Quantstamp Security Assessment Certificate

July 27th 2020 — Quantstamp Verified

Skale Network

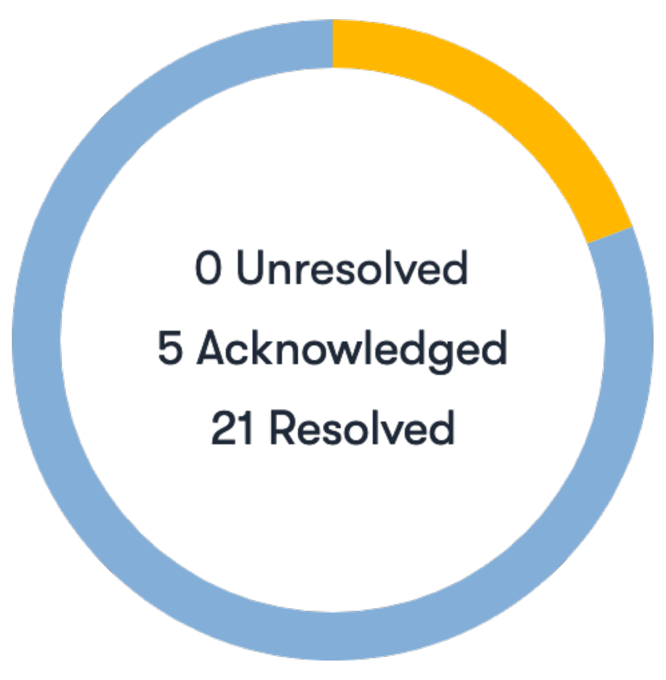
This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Smart Contracts
Auditors	Alex Murashkin, Senior Software Engineer Kacper Bąk, Senior Research Engineer Ed Zulkoski, Senior Security Engineer
Timeline	2020-02-04 through 2020-07-10
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md Spec
Source Code	

Repository	Commit
skale-manager	remediation-3
skale-manager	50c8f4e

Total Issues	**26** (21 Resolved)
High Risk Issues	**3** (3 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	**11** (8 Resolved)
Informational Risk Issues	**7** (5 Resolved)
Undetermined Risk Issues	**4** (4 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

The scope of the audit is restricted to the set of files outlined in the *Quantstamp Audit Breakdown* section.
While reviewing the given files at the commit `50c8f4e`, we identified three issues of high severity, seven issues of low severity, and four issues of informational severity. In addition, we made several suggestions with regards to code documentation, adherence to best practices, and adherence to the specification. We recommend resolving the issues and improving code documentation before shipping to production.
Update: While reviewing the diff `50c8f4e..remediation-3`, we marked some of the issues as resolved or acknowledged, depending on whether a code change was made. Some findings remained marked as "Unresolved" (such as **QSP-1**, **QSP-10**, and **QSP-11**): we recommend taking a look at these as we believe these were not fully addressed or still impose risks. In addition, we found 12 new potential issues of varying levels of severity. For the commit `remediation-3`, the new issue list beings with `QSP-15`, and line numbers now refer to the `remediation-3` commit. The severity of some findings remained as "undetermined" due to the lack of documentation. Moreover, we made additional documentation and best practices recommendations which were placed at the end of the report after the main findings.
We recommend addressing all the issues before running in production.
Update: We reviewed the fixes provided in the following separate commits/PRs:

1. `1da7bbd` (`https://github.com/skalenetwork/skale-manager/pull/267`)
2. `8c6a218` (`https://github.com/skalenetwork/skale-manager/pull/258`)
3. `8652d74` (`https://github.com/skalenetwork/skale-manager/pull/264/commits/8652d743fa273c68664c1acc972067d83b28f098`)
4. `0164b22` (`https://github.com/skalenetwork/skale-manager/pull/264/commits/0164b22436d8c10e22a202ff257581b76e038dec`)
5. `7e040bf` (`https://github.com/skalenetwork/skale-manager/pull/229`)
6. `bd17697` (`https://github.com/skalenetwork/skale-manager/pull/224`)
7. `c671839` (`https://github.com/skalenetwork/skale-manager/blob/c6718397cbbe7f9520b3c7ff62aa5bd1b0df27f5/contracts/ContractManager.sol#L51`)

All main findings (`QSP-1..QSP-26`) from the original commit - `50c8f4e` and the re-audit commit - `remediation-3` - were addressed in the commits above. Some best practices suggestions and documentation issues were addressed, but some were not. Code coverage could also use some improvement.

ID	Description	Severity	Status
QSP-1	Potentially Unsafe Use of Arithmetic Operations	High	Fixed
QSP-2	Ability to register Address that already exists	Low	Fixed
QSP-3	Free Tokens for Owner from Testing Code	High	Fixed
QSP-4	Validator Denial-of-Service	High	Fixed
QSP-5	Unlocked Pragma	Low	Fixed
QSP-6	Use of Experimental Features	Low	Fixed
QSP-7	Centralization of Power	Low	Acknowledged
QSP-8	Transaction-Ordering Dependency in Validator Registration	Low	Fixed
QSP-9	Unintentional Locking of Tokens upon Cancelation	Low	Fixed
QSP-10	Validator Registration "Spamming"	Low	Acknowledged
QSP-11	Misusing `require()/assert()/revert()`	Informational	Acknowledged
QSP-12	Stubbed Functions in DelegationService	Informational	Fixed
QSP-13	Unhandled Edge Case in Validator Existence Check	Informational	Fixed
QSP-14	Potentially Unsafe Use of Loops	Informational	Fixed
QSP-15	Denial-of-Service (DoS)	Medium	Fixed
QSP-16	Denial-of-Service (DoS)	Low	Fixed
QSP-17	Potentially Unsafe Use of Loops (2)	Low	Acknowledged
QSP-18	Lack of Array Length Reduction	Low	Fixed
QSP-19	Unclear Purpose of the Assignment	? Undetermined	Fixed
QSP-20	Unclear State List for `isTerminated(...)`	? Undetermined	Fixed
QSP-21	Unclear Intention for `totalApproved` in `TokenLaunchManager`	? Undetermined	Fixed
QSP-22	Potential Violation of the Spec	Informational	Acknowledged
QSP-23	Inconsistent Behaviour of `addToLockedInPendingDelegations(...)`	? Undetermined	Fixed
QSP-24	Error-prone Logic for `delegated.mul(2)`	Informational	Fixed
QSP-25	Event Emitted Regardless of Result	Informational	Fixed
QSP-26	Missing Input Validation	Low	Fixed

Quantstamp Audit Breakdown

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

```
* ContractManager.sol
* Permissions.sol
* SkaleToken.sol
* interfaces/ISkaleToken.sol
* interfaces/delegation/IDelegatableToken.sol
* interfaces/delegation/IHolderDelegation.sol
* interfaces/delegation/IValidatorDelegation.sol
* interfaces/tokenSale/ITokenSaleManager.sol
* ERC777/LockableERC777.sol
* thirdparty/BokkyPooBahsDateTimeLibrary.sol
* delegation/*
* utils/*
```

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:

- Maian commit sha: ab387e1
- Truffle v4.1.12
- Ganache v1.1.0
- SolidityCoverage v0.5.8
- Mythril v0.2.7
- Securify None
- Slither v0.6.6

Steps taken to run the tools:

1. Cloned the MAIAN tool: `git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian`
2. Ran the MAIAN tool on each contract: `cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol`
3. Installed Truffle: `npm install -g truffle`
4. Installed Ganache: `npm install -g ganache-cli`
5. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
6. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
7. Installed the Mythril tool from Pypi: `pip3 install mythril`
8. Ran the Mythril tool on each contract: `myth -x path/to/contract`
9. Ran the Securify tool: `java -Xmx6048m -jar securify-0.1.jar -fs contract.sol`
10. Installed the Slither tool: `pip install slither-analyzer`
11. Run Slither from the project directory: `slither .s`

Findings

QSP-1 Potentially Unsafe Use of Arithmetic Operations

Severity: *High Risk*

Status: Fixed

File(s) affected: `(multiple)`

Description: Some arithmetic operations in the project may lead to integer underflows or underflows. Examples include (line numbers are for commit `50c8f4e`):

1. `Distributor.sol`, `L72`: `amount - amount * feeRate / 1000` may underflow, which will break all distributions associated with the validator - Not fixed (`contracts/delegation/Distributor.sol`, `L210` (commit `remediation-3`): `uint bounty = amount - fee;` should use SafeMath). **Fixed** in commit `1da7bbd`.
2. `Distributor.sol`, `L77`: a possibility of an underflow - **Fixed**
3. `Distributor.sol`, `L101-105`: unsafe multiplication and addition, there may be an overflow if `msr` is set to too high or `validatorNodes.length` becomes too high - **Fixed**
4. `SkaleBalances.sol`, `L89` - **Fixed**
5. `ValidatorService.sol`: `L166`, `L178`: unclear what the bounds of `MSR` are, however, may overflow under certain conditions - Not fixed in `remediation-3`: should still use `SafeMath` for the multiplication. **Fixed** in commit `1da7bbd`.
6. `DelegationRequestManager.sol`: `L74-75`: this logic should be rewritten using `SafeMath` operations - **Fixed**.
7. `TimeHelpers.sol`: while we do not see immediate issues, we still recommended using `SafeMath` across the board - **Fixed**.
8. `ERC777/LockableERC777.sol`: the addition `locked + amount` should be performed using `SafeMath` - **Fixed**.
9. `DelegationController.sol`, `L67` and `L71`: while we do not see immediate issues, we still recommended using `SafeMath` across the board - **Fixed**.

Recommendation:

1. Use SafeMath for all arithmetic operations
2. Add input validation for any values partaking in math operations, such as validating `feeRate` in `Distributor.sol` and `MSR`.

QSP-2 Ability to register Address that already exists

Severity: *Low Risk*

Status: Fixed

File(s) affected: `delegation/ValidatorService.sol, delegation/DelegationService.sol`

Description: `ValidatorService.sol`, `L92`: It appears that a user can invoke `requestForNewAddress(...)` where `newValidatorAddress` is an already existing address.

Exploit Scenario: There are two cases. If `newValidatorAddress` is the validator's current address, the operation would effectively be a no-op. If it is a different validator's address, the operation would overwrite the old `validatorId` in the list. As one consequence, this would break the `_validatorAddressToId` mapping, which would consequently invalidate any function that uses `getValidatorId()` on `L197`.
This relates to the functions of the same name in `DelegationService.sol`.

Recommendation: Fixing the logic to disallow overwriting behavior.

QSP-3 Free Tokens for Owner from Testing Code

Severity: *High Risk*

Status: Fixed

File(s) affected: `SkaleToken.sol`

Description: `SkaleToken.sol`, `L47-54`: the code labeled as "// TODO remove after testing" issues free tokens for the owner, which is, likely, undesired.

Recommendation: Remove the testing code.

QSP-4 Validator Denial-of-Service

Severity: *High Risk*

Status: Fixed

File(s) affected: `delegation/DelegationService.sol`, `delegation/DelegationRequestManager.sol`

Description: It appears that a third-party can run a denial-of-service attack which causes some methods to run out of gas upon execution.

Exploit Scenario: Using the `delegate()` method of `DelegationService`, an attacker submits multiple delegations of a low amount to a given Validator A. The array `_activeByValidator[validatorId]` becomes big enough to cause the methods `getDelegationsForValidator`, `getActiveDelegationsByValidator`, and `getDelegationsTotal` run out of gas. While it is unclear what a minimum validation amount is (`DelegationRequestManager.sol`, `L62`) and the validator has to be trusted to be able to pick it up (`DelegationRequestManager.sol`, `L65`), it looks like if the attacker has a high balance, the attack is possible.

Recommendation: Perform a gas analysis to identify the number of validations such that `_activeByValidator[validatorId]` does not run out of gas, and cap the number of delegations or increase the minimum price accordingly.

QSP-5 Unlocked Pragma

Severity: *Low Risk*

Status: Fixed

File(s) affected: `(multiple)`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.5.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked."

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

QSP-6 Use of Experimental Features

Severity: *Low Risk*

Status: Fixed

File(s) affected: `delegation/*.sol`

Description: The project is using `pragma experimental ABIEncoderV2`, which enables an experimental version of the ABI encoder. Experimental features may contain bugs, such as this.

Recommendation: We recommend incrementing and fixing `pragma solidity` at or beyond `0.5.7`, staying up-to-date with regards to any new `ABIEncoderV2`-related issues, and addressing them in a timely manner.

QSP-7 Centralization of Power

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `(multiple)`

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

1. Since `allow()` permits the owner of the contract to interact with any of its functions, the owner can grief any wallet by setting an arbitrarily high `timeLimit`
2. The owner can invoke `tokensReceived()` and update the balance of any wallet as desired.

These issues exist for essentially any function using `allow`, which includes most `Delegation*` contracts. For example, `DelegationController.setDelegationAmount()` can set arbitrary delegation amounts.
Apart from these:

1. The owner can change `setDelegationPeriod()` at any time, which could influence upcoming distributions of shares in `Distributor.distribute()` via `L100`: `uint multiplier = delegationPeriodManager.stakeMultipliers(delegation.delegationPeriod);`
2. In `ValidatorService`, the owner may censor validators via `enableValidator()` and `disableValidator()`.

Recommendation:

1. Potentially, removing the `isOwner()` conditional from the `allow` modifiers.
2. Make the centralization of power clear to end-users.

Update: Not fixed, the provided rationale: "SKALE Network plans to communicate clearly the plans for admin control and how this will be gradually decentralized. More importantly, admin is economically incentivized against this attack".

QSP-8 Transaction-Ordering Dependency in Validator Registration

Severity: *Low Risk*

Status: Fixed

File(s) affected: `delegation/DelegationService.sol`

Description: In `DelegationService.sol`, there is transaction-ordering dependency between `registerValidator()` on `L161` and `DelegationService.linkNodeAddress()` `L180`.

Exploit Scenario: Anyone can grief someone attempting to `registerValidator()` by calling `linkNodeAddress()` and setting `nodeAddress` to the `msg.sender` value from the `registerValidator()` transaction. The registration will then fail due to `L67` of `ValidatorService.sol`: `require(_validatorAddressToId[validatorAddress] == 0, "Validator with such address already exists");`.

Recommendation: Transaction-ordering is often difficult to fix without introducing changes to system design. However, we are just bringing the potential issue to the team's attention.

QSP-9 Unintentional Locking of Tokens upon Cancelation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `delegation/TokenState.sol`

Description: `TokenState.sol`, `L205`: the logic enables the possibility of ending up having locked more tokens than expected.

Exploit Scenario:

1. A user gets 20 tokens from the token sale (and they are, therefore, locked)
2. The user receives a transfer of 5 tokens from someone else
3. The user requests a delegation of 25 tokens
4. The user's delegation gets canceled
5. The user ends up having 25 locked tokens instead of the original 20 tokens.

Recommendation: Reconsider the logic in `TokenState.sol`.

QSP-10 Validator Registration "Spamming"

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `delegation/DelegationService.sol`

Description: The `registerValidator()` method is open to the public: anyone can call it and spam the network with registering arbitrary addresses as validators. This will pollute the contract with validator entries that do not do anything yet have an impact on the smart contract's state. For instance, `numberOfValidators` would become high but this would not reflect the actual state of the network.

Recommendation: We suggest adding a mechanism for preventing adding arbitrary validator registrations. Alternatively, making it so that adding new validators does not affect the contract state (e.g., calculate `numberOfValidators` differently, e.g., only adding up trusted validators).
Update: Unresolved, the rationale: "validators must pay for gas to register, so this naturally reduces DoS. Receiving a validator ID does not allow the user to do anything special - unless they are a part of whitelist." However, as of `remediation-3`, there is an added issue, see **QSP-16**.
Update: The implications of this issue are mitigated in `8c6a218`.

QSP-11 Misusing `require()`/`assert()`/`revert()`

Severity: *Informational*

Status: Acknowledged

File(s) affected: `thirdparty/BokkyPooBahsDateTimeLibrary.sol, delegation/DelegationController.sol`

Description: `require()`, `revert()`, and `assert()` all have their own specific uses and should not be switched around.

- `require()` checks that certain preconditions are true before a function is run.
- `revert()`, when hit, will undo all computation within the function.
- `assert()` is meant for checking that certain invariants are true. An `assert()` failure implies something that should never happen, such as integer overflow, has occurred.

Recommendation:

1. `BokkyPooBahsDateTimeLibrary.sol`: use `assert` instead of `require` on lines `217`, `232`, `236`, `240`, `244`, `248`, `262`, `277`, `281`, `285`, `289`, and `293`
2. `DelegationController.sol`, use `assert` instead of `require` on lines `L134`,`L165`, and `L193`.

Update: only fixed in `DelegationController.sol`. However, this is up to the Skale Labs team to decide whether the date-time library should be fixed or not.
Update: the team made a decision to not update the date-time library, which is fair in this case.

QSP-12 Stubbed Functions in DelegationService

Severity: *Informational*

Status: Fixed

File(s) affected: `delegation/DelegationService.sol`

Description: There are several stubbed functions in this contract, e.g., `listDelegationRequests()` on `L90` or `getDelegationRequestsForValidator()` on `L156`, which seem important, particularly, since no events are emitted that would easily alert a delegator that a new delegation offer is associated with them.

Recommendation: Consider implementing the stubbed functions.

QSP-13 Unhandled Edge Case in Validator Existence Check

Severity: *Informational*

Status: Fixed

File(s) affected: `delegation/DelegationService.sol`

Description: `ValidatorService.sol`, `L181`: `validatorExists()` incorrectly returns `true` if the input is `0`, which could affect any function with the modifier `checkValidatorExists()`. Note that `L69` `validatorId = ++numberOfValidators;` defines the first validator and ID 1.

Recommendation: Return `false` when the validator address argument is provided as `0`.

QSP-14 Potentially Unsafe Use of Loops

Severity: *Informational*

Status: Fixed

File(s) affected: `(multiple)`

Description: "For" loops are used throughout to iterate over active delegations or distribution shares. Examples include (with the respective. bounds):

- `DelegationController.sol`: `L75-79, L115-128, L132-137, L182-187, L191-197`: `_activeByValidator[validatorId].length`
- `DelegationController.sol`: `L154-159, L163-169`: `_delegationsByHolder[holderAddress].length`
- `delegation/DelegationService.sol`: `L103-105`: `shares.length`
- `delegation/Distributor.sol`: `L95-107, L109-117`: `activeDelegations.length`
- `delegation/TokenState.sol`: `L63-69, L76-82`: `delegationIds.length`
- `delegation/TokenState.sol`: `L161-163`: `_endingDelegations[holder].length`
- `delegation/TokenState.sol`: `L246-256`: `_endingDelegations[delegation.holder].length`

If the value of a loop's upper bound is very high, it may cause a transaction to run out of gas and potentially lead to other higher-severity issues, such as QSP-4.

Recommendation: We recommend running gas analysis for the loops. This would identify the viable bounds for each loop, which consequently could be used for adding constraints to prevent overflows.
Update: this instance was fixed, however, there are now more potential issues with loops, see `QSP-17`.

QSP-15 Denial-of-Service (DoS)

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `ConstantsHolder.sol`

Description: `ConstantsHolder.sol, L165`: The owner can overwrite the `launchTimestamp` at any point using `setLaunchTimestamp()`. It is unclear why this functionality is necessary, but if abused (with a low likelihood) and set to a time very far into the future, it could lock the two `Distributor` functions `withdrawBounty()` and `withdrawFee()`.

Recommendation: Consider removing the ability to override the launch timestamp.
Update: fixed in `8652d74` and `0164b22`.

QSP-16 Denial-of-Service (DoS)

Severity: *Low Risk*

Status: Fixed

File(s) affected: `ValidatorService.sol`

Description: `L127`: `getTrustedValidators(...)` iterates over all `numberOfValidators` validator entries. However, as per `QSP-10`, any party can arbitrarily submit `registerValidator(...)` requests that increment `numberOfValidators`. While block gas limits should not be an issue for external view methods, the method may end up iterating over many unconfirmed validators, which could extend the load or execution time of the Ethereum node that executes the method.

Recommendation: Consider tracking trusted validators differently, in order to avoid iterating over unconfirmed validators.

QSP-17 Potentially Unsafe Use of Loops (2)

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `delegation/DelegationController.sol`, `delegation/ValidatorService.sol`, `delegation/TokenState.sol`

Description: We found the following locations in code where for loops are still used:

1. In `DelegationController.sol`, there are two invocations of the `processAllSlashes(...)` method: `processAllSlashes(holder);`, `L339`: `processAllSlashes(msg.sender)`. They both use the overloaded instance of `processAllSlashes` that does not take `limit` as an input parameter:

   ```
   function processAllSlashes(address holder) public {
           processSlashes(holder, 0);
       }
   ```

   This eventually calls `processSlashesWithoutSignals(...)` with the limit set to `0` (or, unlimited).
   In addition, `L201`: `processSlashesWithoutSignals(holder)`, `L231`: `processSlashesWithoutSignals(msg.sender);` , and `L280`: `processSlashesWithoutSignals(delegations[delegationId].holder);` also call `processSlashesWithoutSignals(...)` with the limit set to `0`.
   If the limit is set to `0`, `L899` sets `_slashes.length` as the `end` bound for the loop. If `_slashes.length` happens to contain too many slashes, there is a chance for the loop at `L903 for (; index < end; ++index) {` to cause a "block gas limit exceeded" issue.
2. Similarly, in `DelegationController.sol`, `sendSlashingSignals(...)` has a for-loop: if there are too many slashing signals, a block gas limit issue could be hit.
3. In `delegation/ValidatorService.sol`, `L305`and `L337` contain use of potentially unbounded loops. Block gas limits are more difficult to exceed because it requires a specific validator to register too many node addresses, however, we are highlighting it for awareness.
4. `delegation/TokenState.sol`: iterations over the lockers could, in theory, lead to the "block out of gas exceptions". However, since the methods are `ownerOnly`, the risk is pretty low assuming the responsibility of the owner.

Recommendation: Perform gas analysis and define mitigation strategies for the loops in `DelegationController.sol` and `ValidatorService.sol` to ensure the block gas limit is never hit, and none of the contract methods are blocked due to this. Cap the number of slashings if possible.
Update:
The Skale Labs team provided the following responses (quoted):

1. Issue is known and acceptable because initial network phases will operate with slashing off to verify very low probability of slashing events. It is expected that slashing will be rare event, and in the exceptional case of many slashing events, it is possible to batch executions.
2. Same as above in 1.
3. Acknowledged and optimization is planned. For initial network phases it will be difficult to exceed as the Quantstamp team notes.
4. There will only be 3-4 total lockers.

QSP-18 Lack of Array Length Reduction

Severity: *Low Risk*

Status: Fixed

File(s) affected: `delegation/ValidatorService.sol`, `delegation/TokenLaunchLocker.sol`

Description: There are two instances when an array item is removed but the array length is not explicitly reduced:

1. `ValidatorService.sol`, `L212`: The lack of length reduction of `validatorNodes.length` is likely unintentional after `delete validators[validatorId].nodeIndexes[validatorNodes.length.sub(1)];` - **Fixed** in `7e040bf`.
2. `delegation/TokenLaunchLocker.sol`, `L237`: `deletePartialDifferencesValue(...)`: may also need to reset the array lengths for `sequence.addDiff` and `sequence.subtractDiff`. **Fixed** (code removed).

Recommendation: Add length decrements where appropriate.

QSP-19 Unclear Purpose of the Assignment

Severity: *Undetermined*

Status: Fixed

File(s) affected: `delegation/DelegationController.sol`

Description: In `delegation/DelegationController.sol`, `L609`: the purpose of the assignment `_firstUnprocessedSlashByHolder[holder] = _slashes.length;` is not clear.

Recommendation: Clarify the purpose. Improve the developer documentation.
Update: the team has provided a clarification:

```
`_slashes` is a list of all slashing events that have occurred in the network.
When skale-manager calculates the amount of locked tokens, it iterates over this list. Each item is processed
only once. When a token holder creates their first delegation, it is set first as an unprocessed slash as
`_slashes.length` to avoid processing of all slashed before that. This obviously does not affect the holder.
```

QSP-20 Unclear State List for `isTerminated(...)`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `delegation/DelegationController.sol`

Description: Currently, in `delegation/DelegationController.sol`, `isTerminated()` includes two states: `COMPLETED` and `REJECTED`. However, there is also the `CANCELED` state, which seems to be unaccounted for, and it is unclear if this is intentional. Note that this affects the `isLocked()` function below.

Recommendation: Clarifying the intention and accounting for the `CANCELED` state as needed.
Update: the logic has been removed from the code in `1da7bbd`.

QSP-21 Unclear Intention for `totalApproved` in `TokenLaunchManager`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `delegation/TokenLaunchManager.sol`

Description: It is not clear if the require-statement on `L60`: `require(totalApproved <= getBalance(), "Balance is too low");` will work as intended. The `totalApproved` only accounts for the new values passed into the function but does not consider approval amounts made previously. It is unclear which is desirable here.

Recommendation: Clarifying the intention and fixing as necessary.
Update: the logic has been updated to count the approvals globally.

QSP-22 Potential Violation of the Spec

Severity: *Informational*

Status: Acknowledged

File(s) affected: `delegation/TokenState.sol`

Description: In `delegation/TokenState.sol`, if a locker such as the`Punisher` is removed via `removeLocker()`, parts of the spec will not be enforced, e.g., "Slashed funds will be held in an escrow account for the first year.".

Recommendation: Clarifying the intention and fixing as necessary.
Update: the team has provided an explanation that the owner is de-incentivized from harming the network. No fix is provided.

QSP-23 Inconsistent Behaviour of `addToLockedInPendingDelegations(...)`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `delegation/DelegationController.sol`

Description: In `delegation/DelegationController.sol`, `L571`: the behaviour of `addToLockedInPendingDelegations(...)` is inconsistent with the behaviour of `subtractFromLockedInPerdingDelegations(...)`: in the latter case, the `amount` is being subtracted from the existing amount, while in the former, it simply overwrites `_lockedInPendingDelegations[holder].amount` with the new ` amount.

Recommendation: Clarifying the intention of the `addToLockedInPendingDelegations(...)` method and fixing as necessary.
Update: the logic has been updated, and `addToLockedInPendingDelegations(...)` is now consistent with `subtractFromLockedInPerdingDelegations(...)`.

QSP-24 Error-prone Logic for `delegated.mul(2)`

Severity: *Informational*

Status: Fixed

File(s) affected: `delegation/TokenLaunchHolder.sol`

Description: In `delegation/TokenLaunchHolder`, both `L117`: `if (_totalDelegatedAmount[wallet].delegated.mul(2) >= _locked[wallet] &&` and `L163`: `if (_totalDelegatedAmount[holder].delegated.mul(2) < _locked[holder]) {` contain the "magic" value of `2`. This makes the logic error-prone, as the constant is scattered around while lacking a developer documentation.

Recommendation: Centralizing the constant, defining it in a single place. Improving the documentation accordingly.
Update: Fixed in `1da7bbd`.

QSP-25 Event Emitted Regardless of Result

Severity: *Informational*

Status: Fixed

File(s) affected: `delegation/TokenState.sol`

Description: In `delegation/TokenState.sol`, `L73`: the event is emitted regardless of whether the locked gets removed or not.

Recommendation: Confirming if this behaviour is intentional and fixing it as necessary.
Update: Fixed in `1da7bbd`.

QSP-26 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `delegation/TokenLaunchManager.sol, contracts/Permissions.sol`

Description: The following locations are missing input parameter validation:

1. `delegation/TokenLaunchManager.sol`, `L74`: should check that `seller` is a non-zero address.
2. `delegation/TokenLaunchManager.sol`, `L56`: should check that `walletAddress[i]` is a non-zero address while `value[i]` is above zero.
3. `contracts/Permissions.sol`, `L36`: `_contractManager` should be checked to be non-zero.

Recommendation: Adding the missed input validation.
Update: Fixed in `1da7bbd`.

Adherence to Specification

Generally, it is difficult to assess full adherence to the specification since the specification is incomplete, and the code appears to be poorly documented. This remains to be an issue for the latest commit `remediation-3`. For the commit `50c8f4e`:

- `delegation/DelegationPeriodManager.sol`, `L43`: the requirement described in the comment `remove only if there is no guys that stacked tokens for this period` is actually not enforced in the code. **Fixed** in `bd17697`.
- `delegation/TimeHelpers.sol`: `L42`: We believe the `+ 1` is used because "Delegation requests are always for the next epoch (month)." as per the Spec document, but this should be clarified in the code directly. **Fixed** (code removed).
- `delegation/TimeHelpers.sol`: `calculateDelegationEndTime()`- the rationale for the logic in this function is not documented, and therefore, is difficult to assess. **Fixed** (code refactored).

Code Documentation

The code appears to be poorly documented. The function descriptions from the Spec document should be directly inlined in the code. The coupling of the contracts with various `allow()` statements makes the architecture difficult to follow. In addition, there are grammatical errors in documentation, and we suggest spell-checking all the comments in the project. This remains to be an issue for the latest commit `remediation-3` as well, while some of the issues highlighted for commit `50c8f4e` were fixed.

For the commit `50c8f4e`:

1. `delegation/TokenSaleManager.sol`, `L47`: The function `approve()` increases the allowance of each wallet address (`+=` on `L51`). This should be documented, since the typical `ERC20.approve()` function sets the approval to the new value. The semantics of this function more closely relates to `increaseApproval()`. **Fixed**: in `1da7bbd`.
2. `interfaces/delegation/IHolderDelegation.sol`, `L26`: a typo: "begining". **Fixed** (removed).
3. `interfaces/delegation/IHolderDelegation.sol`,`L34`: "it's" -> 'its". **Fixed** (removed).

4. `ContractManager.sol`: `L27`: "for this moment in skale manager system by human name." Perhaps, changing this to "This contract contains the current mapping from contract IDs (in the form of human-readable strings) to addresses."` . **Fixed** (removed).
5. `ContractManager.sol`, `L43` (similar on `L44`): "is not equal zero" -> "is not equal to zero". **Fixed** (removed).
6. `ContractManager.sol`, `L54`: "is not contain code" -> "does not contain code". **Fixed** (removed).
7. `delegation/ValidatorService.sol`, `L68`: typo in "epmtyArray". **Fixed** (removed).
8. `SkaleToken.sol`: `L58`: "specify" -> "the specified". **Fixed**.
9. `delegation/DelegationService.sol`, `L251`: typo: "founds" -> "found". **Fixed**.

For the commit `remediation-3`:

1. `delegation/DelegationController.sol`, `L582`: `subtractFromLockedInPerdingDelegations(...)`: a potential typo (`subtractFromLockedInPendingDelegations` seems to be the intended name). **Fixed**.
2. `ConstantsHolder.sol`, `L146`: `iverloaded`- a typo. **Fixed**.
3. `DelegationPeriodManager.sol`, `L32`: The event field `legth` is misspelled. **Fixed**.

Adherence to Best Practices

For the commit `50c8f4e`:

1. `ContractManager.sol`, `L49-54`: this could be replaced by the Open Zeppelin Address library which contains more robust checks for contracts. **Not fixed**.
2. Favour using `uint256` instead of `uint`. **Not fixed**.
3. Cyclic imports in `delegation/*`. Generally, the architecture is a bit hard to follow. **Not fixed**.
4. The contracts use almost no events, which could make contract monitoring more difficult than necessary. **Fixed** (now events are used).
5. `interfaces/delegation/IValidatorDelegation.sol`, `L53-54`: commented out code and a TODO. **Fixed** (file removed).
6. `delegation/ValidatorService.sol`, `L136` (and similarly, `L127`): `return getValidatorId(validatorAddress) == validatorId ? true : false` could simply be `return getValidatorId(validatorAddress) == validatorId`. **Not fixed**.
7. `Permissions.sol`, `L68`: The constructor should check that `newContractsAddress` is non-zero. **Fixed** in `c671839`.
8. `delegation/DelegationPeriodManager.sol`, `L35`: can simplify to `return stakeMultipliers[monthsCount] != 0;`. **Not fixed**.
9. `delegation/TokenState.sol`, `L132`: a leftover TODO. **Fixed**.
10. `SkaleToken.sol`, `L75`: a modifier is commented out, need to confirm if it is intentional or not. **Addressed** (confirmed it is a comment).
11. `delegation/DelegationController.sol`, `L78`: `getState` is not the most intuitive name, because it actually modifies the contract's state. Suggesting naming it `refreshState`. The same applies to the methods `getActiveDelegationsByValidator`, `getDelegationsTotal`, `getDelegationsByHolder`, `getDelegationsForValidator` and other files, such as `getPurchasedAmount` on `TokenState.sol`, `L159`. **Fixed**.
12. `delegation/DelegationService.sol`, `L180`: `linkNodeAddress()` should check that `nodeAddress` and `validatorAddress` are non-zero. **Fixed**.
13. `interfaces/delegation/IValidatorDelegation.sol` and `delegation/DelegationController.sol`: `L21`, `pragma experimental ABIEncoderV2;` is unnecessary. **Fixed** (a false-positive).
14. `delegation/DelegationController.sol`, `L78`: usused variable `state`. **Fixed**.
15. `delegation/TokenState.sol`, `L246-256`: unless the order matters, instead of running the inner loop (`L249-251`), could move the last element into `_endingDelegations[delegation.holder][i]`. **Fixed**.
16. `delegation/ValidatorService.sol`, `L56`: `registerValidator()` should check that `validatorAddress` is non-zero. **Fixed**.

For the commit `remediation-3`:

1. `DelegationPeriodManager.sol`: the structs `PartialDifferencesValue` and `PartialDifferences` look very similar: the purpose for having both remains unclear. Also, it does not follow a naming practice of not using the word "Value"
2. `DelegationPeriodManager.sol`: readability and potential error-proneness of the `reduce(...)` method. The code of `delegation/DelegationConroller.sol` seems to be complex. The `reduce` function has five overloads, and two of them have overlap. The logic of calculating the differences is intertwined with the handling of corner cases of `firstUnprocessedMonth`. The overloads of `reduce` at `L780` and `L815` have significant overlaps, and it is unclear why both are needed.
3. `DelegationPeriodManager.sol`: `L250` and `L254`: `getAndUpdateLockedAmount(...)` and `getAndUpdateForbiddenForDelegationAmount(...)` seem to be identical - any reason for having two different methods?
4. `DelegationPeriodManager.sol`, `L289-313`: `currentMonth.add(1)` could be replaced with `delegations[delegationId].started` instead of repeatedly adding `1` to each term
5. `DelegationPeriodManager.sol`, `L55-56`: naming does not differentiate the units, which could lead to bugs:

```
uint created; // time of creation - measured as a timestamp
uint started; // month when a delegation becomes active - measured in months
```

6. `DelegationPeriodManager.sol`, `L642`: `i` changes its meaning after a re-assignment, this is not recommended.

```
      for (uint i = startMonth; i > 0 && i < delegations[delegationId].finished; i =
_slashesOfValidator[validatorId].slashes[i].nextMonth) {
```

1. `DelegationPeriodManager.sol`: fraction and GCD methods should be placed in a separate file.
2. `delegation/ValidatorService.sol`, `L99`: `1000` should be a shared constant
3. `delegation/ValidatorService.sol`, `L186` and `L194`: `? true : false;` is unnecessary
4. `delegation/DelegationPeriodManager.sol`, `L38`: `return stakeMultipliers[monthsCount] != 0 ? true : false;` could be `return (stakeMultipliers[monthsCount] != 0)`
5. `delegation/Distributor.sol`, `L83`: the constant `3` should be defined as a named constant at the contract level
6. `delegation/Distributor.sol`, `L108`: `require(now >= timeHelpers.addMonths(constantsHolder.launchTimestamp(), 3), "Bounty is locked");` the message should say "Fee is locked"
7. `delegation/TokenLaunchLocker.sol`, `L46`: duplicate definitions of `PartialDifferencesValue` and `getAndUpdateValue(...)`: already defined in `DelegationController.sol`.
8. `utils/MathUtils.sol`, `L40`: `boundedSubWithoutEvent(...)` is unused (outside tests).
9. `delegation/Punisher.sol`, `L68`: the amount emitted in `Forgive()` should possibly be `_locked[holder]` in the case that `amount > _locked[holder]`.
10. `delegaion/ValidatorService.sol`: the function `deleteNode()` should have the require check that `position < validatorNodes.length` similar to in `checkPossibilityToMaintainNode()`. Otherwise, this function could potentially shorten the list of validators without actually finding the correct one.
11. `delegation/DelegationController.sol`: the internal `init()` function on `L628` does not appear to be used.
12. `delegation/DelegationController.sol`: The else-branch on `L698` of `add()` can never be reached due to `L686` and `L688`. It is not clear what the intended semantics here. Likely, `L686` needs a `.add(1)` similar to `L703`.
13. `delegation/DelegationController.sol`: The if-conditional on `L725` `sequence.firstUnprocessedMonth <= month` can never fail due to the require-condition on `L720` `month.add(1) >= sequence.firstUnprocessedMonth`, and should be removed.
14. `delegation/DelegationController.sol`: Lots of duplicate code in the two `reduce()` functions on `L780` and `L815`.
15. `delegation/DelegationController.sol`: The require on `L586`: `_lockedInPendingDelegations[holder].amount >= amount` is not necessary due to the use of `.sub()` directly below.

Test Results

Test Suite Results

For the commit `remediation-3`, within the scope of the audit, two tests have failed on our side.
We re-ran the tests for the commit `9d60180`, and they all passed.

```
  Contract: ContractManager
     ✓ Should deploy
     ✓ Should add a right contract address (ConstantsHolder) to the register (100ms)

  Contract: Delegation
    when holders have tokens and validator is registered
       ✓ should check 1 month delegation period availability
       ✓ should not allow to send delegation request for 1 month (421ms)
       ✓ should check 2 months delegation period availability (49ms)
       ✓ should not allow to send delegation request for 2 months (379ms)
       ✓ should check 3 months delegation period availability
       ✓ should check 4 months delegation period availability
       ✓ should not allow to send delegation request for 4 months (299ms)
       ✓ should check 5 months delegation period availability
       ✓ should not allow to send delegation request for 5 months (324ms)
       ✓ should check 6 months delegation period availability
       ✓ should check 7 months delegation period availability
       ✓ should not allow to send delegation request for 7 months (333ms)
       ✓ should check 8 months delegation period availability
       ✓ should not allow to send delegation request for 8 months (327ms)
       ✓ should check 9 months delegation period availability
       ✓ should not allow to send delegation request for 9 months (331ms)
       ✓ should check 10 months delegation period availability
       ✓ should not allow to send delegation request for 10 months (351ms)
       ✓ should check 11 months delegation period availability
       ✓ should not allow to send delegation request for 11 months (336ms)
       ✓ should check 12 months delegation period availability
       ✓ should check 13 months delegation period availability
       ✓ should not allow to send delegation request for 13 months (370ms)
       ✓ should check 14 months delegation period availability
       ✓ should not allow to send delegation request for 14 months (300ms)
       ✓ should check 15 months delegation period availability
       ✓ should not allow to send delegation request for 15 months (335ms)
       ✓ should check 16 months delegation period availability
       ✓ should not allow to send delegation request for 16 months (573ms)
       ✓ should check 17 months delegation period availability
       ✓ should not allow to send delegation request for 17 months (345ms)
       ✓ should check 18 months delegation period availability
       ✓ should not allow to send delegation request for 18 months (349ms)
       ✓ should not allow holder to delegate to unregistered validator (320ms)
       ✓ should calculate bond amount if validator delegated to itself (2725ms)
       ✓ should calculate bond amount if validator delegated to itself using different periods (2601ms)
       ✓ should bond equals zero for validator if she delegated to another validator (3819ms)
       ✓ should be possible for N.O.D.E. foundation to spin up node immediately (461ms)
Reduce holders amount to fit Travis timelimit
       ✓ should be possible to distribute bounty accross thousands of holders (23927ms)
      when delegation period is 3 months
         ✓ should send request for delegation (575ms)
        when delegation request is sent
           ✓ should not allow to burn locked tokens (796ms)
           ✓ should not allow holder to spend tokens (2199ms)
           ✓ should allow holder to receive tokens (448ms)
           ✓ should accept delegation request (455ms)
           ✓ should unlock token if validator does not accept delegation request (1152ms)
          when delegation request is accepted
             ✓ should extend delegation period if undelegation request was not sent (6710ms)
      when delegation period is 6 months
         ✓ should send request for delegation (740ms)
        when delegation request is sent
           ✓ should not allow to burn locked tokens (789ms)
           ✓ should not allow holder to spend tokens (2758ms)
           ✓ should allow holder to receive tokens (396ms)
           ✓ should accept delegation request (528ms)
           ✓ should unlock token if validator does not accept delegation request (1382ms)
          when delegation request is accepted
             ✓ should extend delegation period if undelegation request was not sent (7343ms)
      when delegation period is 12 months
         ✓ should send request for delegation (671ms)
        when delegation request is sent
           ✓ should not allow to burn locked tokens (685ms)
           ✓ should not allow holder to spend tokens (2252ms)
           ✓ should allow holder to receive tokens (616ms)
           ✓ should accept delegation request (394ms)
           ✓ should unlock token if validator does not accept delegation request (1397ms)
          when delegation request is accepted
             ✓ should extend delegation period if undelegation request was not sent (6210ms)
      when 3 holders delegated
         ✓ should distribute funds sent to Distributor across delegators (5112ms)
        Slashing
           ✓ should slash validator and lock delegators fund in proportion of delegation share (4354ms)
           ✓ should not lock more tokens than were delegated (1838ms)
           ✓ should allow to return slashed tokens back (1415ms)
           ✓ should not pay bounty for slashed tokens (2847ms)
           ✓ should reduce delegated amount immediately after slashing (625ms)
           ✓ should not consume extra gas for slashing calculation if holder has never delegated (3706ms)

  Contract: DelegationController
    when arguments for delegation initialized
       ✓ should reject delegation if validator with such id does not exist (281ms)
       ✓ should reject delegation if it doesn't meet minimum delegation amount (282ms)
       ✓ should reject delegation if request doesn't meet allowed delegation period (344ms)
       ✓ should reject delegation if holder doesn't have enough unlocked tokens for delegation (952ms)
       ✓ should send request for delegation (876ms)
       ✓ should reject delegation if it doesn't have enough tokens (2037ms)
       ✓ should reject canceling if delegation doesn't exist (63ms)
       ✓ should allow to delegate if whitelist of validators is no longer supports (1324ms)
      when delegation request was created
         ✓ should reject canceling request if it isn't actually holder of tokens (67ms)
         ✓ should reject canceling request if validator already accepted it (448ms)
         ✓ should reject canceling request if delegation request already rejected (248ms)
         ✓ should change state of tokens to CANCELED if delegation was cancelled (209ms)
         ✓ should reject accepting request if such validator doesn't exist (126ms)
         ✓ should reject accepting request if validator already canceled it (364ms)
         ✓ should reject accepting request if validator already accepted it (565ms)
         ✓ should reject accepting request if next month started (526ms)
         ✓ should reject accepting request if validator tried to accept request not assigned to him (295ms)
         ✓ should allow for QA team to test delegation pipeline immediately (2015ms)
        when delegation is accepted
           ✓ should allow validator to request undelegation (530ms)
           ✓ should not allow everyone to request undelegation (713ms)

  Contract: PartialDifferences
     ✓ should calculate sequences correctly (1345ms)

  Contract: TokenLaunchManager
     ✓ should register seller (177ms)
     ✓ should not register seller if sender is not owner (81ms)
    when seller is registered
       ✓ should not allow to approve transfer if sender is not seller (59ms)
       ✓ should fail if parameter arrays are with different lengths (67ms)
       ✓ should not allow to approve transfers with more then total money amount in sum (156ms)
       ✓ should not allow to retrieve funds if it was not approved (114ms)
       ✓ should not allow to retrieve funds if launch is not completed (159ms)
       ✓ should allow seller to approve transfer to buyer (1735ms)
       ✓ should allow seller to change address of approval (851ms)
       ✓ should allow seller to change value of approval (702ms)
      when holder bought tokens
         ✓ should lock tokens (748ms)
         ✓ should not unlock purchased tokens if delegation request was cancelled (1506ms)
         ✓ should be able to delegate part of tokens (4713ms)
      ✓ should unlock all tokens if 50% was delegated for 90 days (3020ms)
      ✓ should unlock no tokens if 40% was delegated (2451ms)
      ✓ should unlock all tokens if 40% was delegated and then 10% was delegated (5434ms)
      ✓ should unlock tokens after 3 month after 50% tokens were used (3733ms)
      ✓ should unlock tokens if 50% was delegated and then slashed (2787ms)
      ✓ should not lock free tokens after delegation request cancelling (2929ms)

Contract: DelegationController
    ✓ should not lock tokens by default (205ms)
    ✓ should not allow to get state of non existing delegation
  when delegation request is sent
      ✓ should be in `proposed` state (98ms)
      ✓ should automatically unlock tokens after delegation request if validator don't accept (270ms)
      ✓ should allow holder to cancel delegation before acceptance (744ms)
      ✓ should not allow to accept request after end of the month (510ms)
    when delegation request is accepted
        ✓ should allow to move delegation from proposed to accepted state (261ms)
        ✓ should not allow to request undelegation while is not delegated (161ms)
        ✓ should not allow to cancel accepted request (168ms)
      when 1 month was passed
          ✓ should become delegated (256ms)
          ✓ should allow to send undelegation request (923ms)

Contract: ValidatorService
    ✓ should register new validator (155ms)
    ✓ should reject if validator tried to register with a fee rate higher than 100 percent (61ms)
  when validator registered
      ✓ should reject when validator tried to register new one with the same address (64ms)
      ✓ should reset name, description, minimum delegation amount (203ms)
      ✓ should link new node address for validator (138ms)
      ✓ should reject if linked node address tried to unlink validator address (150ms)
      ✓ should reject if validator tried to override node address of another validator (333ms)
      ✓ should not link validator like node address (220ms)
      ✓ should unlink node address for validator (460ms)
      ✓ should not allow changing the address to the address of an existing validator (168ms)
      ✓ should reject when someone tries to set new address for validator that doesn't exist (60ms)
      ✓ should reject if validator tries to set new address as null (61ms)
      ✓ should reject if provided validatorId equals zero (65ms)
      ✓ should return list of trusted validators (442ms)
    when validator requests for a new address
        ✓ should reject when hacker tries to change validator address (89ms)
        ✓ should set new address for validator (153ms)
    when holder has enough tokens
        ✓ should allow to enable validator in whitelist (112ms)
        ✓ should allow to disable validator from whitelist (311ms)
        ✓ should not allow to send delegation request if validator isn't authorized (284ms)
        ✓ should allow to send delegation request if validator is authorized (525ms)
        ✓ should be possible for the validator to enable and disable new delegation requests (1642ms)

Contract: SkaleManager
    ✓ should fail to process token fallback if sent not from SkaleToken (80ms)
    ✓ should transfer ownership (199ms)
  when validator has delegated SKALE tokens
      ✓ should create a node (588ms)
      ✓ should not allow to create node if validator became untrusted (1047ms)
    when node is created
        ✓ should fail to init exiting of someone else's node (180ms)
        ✓ should initiate exiting (370ms)
        ✓ should remove the node (457ms)
        ✓ should remove the node by root (410ms)
    when two nodes are created
        ✓ should fail to initiate exiting of first node from another account (168ms)
        ✓ should fail to initiate exiting of second node from another account (168ms)
        ✓ should initiate exiting of first node (382ms)
        ✓ should initiate exiting of second node (389ms)
        ✓ should remove the first node (416ms)
        ✓ should remove the second node (436ms)
        ✓ should remove the first node by root (410ms)
        ✓ should remove the second node by root (756ms)
        ✓ should check several monitoring periods (5157ms)
    when 18 nodes are in the system
        ✓ should fail to create schain if validator doesn't meet MSR (449ms)
        ✓ should fail to send monitor verdict from not node owner (146ms)
        ✓ should fail to send monitor verdict if send it too early (172ms)
        ✓ should fail to send monitor verdict if sender node does not exist (148ms)
        ✓ should send monitor verdict (258ms)
        ✓ should send monitor verdicts (433ms)
      when monitor verdict is received
          ✓ should store verdict block
          ✓ should fail to get bounty if sender is not owner of the node (120ms)
          ✓ should estimate bounty (277ms)
          ✓ should get bounty (3190ms)
      when monitor verdict with downtime is received
          ✓ should store verdict block
          ✓ should fail to get bounty if sender is not owner of the node (271ms)
          ✓ should get bounty (4312ms)
          ✓ should get bounty after break (4211ms)
          ✓ should get bounty after big break (4157ms)
      when monitor verdict with latency is received
          ✓ should store verdict block
          ✓ should fail to get bounty if sender is not owner of the node (115ms)
          ✓ should get bounty (3389ms)
          ✓ should get bounty after break (3186ms)
          ✓ should get bounty after big break (6498ms)
      when developer has SKALE tokens
          ✓ should create schain (3345ms)
        when schain is created
            ✓ should fail to delete schain if sender is not owner of it (213ms)
            ✓ should delete schain (2039ms)
            ✓ should delete schain after deleting node (4645ms)
          when another schain is created
             ✓ should fail to delete schain if sender is not owner of it (180ms)
             ✓ should delete schain by root (2712ms)
      when 32 nodes are in the system
        when developer has SKALE tokens
           ✓ should create 2 medium schains (6351ms)
          when schains are created
             ✓ should delete first schain (2249ms)
             ✓ should delete second schain (2254ms)
      when 16 nodes are in the system
         ✓ should create 16 nodes & create & delete all types of schain (38429ms)

  Contract: SkaleToken
     ✓ should have the correct name
     ✓ should have the correct symbol
     ✓ should have the correct decimal level
     ✓ should return the capitalization of tokens for the Contract
     ✓ owner should be equal owner (41ms)
     ✓ should check 10 SKALE tokens to mint (38ms)
     ✓ the owner should have all the tokens when the Contract is created
     ✓ should return the total supply of tokens for the Contract
     ✓ any account should have the tokens transferred to it (424ms)
     ✓ should not let someone transfer tokens they do not have (1006ms)
     ✓ an address that has no tokens should return a balance of zero
     ✓ an owner address should have more than 0 tokens
     ✓ should emit a Transfer Event (335ms)
     ✓ allowance should return the amount I allow them to transfer (84ms)
     ✓ allowance should return the amount another allows a third account to transfer (81ms)
     ✓ allowance should return zero if none have been approved for the account
     ✓ should emit an Approval event when the approve method is successfully called (57ms)
     ✓ holder balance should be bigger than 0 eth
     ✓ transferFrom should transfer tokens when triggered by an approved third party (404ms)
     ✓ the account funds are being transferred from should have sufficient funds (1000ms)
     ✓ should throw exception when attempting to transferFrom unauthorized account (685ms)
     ✓ an authorized accounts allowance should go down when transferFrom is called (428ms)
     ✓ should emit a Transfer event when transferFrom is called (394ms)
     ✓ should emit a Minted Event (420ms)
     ✓ should emit a Burned Event (320ms)
     ✓ should not allow reentrancy on transfers (1790ms)
     ✓ should not allow to delegate burned tokens (2591ms)
     ✓ should parse call data correctly (423ms)

  Contract: MathUtils
     ✓ should properly compare (42ms)
     ✓ should properly approximately check equality (80ms)
    in transaction
       ✓ should subtract normally if reduced is greater than subtracted
3
string
       ✓ should return 0 if reduced is less than subtracted and emit event (74ms)
    in call
       ✓ should subtract normally if reduced is greater than subtracted
       ✓ should return 0 if reduced is less than subtracted
```

Code Coverage

For the commit 50c8f4e:

The contracts that are in-scope are generally well-covered with tests: for most files, test coverage exceeds 90%. However, some lines could use additional coverage, such as:

- delegation/DelegationController, L119-120: testing delegation completion
- delegation/TokenState.sol, L121: testing an edge case for slashing forgiveness
- delegation/SkaleBalances, L74-75: testing bounty locking

The third-party library BokkyPooBahsDateTimeLibrary.sol has a coverage of only 23.77%, however, it appears to have its own test suite in the upstream repository.

For the commit remediation-3, we recommend improving branch coverage for the files that are lacking coverage. For TokenState.sol specifically, test coverage seems to be low at the commit we are auditing.

Update: We re-ran the tests for the commit 9d60180, and some files still have low coverage, including TokenState.sol. We recommend improving code coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	95.74	74.59	95.69	95.84	
ConstantsHolder.sol	84	25	72.73	84	192,196,200,201
ContractManager.sol	100	62.5	100	100	
Decryption.sol	100	100	100	100	
ECDH.sol	**95.77**	**75**	**100**	**95.77**	**160,187,229**
Monitors.sol	97.44	90.63	95	97.62	127,218,276

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Nodes.sol	98.56	78.26	97.37	98.04	356,509,538
Permissions.sol	88.89	60	83.33	84.62	76,82
Pricing.sol	97.67	85.71	100	97.67	74
Schains.sol	94.81	72	94.74	94.74	… 229,230,231
SchainsInternal.sol	93.85	85.71	94.12	94.34	… 549,648,734
SkaleDKG.sol	95.3	58.33	100	96.15	… 176,478,489
SkaleManager.sol	96.75	76	100	96.75	257,292,314,366
SkaleToken.sol	100	66.67	100	100	
SkaleVerifier.sol	94.74	78.57	100	94.74	61
SlashingTable.sol	100	100	100	100	
contracts/delegation/	93.3	75.31	95.95	93.06	
DelegationController.sol	95.59	84.88	96	95.57	… 736,748,786
DelegationPeriodManager.sol	71.43	100	66.67	71.43	55,57
Distributor.sol	94.29	69.23	100	94.29	179,186,208,214
PartialDifferences.sol	89.11	76.09	100	90	… 295,296,299
Punisher.sol	100	50	100	94.44	94
TimeHelpers.sol	100	50	100	100	
TokenLaunchLocker.sol	97.78	77.27	100	97.96	89
TokenLaunchManager.sol	100	68.75	100	100	
TokenState.sol	64	0	80	59.26	… 104,105,106
ValidatorService.sol	94.32	78.26	93.75	94.68	… 394,420,437
contracts/utils/	98.06	76.32	97.22	98.11	
FieldOperations.sol	96.92	83.33	100	96.92	171,250
FractionUtils.sol	94.44	50	83.33	94.44	43
MathUtils.sol	100	87.5	100	100	
Precompiled.sol	100	50	100	100	
StringUtils.sol	100	100	100	100	
All files	**95.12**	**75**	**95.91**	**95.1**	

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

```
3b5f554e5cb3ba6e111b3bf89c21fe4f20640da5a10e87bcb4670eccc3329d62 ./contracts/ContractManager.sol

e2a5194a012c8dbda09796f9d636a3b0ca9f926f9368ce3d4a5d41f9b05ed908 ./contracts/Permissions.sol

6c233479d7dca03fd9524e1fd5991e60cd4e15b049b61e4ad5200039076e4270 ./contracts/SkaleToken.sol

a6c52a9b24669db1ca15ac63a650c573020bfaa78e67dc63dd9a37e2c0f45188 ./contracts/SkaleManager.sol

2abb171a3bdf413e1a6bd8bfbc920251c162a9240a665bc4eec4415ac27e6f01
./contracts/interfaces/delegation/IDelegatableToken.sol

daf515794090cf18eda69ff4c2f42ae17c28cf9d64e89343e196683167aa57ab ./contracts/interfaces/delegation/ILocker.sol

df46ac05894d217c532804c2a37ae95e2bfab384ff97a0a86e5fdf05da49d9c3
./contracts/thirdparty/BokkyPooBahsDateTimeLibrary.sol

4985fcdd612c04439424e00942b627d604779640fa4216279779547e9624e01a ./contracts/thirdparty/openzeppelin/ERC777.sol

0ab43141c1be6c6764ed9c15c9b87a803250ff4ad57b56bd0be002b9358260d8 ./contracts/delegation/Punisher.sol
```

```
c13f4a58aac93d20b667f62b880627f4681ea5aaa2ec1a86c8f93cee55f342b5
./contracts/delegation/DelegationPeriodManager.sol

921d6266a1bfbe97259e34c90d854841ee87d6dce183edc9bb1fd577c9089370 ./contracts/delegation/TimeHelpers.sol

eab81a275aa9464e4e81d10e99817e867f586c3c81000d8b2472a8497e3527db ./contracts/delegation/TokenLaunchLocker.sol

7a63cff448c2d7b2f4862548444ba6bf8bb137fb22aec68487104dd483ae87a5 ./contracts/delegation/PartialDifferences.sol

1fdeaae183e4c3685cea9b860a5fc1720d7e8a309c9108f70e3e0286ce2f37b0 ./contracts/delegation/TokenState.sol

b8968762050d788c908042a217a4942dcc3a49de961b5f046a26e05acdfe6885 ./contracts/delegation/Distributor.sol

e0ee452239ff287f3719eae71c464422f0963696467b656f1fda3f512ab2c5ed ./contracts/delegation/DelegationController.sol

1459d07fee70ad118790f5797690ea7826170bc2be2d6594b22d8ab0fad288cc ./contracts/delegation/TokenLaunchManager.sol

b2242ae9b1ff600ec27ffe19d4a9ca2f97a72d13573f00ac8e494b3b018203c2 ./contracts/delegation/ValidatorService.sol

9ccbd82dab4f785b8ab3be31daad3bffa7b2a9043a3a78d18c5a65cdc756d115 ./contracts/utils/MathUtils.sol

c5eb14b8c58067d273ca65714ab37dd7245e8dbbcc78f698f741dc265d0b8ecf ./contracts/utils/StringUtils.sol
```

Tests

```
7445aec498345cbcfd2f7e23d544d6162dcd55c83f67094b84198d1c07a63d39 ./test/ECDH.ts

1c6e373bbf0df3ea96edbebb4f37d71668c752055127d5f806be391d1991dbba ./test/SkaleToken.ts

8fe27026eaa30c437fac34bf3fa16e6d148b7c3468af5c91d88ccf0e108ca0b2 ./test/Schains.ts

269ce78a5ede9b3f142b1bf2c6ea4c9e930ad46e0bc5874e96f94b30ce836c67 ./test/Monitors.ts

774772f446f0a778e652799612a5d25864f39cebc956d0010ad3006146dba14d ./test/SkaleDKG.ts

85de8a2a0f8c62ea54b599d11a72e88fe4e1911fd116f0523e861b59cef06fd2 ./test/Pricing.ts

38758b00b5a8f7218103e12f5b680d10392722110e2272c6443b71b386348c56 ./test/SkaleVerifier.ts

0a019c657a6e51b17f143906802139a99d99c4649a38dcdda9632023f2aaf822 ./test/ContractManager.ts

357e8e0d92cd9de52c3a2894688e95d3978136977a6e525d733117bd5ae922c4 ./test/ConstantsHolder.ts

31ccd2f2d91adeeb739b7ebb3c4ae81b785f1c1f0c0c4f31e108f9d94435b1a3 ./test/NodesFunctionality.ts

f4c07816c16a02470b487bd3e0f9deca7cc766ea8b539bd9dd1b10c7a274cca4 ./test/Decryption.ts

8dc44b1e7fa1796ee22036e1d5d82560714d7f24727a0f352db5f4a864b1d434 ./test/SkaleManager.ts

68fc99377576c7aea33cef124a60dfdc0e0cbe68280c60a69e9b07e753d31992 ./test/SchainsInternal.ts

a2151cd98c0ca5c72ef12eda9331ccd28d7ee802d5e34dc22ef3d75b8d590cf2 ./test/NodesData.ts

0f90907534d86572f02d497f94c2d5d5b24d2d94b8c3df127d3445c89ea91fd4 ./test/delegation/Delegation.ts

0382677cdd6689a25c897f1449ba7cd7bfc8f546584275a9972053ba89ccd917 ./test/delegation/TokenState.ts

172239c144c2ad8c1656de650f74f9b678b70e5e7ec6341fc22e4486dd8e9a5c ./test/delegation/TokenLaunch.ts

ee262687c8073fdc4489e6706413566731e3be1948f000f3b2f3c3add7308c5c ./test/delegation/PartialDifferences.ts

337c1b7ec194402da5204b1fc12ed84493079410a7acf160a03254f3c771998c ./test/delegation/DelegationController.ts

a0392eb0a3b5c33d7a3c55f8dbeafbe7c0867542a4ed82d9eac19cc8816d28c7 ./test/delegation/ValidatorService.ts

a3aaaacce8e943257a9861d9bc8400a6531ced268203a820385116a7e0d247b2 ./test/utils/MathUtils.ts

a01b72aafb9900064e91dc53380d8aa8737f4baa72f54136d7d55553cceb9742 ./test/tools/types.ts

90c2bdf776dc095f4dd0c1f33f746bea37c81b3ad81d51571d8dc84d416ecd13 ./test/tools/elliptic-types.ts

166431169af5c5def41b885179f26f0b9a977091c62e2ef4c05547a0bd9ab4f0 ./test/tools/command_line.ts

d41442de652a1c152afbb664069e4dd836c59b974c2f2c0955832fdcc617f308 ./test/tools/time.ts

862b66e303fdcd588b5fcd9153812893d22f9fe1acabf323e0eaaa5cfd778a0d ./test/tools/deploy/ecdh.ts

36f24255c90a436abce5ab1b2a14d2e19dda0402b7292bc5b52089689f56b20c ./test/tools/deploy/skaleToken.ts

7d52ba16d9dfed2314f6d140aa99f9e76a674d35a1960dcaf268f57b4d777aa5 ./test/tools/deploy/schains.ts

d020d06aa6b43356332f078b22b9639d7a224642ac447618ce2f21e00252c15f ./test/tools/deploy/monitors.ts

1aa57879dc8cc4150e55f84c884c041ca387d82784fe49163e4ba00ac0c4c917 ./test/tools/deploy/nodes.ts

2eef616fd6dd94f7e71bbbfa49a280db9453d953867f7cd2d9ccbe5150e6ac42 ./test/tools/deploy/skaleDKG.ts

b737e22a2a4958a1d8d6c0348bccc77b373ba71b4c74aa7798bb433014c120ce ./test/tools/deploy/pricing.ts

63e4802674e494a990b3ed5fe2dfcd32a5333320126f9ecc2856ac278b2dbb5c ./test/tools/deploy/slashingTable.ts

a911fb09c3f47d109a5a668df47a8b7a6e99788966ab82dd1565bbf9515a59b5 ./test/tools/deploy/skaleVerifier.ts

d98550c2db28471ff0a861987f5a3b0616b19ae08cd4256ab88c28194ebd947d ./test/tools/deploy/contractManager.ts

de7143ec7676eb978f26e9c8bb55ac861a43d4b4089ab7701d71e42335943881 ./test/tools/deploy/factory.ts
```

```
928b1bd747e8cb8715d6547f445e186949c0a70bf9257711e1e7d0cd62126385 ./test/tools/deploy/constantsHolder.ts

af7bcb8cfef5f110a1ee4c6a14451f9caaf0d00308a9fd5cd30bcf7759bb7720 ./test/tools/deploy/skaleManager.ts

6441be737f66551ec611210b7eff5128f8ced40cc4e6ac2e8bc865f866889480 ./test/tools/deploy/schainsInternal.ts

182d8295b09d1184f8bae29aef2a86e7b98c164d506997ed789625867534c799 ./test/tools/deploy/dectyption.ts

c1e81470f0ae2a1a2ca82c145b5050d95be248ca7e5d8a7b0f40f08cc568df34 ./test/tools/deploy/delegation/punisher.ts

4224ab4d3e35bf3de906f9ac07ec553d719cc0810a01dcb750b18d8f59e8d2bb ./test/tools/deploy/delegation/tokenState.ts

40cebd0731796b38c584a881d4cef78532ca4a7bab456d79194de54cd6aae740
./test/tools/deploy/delegation/tokenLaunchLocker.ts

29f342e6ec0b662fe021acf4e57691dfeec1a78c8624134287b9ef70cbba9de9
./test/tools/deploy/delegation/tokenLaunchManager.ts

3b3657a436f5437bba6c9ff07e9a90507aceb7456466b851d33b4aa84c3377d3
./test/tools/deploy/delegation/delegationController.ts

14663bff4b527d8b08c81ea577dceff2a70ea173816590a9d3b72bd8b5bf9b9e ./test/tools/deploy/delegation/distributor.ts

447f41a342e6645a08a6b96bd4e88abfc5009973af2526f8d410bfc7b7fa7046
./test/tools/deploy/delegation/delegationPeriodManager.ts

ff7d5f2a19c6766e728f1fcbc037c412f532649be1f448487c51e5f2ffa38c1b ./test/tools/deploy/delegation/timeHelpers.ts

9a19073e52af0e230516006d0f1eecafe320defa4808d9f3484b11db0e584c36
./test/tools/deploy/delegation/validatorService.ts

8bee5eef07f2e98c9fb49880d682b7e350cd0ace0d1e1c56a56d220361616355
./test/tools/deploy/test/partialDifferencesTester.ts

169c8f3df4758ae9780cff15eef9ada9549d1a082aff0302a25dc467e1616e4a
./test/tools/deploy/test/timeHelpersWithDebug.ts

65a100c7361f1fbb6c537c68aa9319519b39acf58e870bca6148725747638644 ./test/tools/deploy/test/reentracyTester.ts
```

Changelog

- 2020-02-12 - Initial report
- 2020-02-20 - Severity level of QSP-2 lowered after discussing with the Skale team.
- 2020-06-19 - Diff audited (`50c8f4e..remediation-3`)
- 2020-06-25 - Diff audited (multiple commits)
- 2020-07-10 - Severity level of QSP-17 lowered

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 $1B 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.