Quantstamp Security Assessment Certificate

June 21st 2021 — Quantstamp Verified

API3

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

Executive Summary

Type	Staking Pool and DAO
Auditors	Fayçal Lalidji, Security Auditor Ed Zulkoski, Senior Security Engineer Jose Ignacio Orlicki, Senior Engineer
Timeline	2021-05-03 through 2021-06-18
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
api3-dao (283e6ba)	283e6ba
api3-dao (d3db4e7)	d3db4e7

Total Issues	9 (9 Resolved)
High Risk Issues	1 (1 Resolved)
Medium Risk Issues	1 (1 Resolved)
Low Risk Issues	2 (2 Resolved)
Informational Risk Issues	3 (3 Resolved)
Undetermined Risk Issues	2 (2 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Initial Audit:
While the inline documentation was in general of high quality, the file `Api3Template.sol` could use more documentation. Through reviewing the code, we found **9 potential issues** of various levels of severity. We recommend addressing the findings prior to deploying the smart contracts to the main network.
Update:
All highlighted issues have been either fixed or mitigated by the API3 team.

ID	Description	Severity	Status
QSP-1	Approved Funds Can Be Exploited by Attackers	High	Fixed
QSP-2	Gas Usage / `for` Loop Concerns	Medium	Mitigated
QSP-3	`updateCheckpointArray` may be susceptible to griefing	Low	Fixed
QSP-4	Race Condition on `setDaoApps()`	Low	Fixed
QSP-5	Unchecked function arguments	Informational	Fixed
QSP-6	Undocumented Constants	Informational	Fixed
QSP-7	Privileged Roles and Ownership	Informational	Fixed
QSP-8	Off-by-One Error in `updateCheckpointArray`	Undetermined	Fixed
QSP-9	Unclear Condition	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 v0.7.1
- Mythril v0.22.19

Steps taken to run the tools:

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

Findings

QSP-1 Approved Funds Can Be Exploited by Attackers

Severity: *High Risk*

Status: Fixed

Description: The funds approved to the `Api3Pool` can be withdrawn by an attacker using `TimelockUtils.depositWithVesting` and `TimelockUtils.updateTimelockStatus`. This issue is due to allowing the `msg.sender` in `TimelockUtils.depositWithVesting` to input any value for `source` address while not restricting the function access to only the the timelock manager address. Similarly, the `msg.sender` is free to input any value for `timelockManagerAddress` input in `TimelockUtils.updateTimelockStatus` function.

Exploit Scenario:

1. Alice approve all her funds to be spent by `Api3Pool` (please note that even when approving only the amount to be staked, Alice still runs the risk to be front-runned following the same scenario).
2. Bob the attacker calls `TimelockUtils.depositWithVesting` while setting the `source` address to Alice's address and the `userAddress` to its own address. This step allows Bob to transfer Alice's fund to its own balance in `Api3Pool`.
3. After the `releaseStart`, Bob calls `TimelockUtils.updateTimelockStatus` with both `userAddress` and `timelockManagerAddress` set to its own address. This step allows Bob to vest his deposited amount in step 2.
4. Bob calls `TransferUtils.withdraw` to transfer the stolen funds to its own wallet.

Recommendation:

We recommend to:

1. Restrict the `TimelockUtils.depositWithVesting` function calls to only allow the timelock manager address.
2. Remove `timelockManagerAddress` from `TimelockUtils.updateTimelockStatus` and use state variable that saves the timelock manager address instead.

QSP-2 Gas Usage / `for` Loop Concerns

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: `GetterUtils.sol`

Description: Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely.
Affected functions:

1. `GetterUtils.userDelegateAt`.
2. `GetterUtils.getUserLocked`.
3. `GetterUtils.getValueAt`.

Recommendation:

1. We recommend to implement the same binary search algorithm as implemented in `GetterUtils.userSharesAtWithBinarySearch`.
2. It is best to break such loops into individual functions as possible.
3. We recommend to implement the same binary search algorithm as implemented in `GetterUtils.userSharesAtWithBinarySearch`.

Update: Team answer: "All linear searches are replaced with binary searches except `getUserLocked()` because for the majority of the users (which will stake once and remain staked for long duration) the current implementation is extremely more gas-efficient compared to the alternatives. However, the current implementation disables users from withdrawing if they have made a lot of staking updates in the last year (more than 50 per week according to tests). As a solution, an additional `withdrawPrecalculated()` method is implemented to allow such users to withdraw their funds by making multiple transactions (note that we don't consider this as normal user flow, it's a fail-safe). This makes `MAX_INTERACTION_FREQUENCY` unnecessary, which is why it's removed. Finally, the delegate address checkpoint array is kept in a naive way (doesn't overwrite if there hasn't been a new proposal) to simplify the implementation, as the binary search allows this (and users can't update delegates frequently anyway so this wasn't providing any benefits assuming there is going to be at least a proposal a week, which means delegate checkpoints wouldn't have been overwritten anyway)."

QSP-3 `updateCheckpointArray` may be susceptible to griefing

Severity: *Low Risk*

Status: Fixed

File(s) affected: `DelegationUtils.sol, StakeUtils.sol, StateUtils.sol`

Description: When a user stakes or unstakes tokens, `DelegationUtils.updateDelegatedVotingPower` is invoked to ensure that the user's delegate's voting power will be updated (if they exist). However, if many users also delegate to the same address, updating stake may fail due to the `MAX_INTERACTION_FREQUENCY` check on L523-529 of `updateCheckpointArray`, which in the worst case would prevent users from unstaking their tokens.
This also affects the `DelegationUtils.delegateVotingPower` and `DelegationUtils. undelegateVotingPower` functions.

Exploit Scenario: Suppose address `0xAlice` has a large stake that she has delegated to `0xBob`. The malicious users creates 20 addresses and stakes `1 wei` that gets delegated to `0xBob`. This prevents `0xAlice` from unstaking until the end of the epoch, however the sybil attack can be repeated to continue griefing.

Recommendation: Revise the `updateCheckpointArray` and `getValueAt` logic, possibly using binary search.

QSP-4 Race Condition on `setDaoApps()`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `StateUtils.sol`

Description: If `setDaoApps()` is called in a different block as the `contructor` of `Api3Pool` or in a different transaction, the an attacker can front-run the transaction to insert its own to `setDaoApps()`.

Recommendation: We recommend to either document this behavior or enforce it with by adding a call to `setDaoApps()` in the the constructor of `Api3Pool`.

QSP-5 Unchecked function arguments

Severity: *Informational*

Status: Fixed

File(s) affected: `StateUtils.sol`, `Api3Template.sol`

Description: The following functions should have additional input validation. Extra checks in constructors may mitigate faulty deployments that use default arguments and could require re-deployment.

1. `StateUtils.constructor` should check that `api3TokenAddress` is non-zero.
2. `Api3Template.constructor` should invoke `_ensureMiniMeFactoryIsValid(_minimeTokenFactory)` to ensure that `_minimeTokenFactory` is a contract.

Recommendation: Add `require` statements to each function.

QSP-6 Undocumented Constants

Severity: *Informational*

Status: Fixed

File(s) affected: `Api3Template.sol`

Description: The two `API3_VOTING_APP_ID` constants on L9-10 do not mention the strings used to derive the hashes. Therefore we cannot validate them.

Recommendation: Add documentation for the constants.

QSP-7 Privileged Roles and Ownership

Severity: *Informational*

Status: Fixed

File(s) affected: `ClaimUtils.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.

Following the inline documentation in `ClaimUtils.sol` L15, `payOutClaim()` can be "called by a claims manager to pay out an insurance claim".

Recommendation: This centralization of power needs to be made clear to the users (should be stated in the user documentation), especially depending on the level of privilege the contract allows to the claims manager.

QSP-8 Off-by-One Error in `updateCheckpointArray`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `StateUtils.sol`

Description: In this snippet:

```
if (checkpointArray.length + 1 >= MAX_INTERACTION_FREQUENCY)
{
    uint256 interactionTimestampMaxInteractionFrequencyAgo = snapshotBlockToTimestamp[checkpointArray[checkpointArray.length + 1 - MAX_INTERACTION_FREQUENCY].fromBlock];
    require(
        block.timestamp - interactionTimestampMaxInteractionFrequencyAgo > EPOCH_LENGTH,
    ERROR_FREQUENCY
    );
}
```

For example, if `MAX_INTERACTION_FREQUENCY` is set to 20, this seems to only allow 19 entries.

Recommendation: Change the if-conditional to `checkpointArray.length + 1 > MAX_INTERACTION_FREQUENCY`. Change the array access `checkpointArray[checkpointArray.length + 1 - MAX_INTERACTION_FREQUENCY]` to `checkpointArray[checkpointArray.length - MAX_INTERACTION_FREQUENCY]`.

QSP-9 Unclear Condition

Severity: *Undetermined*

Status: Fixed

File(s) affected: `DelegationUtils.sol`

Description: In `updateDelegatedVotingPower`, we have the following ternary statement on L117-119:

```
            newDelegatedTo = currentlyDelegatedTo > shares
                ? currentlyDelegatedTo - shares
                : 0;
```

It is not clear if `currentlyDelegatedTo > shares` can ever return false.

Recommendation: If the condition can never fail, the statement should be simplified. If not, inline documentation should explain when this could occur.

Automated Analyses

Slither

1. Slither reported several potentially dangerous strict equalities oof the form totalSharesCheckpoint2.fromBlock == block.number, however these were used in expected ways.
2. In `StateUtils.setStakeTarget`, the requirement `_stakeTarget >= 0` is redundant and can be removed as the variable is a `uint256`.
3. In several function (`TransferUtils.deposit`, TransferUtils.withdraw`,`ClaimUtils.payOutClaim`, and`TimelockUtils.depositWithVesting`), the return value of the ERC20`transfer`and`transferFrom`calls are not checked to be true. Note that all calls are to the`Api3Token` however.

Mythril

Mythril did not report any issues.

Code Documentation

1. In `GetterUtils.sol`, the comment on L10-11: "This method is used to implement the MiniMe interface for the Api3Voting app" is unclear. The Minime Token seems to support cloning and a `totalSupplyAt` function, neither of which are implemented here.
2. In the comment for `DelegationUtils.updateDelegatedVotingPower` on L95-96, it is stated: "User shares only get updated while staking, scheduling unstake or unstaking". However, shares are not updated when scheduling unstake (and `updateDelegatedVotingPower` is not called from `scheduleUnstake`).

Adherence to Best Practices

1. The literal `1_000_000_000` on `StateUtils.sol#L359` should be a declared constant. It is also not clear why this needs to be so large (equivalent to 1000% based on other constants).
2. In `StateUtils.sol`, the `updateCheckpointArray` has the `MAX_INTERACTION_FREQUENCY` check, whereas `updateAddressCheckpointArray` does not. It seems this is likely to save gas since these two functions are always invoked from the same external function. However, it should be documented that `updateAddressCheckpointArray` is not safe when called on its own (i.e., without `updateCheckpointArray`).
3. It is mentioned on L10 of `TransferUtils.sol` that `deposit` is used by `TimelockManager.sol`. Is there any reason a user should call this function directly? If not, it should be restricted to only be callable contracts that inherit from it. Similarly, should `TimelockUtils.depositWithVesting` be restricted to only callable from `TimelockManager` contracts?
4. In `StakeUtils.depositAndStake`, it is unclear why `userAddress` is needed as a parameters since it is required to always equal `msg.sender`.
5. In `Api3Template.sol`, L9-10 define two `API3_VOTING_APP_ID` constants, where one is commented out depending on the chain used (rinkeby or mainnet). This should instead be controlled by deployment scripts.
6. Some variables names are to long, for example `interactionTimestampMaxInteractionFrequencyAgo`.
7. Document what is the initial or minimum (mea culpa) `aprUpdateCoefficient` as this parameter defines the speed of APR update and is critical.

Test Results

Test Suite Results

```
payOutClaim
  Caller is claims manager
    Pool has enough funds
      ✓ pays out claim (116ms)
    Pool does not have enough funds
      ✓ reverts (57ms)
  Caller is not claims manager
    ✓ reverts

delegateVotingPower
  Delegate address is not zero
    Delegate address is not caller
      Delegate is not delegating
        User has not updated their delegation status less than reward epoch ago
          User did not have the same delegate
            Receiving user has not been delegated to too frequently
              ✓ delegates voting power (170ms)
            Receiving user has been delegated to too frequently
              ✓ reverts (3052ms)
          User had the same delegate
            ✓ reverts (140ms)
        User has updated their delegation status less than reward epoch ago
          ✓ reverts
      Delegate is delegating
        ✓ reverts
    Delegate address is caller
      ✓ reverts
  Delegate address is zero
    ✓ reverts

undelegateVotingPower
  User has delegated before
    User has not updated their delegation status less than reward epoch ago
      ✓ undelegates voting power (155ms)
    User has updated their delegation status less than reward epoch ago
      ✓ reverts
  User has not delegated before
    ✓ reverts

totalSupplyOneBlockAgo
  ✓ gets total supply one block ago (116ms)

userSharesAt
  ✓ gets user shares at (105ms)

userSharesAtWithBinarySearch
  ✓ gets user shares at (112ms)

userReceivedDelegationAt
  Searched block is within MAX_INTERACTION_FREQUENCY
    ✓ gets user's received delegation at the block (3212ms)
  Searched block is not within MAX_INTERACTION_FREQUENCY
    ✓ reverts (3791ms)

getDelegateAt
  ✓ gets delegate at (142ms)

getUserLocked
  It has been more than REWARD_VESTING_PERIOD since the genesis epoch
    User has staked
      ✓ returns the rewards paid to the user in the last REWARD_VESTING_PERIOD (3740ms)
    User has not staked
      ✓ returns 0 (969ms)
  It has not been more than REWARD_VESTING_PERIOD since the genesis epoch
    ✓ returns the rewards paid to the user (1168ms)

payReward
  Reward for the previous epoch has not been paid
    Pool contract is authorized to mint tokens
      Stake target is not zero
        Total stake is above target
```

```
          ✓ updates APR and pays reward (473ms)
        Total stake is below target
          ✓ updates APR and pays reward (187ms)
      Stake target is zero
        ✓ sets APR to minimum and pays reward (148ms)
    Pool contract is not authorized to mint tokens
      ✓ skips the payment and APR update (137ms)
  Rewards for multiple epochs have not been paid
    Pool contract is authorized to mint tokens
      ✓ updates APR and only pays the reward for the current epoch (174ms)
    Pool contract is not authorized to mint tokens
      ✓ skips the payment and APR update (135ms)
  Reward for the current epoch has been paid
    ✓ does nothing (167ms)

stake
  User has enough to stake
    User has a delegate and has staked before
      ✓ stakes and updates delegated voting power (145ms)
    User does not have a delegate
      ✓ stakes (68ms)
  User does not have enough to stake
    ✓ reverts

depositAndStake
  Caller is the beneficiary
    ✓ deposits and stakes (51ms)
  Caller is not the beneficiary
    ✓ reverts

scheduleUnstake
  User has enough staked to schedule unstake
    ✓ schedules unstake (69ms)
  User does not have enough staked to schedule unstake
    ✓ reverts

unstake
  Enough time has passed since the unstake scheduling
    The unstake has not expired
      User still has the tokens to unstake
        User has a delegate
          ✓ unstakes and updates delegated voting power (210ms)
        User does not have a delegate
          ✓ unstakes (142ms)
      User no longer has the tokens to unstake
        ✓ unstakes as much as possible (183ms)
    The unstake has expired
      ✓ reverts (127ms)
  Not enough time has passed since the unstake scheduling
    ✓ reverts (92ms)

unstakeAndWithdraw
  ✓ unstakes and withdraws (135ms)

constructor
  ✓ initializes with the correct parameters (112ms)

setDaoApps
  DAO apps are not set before
    DAO app addresses to be set are not zero
      ✓ sets DAO apps
    DAO app addresses to be set are zero
      ✓ reverts (45ms)
  DAO apps are set before
    Caller is primary Agent
      ✓ sets DAO apps (52ms)
    Caller is a random person
      ✓ reverts

setClaimsManagerStatus
  Caller is primary Agent
    ✓ sets claims manager status (63ms)
  Caller is not primary Agent
    ✓ reverts

setStakeTarget
  Caller is Agent
    Stake target to be set is smaller than or equal to 100,000,000
      ✓ sets stake target (65ms)
    Stake target to be set is larger than 100,000,000
      ✓ reverts
  Caller is not DAO Agent
    ✓ reverts

setMaxApr
  Caller is DAO Agent
    Max APR to be set is larger than or equal to min APR
      ✓ sets max APR (84ms)
    Max APR to be set is smaller than min APR
      ✓ reverts
  Caller is not DAO Agent
    ✓ reverts

setMinApr
  Caller is DAO Agent
    Min APR to be set is smaller than or equal to max APR
      ✓ sets min APR (67ms)
    Min APR to be set is larger than max APR
      ✓ reverts
  Caller is not DAO Agent
    ✓ reverts

setUnstakeWaitPeriod
  Caller is primary DAO Agent
    Unstake wait period to be set is larger than or equal to epoch length
      ✓ sets unstake wait period (49ms)
    Unstake wait period to be set is smaller than epoch length
      ✓ reverts (42ms)
  Caller is not primary DAO Agent
    ✓ reverts

setAprUpdateCoefficient
  Caller is DAO Agent
    APR update coefficient to be set is larger than 0 and smaller than or equal to 1,000,000,000
      ✓ sets APR update coefficient (65ms)
    APR update coefficient to be set is 0 or larger than 1,000,000,000
      ✓ reverts
  Caller is not DAO Agent
    ✓ reverts

setProposalVotingPowerThreshold
  Caller is primary DAO Agent
    Proposal voting power threshold to be set is smaller than or equal to 10,000,000
      ✓ sets proposal voting power threshold (56ms)
    Proposal voting power threshold to be set is larger than 10,000,000
      ✓ reverts
  Caller is not primary DAO Agent
    ✓ reverts

publishSpecsUrl
  ✓ publishes specs URL

updateLastVoteSnapshotBlock
  Caller is a Voting app
    ✓ updates lastVoteSnapshotBlock (49ms)
  Caller is not an authorized Api3Voting app
    ✓ reverts

depositWithVesting
  User has not received from this timelock manager
    Release end is later than release start
      Amount is not zero
        ✓ deposits with vesting (46ms)
      Amount is zero
        ✓ reverts
    Release end is not later than release start
      ✓ reverts
  User has received from this timelock manager before
    ✓ reverts (68ms)

updateTimelockStatus
  Timelock has started releasing
    Timelock has remaining tokens
      It is past release end
        ✓ updates timelock status (63ms)
      It is not past release end
        ✓ updates timelock status (169ms)
    Timelock does not have remaining tokens
      ✓ reverts (69ms)
  Timelock has not started releasing
```

```
      ✓ reverts (55ms)

  deposit
    ✓ deposits

  withdraw
    User has enough withdrawable funds
      ✓ updates user locked and withdraws (2701ms)
    User does not have enough withdrawable funds
      ✓ reverts (73ms)
    User does not have enough funds
      ✓ reverts


  82 passing (4m)


api3-voting$ npm run test

Contract: API3 Voting App delegation tests
Solidity stack traces only work with Solidity version 0.5.1 or higher.
    delegation for the voting working properly
      ✓ delegate to myself or to 0 address (63ms)
      ✓ need to delegate (243ms)
      ✓ delegate after already delegated
      ✓ undo delegate earlier then after a week
      ✓ undo delegate (170ms)
      ✓ delegate delegated (170ms)
      ✓ delegate delegated in a cycle (97ms)

  Contract: API3 Voting App
    normal token supply, common tests
      ✓ fails on reinitialization
      ✓ cannot initialize base app (55ms)
      ✓ checks it is forwarder
      ✓ can change required support (39ms)
      ✓ fails changing required support lower than minimum acceptance quorum (45ms)
      ✓ fails changing required support to 100% or more (98ms)
      ✓ can change minimum acceptance quorum (39ms)
      ✓ fails changing minimum acceptance quorum to greater than min support
  normal token supply, 0 decimals
    ✓ deciding voting is automatically executed (57ms)
    ✓ deciding voting is automatically executed (long version) (42ms)
    ✓ execution scripts can execute multiple actions (57ms)
    ✓ execution script can be empty (45ms)
    ✓ execution throws if any action on script throws (85ms)
    ✓ forwarding creates vote (50ms)
    creating vote
      ✓ has correct state
      ✓ fails getting a vote out of bounds
      ✓ changing required support does not affect vote required support (124ms)
      ✓ changing min quorum doesnt affect vote min quorum (96ms)
      ✓ holder can vote (42ms)
      ✓ holder can modify vote (77ms)
      ✓ token transfers dont affect voting (47ms)
      ✓ throws when non-holder votes
      ✓ throws when voting after voting closes
      ✓ can execute if vote is approved with support and quorum (101ms)
      ✓ cannot execute vote if not enough quorum met (63ms)
      ✓ cannot execute vote if not support met (95ms)
      ✓ vote can be executed automatically if decided (53ms)
      ✓ vote can be not executed automatically if decided (64ms)
      ✓ cannot re-execute vote (82ms)
      ✓ cannot vote on executed vote (68ms)
  normal token supply, 2 decimals
    ✓ deciding voting is automatically executed (80ms)
    ✓ deciding voting is automatically executed (long version) (64ms)
    ✓ execution scripts can execute multiple actions (79ms)
    ✓ execution script can be empty (71ms)
    ✓ execution throws if any action on script throws (135ms)
    ✓ forwarding creates vote (59ms)
    creating vote
      ✓ has correct state (43ms)
      ✓ fails getting a vote out of bounds
      ✓ changing required support does not affect vote required support (234ms)
      ✓ changing min quorum doesnt affect vote min quorum (127ms)
      ✓ holder can vote (64ms)
      ✓ holder can modify vote (119ms)
      ✓ token transfers dont affect voting (85ms)
      ✓ throws when non-holder votes (41ms)
      ✓ throws when voting after voting closes (56ms)
      ✓ can execute if vote is approved with support and quorum (131ms)
      ✓ cannot execute vote if not enough quorum met (94ms)
      ✓ cannot execute vote if not support met (94ms)
      ✓ vote can be executed automatically if decided (52ms)
      ✓ vote can be not executed automatically if decided (69ms)
      ✓ cannot re-execute vote (69ms)
      ✓ cannot vote on executed vote (68ms)
  normal token supply, 18 decimals
    ✓ deciding voting is automatically executed (72ms)
    ✓ deciding voting is automatically executed (long version) (59ms)
    ✓ execution scripts can execute multiple actions (81ms)
    ✓ execution script can be empty (58ms)
    ✓ execution throws if any action on script throws (140ms)
    ✓ forwarding creates vote (66ms)
    creating vote
      ✓ has correct state
      ✓ fails getting a vote out of bounds
      ✓ changing required support does not affect vote required support (180ms)
      ✓ changing min quorum doesnt affect vote min quorum (122ms)
      ✓ holder can vote (65ms)
      ✓ holder can modify vote (110ms)
      ✓ token transfers dont affect voting (68ms)
      ✓ throws when non-holder votes (40ms)
      ✓ throws when voting after voting closes (41ms)
      ✓ can execute if vote is approved with support and quorum (113ms)
      ✓ cannot execute vote if not enough quorum met (62ms)
      ✓ cannot execute vote if not support met (85ms)
      ✓ vote can be executed automatically if decided (48ms)
      ✓ vote can be not executed automatically if decided (73ms)
      ✓ cannot re-execute vote (68ms)
      ✓ cannot vote on executed vote (102ms)
  normal token supply, 26 decimals
    ✓ deciding voting is automatically executed (84ms)
    ✓ deciding voting is automatically executed (long version) (84ms)
    ✓ execution scripts can execute multiple actions (90ms)
    ✓ execution script can be empty (64ms)
    ✓ execution throws if any action on script throws (201ms)
    ✓ forwarding creates vote (109ms)
    creating vote
      ✓ has correct state
      ✓ fails getting a vote out of bounds
      ✓ changing required support does not affect vote required support (165ms)
      ✓ changing min quorum doesnt affect vote min quorum (129ms)
      ✓ holder can vote (65ms)
      ✓ holder can modify vote (103ms)
      ✓ token transfers dont affect voting (68ms)
      ✓ throws when non-holder votes (58ms)
      ✓ throws when voting after voting closes (63ms)
      ✓ can execute if vote is approved with support and quorum (156ms)
      ✓ cannot execute vote if not enough quorum met (84ms)
      ✓ cannot execute vote if not support met (91ms)
      ✓ vote can be executed automatically if decided (52ms)
      ✓ vote can be not executed automatically if decided (69ms)
      ✓ cannot re-execute vote (98ms)
      ✓ cannot vote on executed vote (79ms)
  wrong initializations
    ✓ fails if min acceptance quorum is greater than min support
    ✓ fails if min support is 100% or more (57ms)
  empty token
    ✓ fails creating a vote if token has no holder (63ms)
  token supply = 1
    ✓ new vote cannot be executed before voting (108ms)
    new vote parameters
      ✓ creating vote as holder executes vote (if _canExecute param says so) (85ms)
      ✓ creating vote as holder doesn't execute vote if _canExecute param doesn't says so (80ms)
  token supply = 3
    ✓ new vote cannot be executed before holder2 voting (181ms)
    ✓ creating vote as holder2 executes vote (92ms)
  changing token supply
    ✓ uses the correct snapshot value if tokens are minted afterwards (76ms)
    ✓ uses the correct snapshot value if tokens are minted in the same block (79ms)
  before init
    ✓ fails creating a vote before initialization
    ✓ fails to forward actions before initialization
  isValuePct unit test
    ✓ tests total = 0
    ✓ tests value = 0
    ✓ tests pct ~= 100
    ✓ tests strict inequality
```

```
  119 passing (2m)
```

Code Coverage

```
cd pool/ & npm run test:coverage
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
Api3Pool.sol	100	100	100	100	
ClaimUtils.sol	100	100	100	100	
DelegationUtils.sol	100	100	100	100	
GetterUtils.sol	100	100	100	100	
RewardUtils.sol	100	100	100	100	
StakeUtils.sol	100	100	100	100	
StateUtils.sol	100	100	100	100	
TimelockUtils.sol	100	100	100	100	
TransferUtils.sol	100	100	100	100	
contracts/auxiliary/interfaces/	100	100	100	100	
IApi3Pool.sol	100	100	100	100	
IApi3Token.sol	100	100	100	100	
ITimelockManager.sol	100	100	100	100	
contracts/auxiliary/interfaces/v0.8.**2/**	100	100	100	100	
IApi3Token.sol	100	100	100	100	
IERC20.sol	100	100	100	100	
ITimelockManager.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IApi3Pool.sol	100	100	100	100	
IClaimUtils.sol	100	100	100	100	
IDelegationUtils.sol	100	100	100	100	
IGetterUtils.sol	100	100	100	100	
IRewardUtils.sol	100	100	100	100	
IStakeUtils.sol	100	100	100	100	
IStateUtils.sol	100	100	100	100	
ITimelockUtils.sol	100	100	100	100	
ITransferUtils.sol	100	100	100	100	
contracts/mock/	100	100	100	100	
MockApi3Staker.sol	100	100	100	100	
MockApi3Voting.sol	100	100	100	100	
All files	**100**	**100**	**100**	**100**	

```
cd api3-voting/ & npm run coverage
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	97.96	95.65	91.3	97.96	
Api3Voting.sol	97.96	95.65	91.3	97.96	154,216
contracts/interfaces/	100	100	100	100	
IApi3Pool.sol	100	100	100	100	
All files	**97.96**	**95.65**	**91.3**	**97.96**	

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

```
adcfa08eb5348d3c05244ea53a493d122f0d26fc060fddcfea275098f0fe2ee3 ./Api3Template.sol
8138006671e707b007e7af7e4a3039532b72835d416d473933448512cd9cb3ed ./Api3Voting.sol
32eee74c77361eba3f58a94a7b58ba13d91d1c5338cd81401d1cc3fa76daf7f6 ./contracts/GetterUtils.sol
8cbc970457796ce6fbc70333862261c39217b5a86565fa609e18e7da9d09a21f ./contracts/TransferUtils.sol
e2d70e8f646070d54ee6ab28216733def0483ef855b17e92324c866e8e0ca749 ./contracts/StakeUtils.sol
678030af6afa7474836bd5f6fec3ee784941a4e3df2e5f0ab51ccc78cb81ff22 ./contracts/Api3Pool.sol
c384dce5ac5a9dac3c224841c577d1561a5f31fe897999acb5197ebd96b80b03 ./contracts/DelegationUtils.sol
ca58d3a279b934293788cf0fa2c987769b10d7ee529bf532b48faeb5838f19f4 ./contracts/TimelockUtils.sol
398592d423b9eb1b5051303d6ff29220e499840485c87c29f347fce3558d33d9 ./contracts/ClaimUtils.sol
b624b27216e0cace4804ad70aea88bff46da5238487c51827577cb1566f9590a ./contracts/StateUtils.sol
5956d48673d4c917c6a6a1d42fbdacfedb7ddd0fcefb9a99e4c8ab540f3317c5 ./contracts/RewardUtils.sol
8955d658abbc7fc99c954486e98b80e82159f2c538a8f85f0e5373a83200eb68 ./contracts/interfaces/IStateUtils.sol
5fb49bf8ff93c8eebcc3bcc2e6bf6be9eace90aab290a469a5b29c1644283101 ./contracts/interfaces/ITransferUtils.sol
32b242eacd40a12808f8b5404d616ac34a75dd0a7b77ab6d2ee2d050e755776a ./contracts/interfaces/IStakeUtils.sol
df2aa0a1500da7b79794de9740f7fa17c2adfa3d11cf0d82de2cb030d61cce51 ./contracts/interfaces/IApi3Pool.sol
33bc2e85777d27427ea375b7a7ed3a5070b89d6980589fec3d0a0e9f74d6aa79 ./contracts/interfaces/ITimelockUtils.sol
91a9fd3bbd620eac79684d48bdfa1953d67083e691c815845ea0b04db313c502 ./contracts/interfaces/IDelegationUtils.sol
926fec1ecbec1d0e521837209e1b14d98450c5bb8bb0330b38d58006c8bf66b2 ./contracts/interfaces/IRewardUtils.sol
b331158ded472ef51d27078d034a1f8c5b34177da4d77355271c2aceaf7b43b5 ./contracts/interfaces/IGetterUtils.sol
b747cd98f0cb1a11f9c424f8f21a105093e4cd12c219b2e7a4ab6d33b4f53f9f ./contracts/interfaces/IClaimUtils.sol
```

Tests

```
ae31e818e49ccd602fc8cbdf62337c5e06fe7bf0595ff235b0bf0bfa7f7cbaa2 ./api3template.js
624ce605cc55e321b8c4066f61015ee4ce225111c0660eb73ea1add276bd7804 ./voting.js
c6aee5753d46b96297210a61aaa5e362a40f6d1341d9da68c76befea622c610b ./delegation-integration.js
5dda8ffa80edb2bd970683edc7389f981488a28468a679a886065e176d0ff5a3 ./errors.js
d2ba2048b478df1eb4864e105c70eb4f3f66a5037513a9c641ecd93b1b20d29c ./test/DelegationUtils.sol.js
d7166ab8d2095f3e29cdf4db7f666aff104a34f81238a7f8788402dc03eff42b ./test/TransferUtils.sol.js
6cdef3b4abc62be9f984112ac0fbd01f12ee81f4f4dccc349fbfb2ad42817706 ./test/ClaimUtils.sol.js
c380164a0b5b52c2fa6965b3c8d31b1b8c0608496a210de9cd9be672239ee677 ./test/RewardUtils.sol.js
62530c0fc1ab4e58ffb7ff4a6f5e42670de8de8c3fd913e63e01b7038d6a80e9 ./test/StakeUtils.sol.js
ab3d68d53c8e58f5fb0cd8b2b4eb451ff01e52611c49c4fb1704f95e1766a536 ./test/StateUtils.sol.js
3f3f0c189a11dd375d03761dc7c486a941cd749c3009ec6b4892ed26f4ad1fca ./test/GetterUtils.sol.js
935163bbcfad488df062584e05f96984afcb8e2fcdbce23120b09d8e0071260f ./test/TimelockUtils.sol.js
```

Changelog

- 2021-05-18 - Initial report
- 2021-06-18 - Re-audit (d3db4e7)

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.