exactly works.
- For `contracts\GaugeController.vy`, the following function names do not fully reflect the actual computations that are performed. It is recommended to change the name to match the side effects (i.e., filling up missing check in during checkpoints, and obtaining the specific variable of a weight point): 1) `_get_type_weight` -> `update_n_get_type_weight`, 2) `_get_weight` -> `update_n_get_weight_bias`, 3) '_get_sum' -> `update_n_get_sum_weight_bias`, 4) '_get_total' -> `update_n_get_total_weight`
- For `contracts\GaugeController.vy` the function `vote_for_gauge_weights`, the user cannot vote if their locked period is too close to the end date (specifically within one week) due to `assert lock_end > next_time`. This should be documented in the pdf file.
- For `contracts\GaugeController.vy` it is recommended to add comments to explicitly explain that the constant `10000` in `L482` and `L493` is used for reflecting the maximal value and the precision of the percentage.

** 2020-08-05 update **

- For `contracts\VestingEscrow.vy`, the `_recipient` is not used in the function `disable_can_disable`.
- On `L83` of `contracts\VestingEscrow.vy`, consider do `is_disabled: bool = false` instead.
- For `contracts\VestingEscrow.vy`, check the TODO list on `L49-L52` and finish the TODOs.
- Duplicated code. In `contracts\LiquidityGaugeReward.vy`, `claim_rewards()` can simply invoke `claim_rewards_for(msg.sender)`.
- In `contracts\LiquidityGaugeReward.vy`, the function `claimable_reward`, the variable `addr` seems to be set to `msg.sender` no matter what the parameter `addr` is. This assignment operation below it seems to be unnecessary.
- For `contracts\LiquidityGaugeReward.vy`, consider to rename the mapping `rewards_for` to `total_rewards_for` for clarity purposes.

Test Results

Test Suite Results

To summarize, 179 tests passed as of commit `093138f`.

```
Brownie v1.10.3 - Python development framework for Ethereum
Compiling contracts...
  Vyper version: 0.2.3
Generating build data...
 - Registry...
 - ERC20...
 - ERC20CRV...
 - VotingEscrow...
 - PoolProxy...
 - LiquidityGauge...
 - CurvePool...
 - GaugeController...
 - Minter...
 - ERC20LP...
============================= test session starts ==============================
platform linux -- Python 3.8.2, pytest-5.4.3, py-1.9.0, pluggy-0.13.1
rootdir: /root/curve-dao/20200731-ebf4aec-reaudit1
plugins: eth-brownie-1.10.3, forked-1.2.0, web3-5.11.1, xdist-1.33.0, hypothesis-5.20.1
Launching 'ganache-cli --port 8545 --gasLimit 12000000 --accounts 10 --hardfork istanbul --mnemonic brownie'...
collected 179 items
tests/test_scalability.py .                                              [  0%]
tests/integration/ERC20CRV/test_mint_integration.py ...                  [  2%]
tests/integration/ERC20CRV/test_mintable_in_timeframe.py ....            [  4%]
tests/integration/GaugeController/test_types_and_weights.py .            [  5%]
tests/integration/GaugeController/test_vote_weight.py .                  [  5%]
tests/integration/LiquidityGauge/test_deposits_withdrawals.py .          [  6%]
tests/integration/LiquidityGauge/test_liquidity_gauge.py ..              [  7%]
tests/integration/Minter/test_components.py ..                           [  8%]
tests/integration/Minter/test_minter_integration.py .                    [  8%]
tests/integration/VotingEscrow/test_deposit_withdraw_voting.py .         [  9%]
tests/integration/VotingEscrow/test_voting_escrow.py .                   [ 10%]
tests/integration/VotingEscrow/test_zero_balance_at_unlock_time.py ..    [ 11%]
tests/unitary/ERC20CRV/test_burn.py ....                                 [ 13%]
tests/unitary/ERC20CRV/test_epoch_time_supply.py .......                 [ 17%]
tests/unitary/ERC20CRV/test_mint.py .....                                [ 20%]
tests/unitary/ERC20CRV/test_setters.py ......                            [ 23%]
tests/unitary/GaugeController/test_gauges_weights.py ..........          [ 29%]
tests/unitary/GaugeController/test_timestamps.py .                       [ 30%]
tests/unitary/GaugeController/test_total_weight.py ....                  [ 32%]
tests/unitary/GaugeController/test_vote.py ..........                    [ 38%]
tests/unitary/GaugeController/test_vote_weight_unitary.py ....           [ 40%]
tests/unitary/LiquidityGauge/test_checkpoint.py ...                      [ 42%]
tests/unitary/LiquidityGauge/test_deposit_withdraw.py ......             [ 45%]
tests/unitary/LiquidityGauge/test_kick.py .                              [ 46%]
tests/unitary/Minter/test_minter.py .........                            [ 51%]
tests/unitary/PoolProxy/test_emergency_admin.py ........                 [ 55%]
tests/unitary/PoolProxy/test_owner_admin.py ............................ [ 71%]
....                                                                     [ 73%]
tests/unitary/PoolProxy/test_parameter_admin.py ........................ [ 87%]
....                                                                     [ 89%]
tests/unitary/PoolProxy/test_proxy_burn.py ..............                [ 97%]
tests/unitary/PoolProxy/test_set_admins.py ....                          [100%]
=============================== warnings summary ===============================
tests/integration/VotingEscrow/test_voting_escrow.py:12
tests/integration/VotingEscrow/test_voting_escrow.py::test_voting_powers
  /root/curve-dao/20200731-ebf4aec-reaudit1/tests/integration/VotingEscrow/test_voting_escrow.py:12: DeprecationWarning: invalid escape sequence \
    """
-- Docs: https://docs.pytest.org/en/latest/warnings.html
================= 179 passed, 2 warnings in 2955.92s (0:49:15) =================
Terminating local RPC client...
```

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

```
5df6489e8e0690c871a442549467599a44c3231dbbb6168091345f6da19ef0d9 ./contracts/ERC20CRV.vy
b0f02e124222a64fe8783bacadb3fad19ea9828629d079dea9440a551dd210e4 ./contracts/GaugeController.vy
277027e1fb91bfe324b6add88022f4dde9692e9609b17da4bf30740c53a1dd09 ./contracts/LiquidityGauge.vy
ee11e160197939ad7ad91cf735e52b414f43da063d67dad2044d0a4c1d279186 ./contracts/Minter.vy
f1ca1568cea74d3222e3d39e3a9fbfc5da672248ca8eae5a176011f4cee0da88 ./contracts/PoolProxy.vy
f5551fdf655fe61f0f86e0b0537dc3c02c520c03e7c64ff9f4bd898f2e0277f7 ./contracts/VotingEscrow.vy
4e472b11babb778b3af2c224d15319fde5f9fe5d739c883bc73e2c6566a043fe ./contracts/testing/CurvePool.vy
a70eca707239b7b71eb223c2abe98980c5bbcbbb1a7f79da6844a1ca0ec28bf1 ./contracts/testing/ERC20.vy
8bdb29d91261589b3cb5ec3e1378b5cabdd642b073d7bd9fea971ed6bc00fcbe ./contracts/testing/ERC20LP.vy
94daf7123a052252aea8b9fa8e0de669a15bf1b4ebc153f0fe0a845617eee1ac ./contracts/testing/Registry.vy
```

Tests

```
4544f3e5d1cac15940b9b7dcb2120817565fa657df7e00eb8ab8f5a7c40caa09 ./tests/conftest.py
5d04fec52712504cdf3752bd2c7588462c7ec490628bc2b8063e52cf66cdcca3 ./tests/test_scalability.py
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./tests/__init__.py
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./tests/unitary/__init__.py
7213ff3a7cc42c8ef41727a441ab7ed98f79589079f29273d47c05f50797a36a ./tests/unitary/PoolProxy/conftest.py
974701ba58cab99a6f8e6ad92d7bf8af02eee4362fd34f75458c37d54e383900 ./tests/unitary/PoolProxy/test_emergency_admin.py
60b4b27c750ab209dc5f27d8f54dd2927f22a4ca2f7384790d5592bc33b6d394 ./tests/unitary/PoolProxy/test_owner_admin.py
5d6d2ab65a7285302c6f9607b824fcd94bcb765fd72f456187a61f5b499f5a2d ./tests/unitary/PoolProxy/test_parameter_admin.py
89615f76068ed9434c411f191fcc95d0ea0d93d68a319da0457b3672df7f5b82 ./tests/unitary/PoolProxy/test_proxy_burn.py
19443dbf4b6af32de8c1cb516655ddf9e3e3204a0a58b4c6d558d48e326df5a8 ./tests/unitary/PoolProxy/test_set_admins.py
0ab66f078d39ab9da158d5065005113fafa66f36cfcff761f4d3b63ecb2bb8a5 ./tests/unitary/Minter/test_minter.py
f73a87fca2514a7915c7ebb1d69dadbc51b4fa4cda81f15938f4c59bd29d339d ./tests/unitary/LiquidityGauge/test_checkpoint.py
38cc357690bf051c0babb556dec89adf15cbaefa4b1079005f2a04cb62a61f31 ./tests/unitary/LiquidityGauge/test_deposit_withdraw.py
1bb7d41d793505f22fa1272f19e586f86b34c2b48ba5ddbaaea287ef2a44600d ./tests/unitary/LiquidityGauge/test_kick.py
2639b6428af365dc0678c0736f06a8370fa1ab8a5acfb9f65746ba7aecc3ef7c ./tests/unitary/GaugeController/conftest.py
0990eaecb32a6730f7028874e19af3f64038867470d2cbd5b9d20bf06bad7e07 ./tests/unitary/GaugeController/test_gauges_weights.py
7c1a18cb4c8a22ec05da9524702835a617132f2f82f97080e5a2c39c5bb03f2b ./tests/unitary/GaugeController/test_timestamps.py
0c30dda8dae0727bf6e271e7f2e456a97faecbde49bbb885e8f39c55d61bdd03 ./tests/unitary/GaugeController/test_total_weight.py
ed133eb5e6458677ac6368c83bff608986beec8517de7a825109b2d9163b0fd5 ./tests/unitary/GaugeController/test_vote.py
78a93ea47c45a7d7d0b4019ee6d6b67d512b6b9ec953e619e0aa32adc53ff9c8 ./tests/unitary/GaugeController/test_vote_weight_unitary.py
9c4f03c3afb88150710520a51f7ed0f76ee0438e2a8826d88c37aac5ee7e8593 ./tests/unitary/ERC20CRV/test_burn.py
e7fe9f246b955c5b8981a9fbbe7df9b85c89926020074bebf621753e410f9f12 ./tests/unitary/ERC20CRV/test_epoch_time_supply.py
b760b6ce4a071f5cc7e286b234ae43d124d3cf1f8a40327e37e12416d66edf9c ./tests/unitary/ERC20CRV/test_mint.py
0500ab64476c1338258dbd7eeffaad2714342fde1e987eea386661be4238a98f ./tests/unitary/ERC20CRV/test_setters.py
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./tests/integration/__init__.py
83acbe9d13fd18c769d6e7e68c96e43242ed80c81d497dce5835d68bb34a7d43 ./tests/integration/VotingEscrow/test_deposit_withdraw_voting.py
410a462dcc0f535b279054431df72bc5a90680a1fc44682073cd90f2e68574b8 ./tests/integration/VotingEscrow/test_voting_escrow.py
fb27c629c846c317b6109bfdae7ccabf4e20ee402b14248441c2a79ffdb14ec0 ./tests/integration/VotingEscrow/test_zero_balance_at_unlock_time.py
4d73b56266d01c36d8c069879057cd650a999f4d4b63ba6136aa23de44374a3a ./tests/integration/Minter/test_components.py
284113e8ce19d70f6d8e36e88e2ba72f13cef86529ce52ef14fd5190033ed750 ./tests/integration/Minter/test_minter_integration.py
69730151efd053919100fdf5d62740fc993cda8eb28f57e11853acfc152016ff ./tests/integration/LiquidityGauge/test_deposits_withdrawals.py
c0de3508ee1cc147a78de8ecb6bbcae642d8e6201c9bff24da22def224b75fe0 ./tests/integration/LiquidityGauge/test_liquidity_gauge.py
910c37eb93d96a59d6bfc8cfde21fbae188e97b777ef4e7a4114bf16d0235ce5 ./tests/integration/GaugeController/test_types_and_weights.py
2582f3ff4eefc68caf3d8668d3b47a75777aa6011f344d3eec4fe5f595c16cca ./tests/integration/GaugeController/test_vote_weight.py
bb5a36bb4bde7404bfcc7d18afb8289994eb558640ddb214b7d293ebe0f7fbdf ./tests/integration/ERC20CRV/test_mintable_in_timeframe.py
c34de0d44822760a5144e62f9d270b6c98a558deaf042845f7e5b973ae5749eb ./tests/integration/ERC20CRV/test_mint_integration.py
```

Changelog

- 2020-07-28 - Initial report

- 2020-07-31 - reaudit report
- 2020-08-05 - additional 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.