Quantstamp Security Assessment Certificate

September 8th 2020 — Quantstamp Verified

ArcX Finance

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

Executive Summary

Type	Audit
Auditors	Poming Lee, Research Engineer Jan Gorzny, Blockchain Researcher Jake Goh Si Yuan, Research Engineer
Timeline	2020-08-17 through 2020-09-08
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None

Source Code	
Repository	Commit
contracts	9734a75
contracts	5e07c9d
contracts	d22c805

Total Issues	12 (6 Resolved)
High Risk Issues	2 (2 Resolved)
Medium Risk Issues	2 (1 Resolved)
Low Risk Issues	4 (2 Resolved)
Informational Risk Issues	4 (1 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

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

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

Summary of Findings

The code looks well-structured and concise, with decent in-code comments. During auditing, Quantstamp found eleven potential issues of various levels of severity: two high-severity, two medium-severity, three low-severity, zero undetermined-severity, as well as four informational-level findings. We made fifteen best practices recommendations. Quantstamp highly recommends addressing the findings before going live.
** 2020-09-03 update **: Arcx team fixed most of the findings with high-severity or medium-severity, whereas the others remain unresolved. In addition, two new contracts were added to the repository by the Arcx team in this audit. They are `KYF.sol` and `KYFV2.sol`. To summarize, one low-severity finding was found in this round of audit and six best practices recommendations were made.
** 2020-09-08 update **: Arcx team added more fixes based on suggestions in the Best Practice section. In addition, `StakingRewardsAccrualCapped.sol` was added to the repository and not included in this audit.

ID	Description	Severity	Status
QSP-1	Unprotected and Unauthenticated function `transferCollateral`	High	Fixed
QSP-2	Unchecked Transfer Result	High	Fixed
QSP-3	Race Conditions / Front-Running	Medium	Acknowledged
QSP-4	Potential Catastrophic Positions Overflow	Medium	Fixed
QSP-5	[False-positive] Functional Bug in Function `repay`: Cannot repay	Low	Fixed
QSP-6	[False-positive] Functional Bug in Function `liquidate`: Cannot repay	Low	Fixed
QSP-7	Missing Input Checks	Low	Unresolved
QSP-8	Potential Overflow/Underflow	Low	Unresolved
QSP-9	Unlocked Pragma	Informational	Unresolved
QSP-10	Wrong Event Emitted	Informational	Fixed
QSP-11	Privileged Roles	Informational	Unresolved
QSP-12	TODOs not Implemented	Informational	Unresolved

Quantstamp Audit Breakdown

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Mythril 0.22.8
- Slither v0.6.6

Steps taken to run the tools:

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

Findings

QSP-1 Unprotected and Unauthenticated function `transferCollateral`

Severity: *High Risk*

Status: Fixed

Description: The synthetic token, as implemented in `token/SyntheticToken.sol` is designed to hold the collateral of the system. This means that the token is potentially holding all of the value of the system. The current implementation uses the method `transferCollateral` to move the collateral held to users who close their positions or are getting liquidated. However, `transferCollateral` is set to external visibility and is also not protected by any validation or authentication. This means that any arbitrary address may call that method and transfer any amount of tokens or collateral held by the synthetic token contract address.

Recommendation: Consider adding authentication such that only the `v1/CoreV1.sol` contract is able to call that method.

QSP-2 Unchecked Transfer Result

Severity: *High Risk*

Status: Fixed

Description: On `L462`, `L594`, `L604`: `transferCollateral` should be in a `require` statement.

QSP-3 Race Conditions / Front-Running

Severity: *Medium Risk*

Status: Acknowledged

Related Issue(s): SWC-114

Description: A block is an ordered collection of transactions from all around the network. It's possible for the ordering of these transactions to manipulate the end result of a block. A miner attacker can take advantage of this by generating and moving transactions in a way that benefits themselves.
In `token/BaseERC20.sol`, the function `approve` is susceptible to front-running. The ERC20 function `approve` is commonly known to have this issue with its base implementation, see more in the SWC-114.
** 2020-09-03 update **: Arcx team added functions `increaseAllowance` and `decreaseAllowance`.

Exploit Scenario: An example of an exploit goes as follows:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance` and `decreaseAllowance`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if the previous value has changed from its expected value before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

QSP-4 Potential Catastrophic Positions Overflow

Severity: *Medium Risk*

Status: Fixed

Description: In `v1/StateV1.sol`, the function `savePosition` uses an unsafe addition in `L161` to increment itself after setting a new position. This could lead to an overflow issue leading to an existing position being overwritten.

Recommendation: Use `SafeMath` for all `uint256` operations.

QSP-5 [False-positive] Functional Bug in Function `repay`: Cannot repay

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\v1\CoreV1.sol`

Description: `L418`, when a borrower repays, in `L420` should use `false` instead, to reduce the debt.

QSP-6 [False-positive] Functional Bug in Function `liquidate`: Cannot repay

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\v1\CoreV1.sol`

Description: `L552`, when a borrower repays, in `L554` should use `false` instead, to reduce the debt.

QSP-7 Missing Input Checks

Severity: *Low Risk*

Status: Unresolved

Description: * For `contracts\staking\StakingRewards.sol`, please check that all addresses passed to the `constructor` are not `0x0`.

- For `contracts\TokenStakingAccrual.sol`, please check that all addresses passed to the `constructor` are not `0x0`.
- For `contracts\v1\CoreV1.sol`, please check that all addresses passed to the `constructor` are not `0x0`.
- For `contracts\v1\StateV1.sol`, please check that all addresses passed to the function `setOracle` are not `0x0`.

QSP-8 Potential Overflow/Underflow

Severity: *Low Risk*

Status: Unresolved

Description: 1. SafeMath is not used on `L55` and `L77` of `KYF.sol`.

1. SafeMath is not used on `L74` and `L96` of `KYFV2.sol`.

QSP-9 Unlocked Pragma

Severity: *Informational*

Status: Unresolved

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked." For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

Recommendation: Pragma should be fixed or at least capped as per the standard security recommendations. Unknown side effects of new compiler versions could prove costly. Currently all Solidity files are affected.

QSP-10 Wrong Event Emitted

Severity: *Informational*

Status: Fixed

Description: In `v1/StateV1.sol` function `setOracle`, the event `MarketParamsUpdated` is emitted upon successful execution. This is incorrect as it should be some `OracleUpdated`, which is unfortunately an event that is currently non-existent.

QSP-11 Privileged Roles

Severity: *Informational*

Status: Unresolved

Description: * Admin can use the functions `increaseShare` and `decreaseShare` in `contracts\staking\AddressAccrual.sol` to manage shares of anyone.

- Admin can use the functions `mint` and `burn` in `contracts\token\ArcxToken.sol` to control the ArcxToken balance for anyone.
- Admin can update the logics of `CoreV1.sol` by calling `contracts\ArcProxy.sol`.

Recommendation: Please note these points to the official website or public document to make end users of ArcX platform be aware of this.

QSP-12 TODOs not Implemented

Severity: *Informational*

Status: Unresolved

File(s) affected: `contracts\v1\CoreV1.sol`

Description: Several `SUGGESTION` comments which probably should be implemented.
** 2020-09-03 **: Arcx team addressed most of the `SUGGESTION`. Currently still one suggestion in the file on `L163`

Automated Analyses

Mythril

The analysis was completed successfully. No issues were detected.

Slither

Slither ran into an error as shown and did not complete the analysis.

ERROR:root:Error in . ERROR:root:Traceback (most recent call last): File "/usr/local/lib/python3.8/dist-packages/slither/**main**.py", line 604, in main_impl (slither_instances, results_detectors, results_printers, number_contracts) = process_all(filename, args, File "/usr/local/lib/python3.8/dist-packages/slither/**main**.py", line 67, in process_all (slither, current_results_detectors, current_results_printers, current_analyzed_count) = process_single(File "/usr/local/lib/python3.8/dist-packages/slither/**main**.py", line 57, in process_single return _process(slither, detector_classes, printer_classes) File "/usr/local/lib/python3.8/dist-packages/slither/**main**.py", line 89, in _process detector_results = slither.run_detectors() File "/usr/local/lib/python3.8/dist-packages/slither/slither.py", line 161, in run_detectors results = [d.detect() for d in self._detectors] File "/usr/local/lib/python3.8/dist-packages/slither/slither.py", line 161, in results = [d.detect() for d in self._detectors] File "/usr/local/lib/python3.8/dist-packages/slither/detectors/abstract_detector.py", line 109, in detect all_results = self._detect() File "/usr/local/lib/python3.8/dist-

packages/slither/detectors/reentrancy/reentrancy_eth.py", line 81, in _detect super()._detect() File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 280, in _detect self.detect_reentrancy(c) File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 266, in detect_reentrancy self._explore(function.entry_point, []) File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 257, in _explore self._explore(son, visited) File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 257, in _explore self._explore(son, visited) File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 257, in _explore self._explore(son, visited) [Previous line repeated 8 more times] File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 242, in _explore contains_call = fathers_context.analyze_node(node, self) File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 140, in analyze_node if detector.can_callback(ir): File "/usr/local/lib/python3.8/dist-packages/slither/detectors/reentrancy/reentrancy.py", line 190, in can_callback return isinstance(ir, Call) and ir.can_reenter() File "/usr/local/lib/python3.8/dist-packages/slither/slithir/operations/library_call.py", line 24, in can_reenter return self.function.can_reenter(callstack) AttributeError: 'NoneType' object has no attribute 'can_reenter'

Adherence to Best Practices

• For `L231` in `contracts\staking\StakingRewards.sol`, consider using `reward.sub(reward.mul(2).div(3))` to avoid any rounding precision issues that might cause this function to revert unexpectedly.

• For `L607` in `contracts\v1\CoreV1.sol`, consider using `collateralDelta.value - Decimal.mul(collateralDelta.value, userSplit)` to avoid any rounding precision issues that might cause this function to revert unexpectedly.

• Remove the function `combine` in `contracts\lib\SignedMath.sol` since this function will return an unexpected result when `sint1` is not positive.

• The use of functions `to128`, `to96`, and `to32` may be problematic. The overflows were not explicitly checked. We recommend updating those to make sure that the value they are converting fits in the appropriate bit length.

• Typos or misleading comments/error messages: `contracts\v1\CoreV1.sol`: The comment on `L279` is incomplete.

• Typos and misleading comments/error messages: `contracts\v1\CoreV1.sol`: L493: “collatera” is a typo.

• Typos or misleading comments/error messages: `token/BaseERC20.sol` on L106; cannot burn to zero -> Cannot burn from zero, you are always burning to zero.

• Typos or misleading comments/error messages: `v1/StateV1.sol` on `L89`; error message should be State: only admin can call instead of core can call

• Typos or misleading comments/error messages: on `L68` of `staking/AddressAccrual.sol`

• Typos or misleading comments/error messages: on `L81` of `staking/AddressAccrual.sol`

• In `lib/SignedMath.sol`, consider changing the comments `Returns a new signed integer equal to a signed integer plus an unsigned integer.` into `Returns a new signed integer equal to an unsigned integer plus a signed integer.`; and `Returns a new signed integer equal to a signed integer minus an unsigned integer.` into `Returns a new signed integer equal to an unsigned integer minus a signed integer.` because although the base arithmetic operations are commutable, the `Int` operations are not.

• In `lib/Math.sol, the function`getPartialRoundUp` is never used. Dead code adds unnecessary gas costs. Consider removing unused dead code.

• State Variable Default Visibility: it is a smart contract best practice to explicitly set the visibility of state variables. In these contracts, some are set whilst some are ignored for some reason. The following are where some of these inconsistency has been spotted: -`v1/StateV1.sol`: `core` and `admin` -`staking/TokenStakingAccrual`: `stakedBalance`

• Inconsistent naming: In `v1/StateV1.sol`, the functions `setAmount` and `updatePositionAmount` are similar in operations and yet they are named quite differently. It is recommended to have the functions be named consistently.

• Magic Numbers: Some magic numbers are used in some files, most commonly `1e18`. Although it is understood what that does, it might be better to place it in a state const variable with an appropriate name for clarity. -`staking/StakingRewards.sol`: `L121` -`staking/StakingRewards.sol`: `L139` -`staking/Accrual.sol`: `L69`

** 2020-09-03 update **:

• For `KYF.sol` and `KYFV2.sol`: gas concerns for `removeMultiple`, consider adding a starting index to the function.

• For `KYF.sol` and `KYFV2.sol`: trivial input validation missing for `setVerifier`.

• For `KYFV2.sol`: It may be desirable to have `setHardCap` require that the new hard cap is at least as large as the current `count`. For `SynthRegistry.sol`: trivial input validation missing for `addSynth`. For `SynthRegistry.sol`: gas concerns for removeSynth. Consider adding a limit to the number of Synths that can be added to the contract.

• In `BaseERC20.sol`, the function `_setupDecimals` is never called in the constructor. Given that it should not be called elsewhere, this function is never used and can be removed (if it is called elsewhere, the comment is inconsistent and should be updated).

Test Results

Test Suite Results

All the current tests pass within a reasonable time.

```
 PASS  test/contracts/v1/integration.test.ts (19.23 s)
 PASS  test/contracts/staking/stakingRewardsCapped.test.ts (63.988 s)
 PASS  test/contracts/v1/state.test.ts (14.164 s)
 PASS  test/contracts/token/syntheticToken.test.ts (14.148 s)
 PASS  test/contracts/staking/integration.test.ts (13.635 s)
 PASS  test/contracts/staking/tokenAccrual.test.ts (11.828 s)
 PASS  test/contracts/staking/stakingRewards.test.ts (14.973 s)
 PASS  test/contracts/v1/borrowPosition.test.ts (22.981 s)
 PASS  test/contracts/staking/addressAccrual.test.ts (10.953 s)
 PASS  test/contracts/v1/liquidatePosition.test.ts (15.989 s)
 PASS  test/contracts/v1/repayPosition.test.ts (20.624 s)
 PASS  test/contracts/v1/openPosition.test.ts (6.91 s)
 PASS  test/contracts/v1/arc.test.ts (7.478 s)

Test Suites: 13 passed, 13 total
Tests:       101 passed, 101 total
Snapshots:   0 total
Time:        237.035 s
Ran all test suites matching /test\/contracts/i.
Done in 238.44s.
```

Code Coverage

The coverage data below was generated by Quantstamp. The error message shows that it wasn't a complete run. Quantstamp strongly recommends measuring the code coverage of the implemented test suite and making sure that the coverage is 100% or close to it. Otherwise, part of the code functionality will not be tested and could include bugs/vulnerabilities.

Compiled 45 contracts successfully All contracts have already been compiled, skipping compilation. ----------|----------|----------|----------|----------|----------------|

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	0	100	
ArcProxy.sol	100	100	0	100	
contracts/global/	0	0	0	0	
KYF.sol	0	0	0	0	… 77,79,88,89
KYFV2.sol	0	0	0	0	… 108,117,118
SynthRegistry.sol	0	0	0	0	… 108,109,111
contracts/impl/	0	100	0	0	
ChainLinkOracle.sol	0	100	0	0	21,29
contracts/interfaces/	100	100	100	100	
IChainLinkAggregator.sol	100	100	100	100	
IKYFV2.sol	100	100	100	100	
IMintableToken.sol	100	100	100	100	
IOracle.sol	100	100	100	100	
IStakingRewards.sol	100	100	100	100	
ISyntheticToken.sol	100	100	100	100	
contracts/lib/	0	0	0	0	
Adminable.sol	0	0	0	0	25,29,40
Decimal.sol	0	100	0	0	… 66,77,88,99
Math.sol	0	0	0	0	… 71,75,86,97
Storage.sol	0	100	0	0	18,20,23,36
contracts/staking/	0	0	0	0	
Accrual.sol	0	0	0	0	… 118,121,124
AddressAccrual.sol	0	0	0	0	… 71,81,83,84
RewardsDistributionRecipient.sol	0	0	0	0	14,18,27
StakingRewards.sol	0	0	0	0	… 296,300,301
StakingRewardsAccrual.sol	0	100	0	0	38,46
StakingRewardsAccrualCapped.sol	0	0	0	0	… 154,161,166
TokenStakingAccrual.sol	0	100	0	0	… 76,77,79,84
contracts/test/	0	100	0	0	
MockOracle.sol	0	100	0	0	19,23
TestToken.sol	0	100	0	0	30,39
contracts/token/	0	0	0	0	
ArcxToken.sol	0	100	0	0	29,39
BaseERC20.sol	0	0	0	0	… 374,375,386
SyntheticToken.sol	0	0	0	0	… 142,152,164
contracts/v1/	0	0	0	0	
CoreV1.sol	0	0	0	0	… 616,625,630
StateV1.sol	0	0	0	0	… 455,461,468
StorageV1.sol	100	100	100	100	
TypesV1.sol	0	0	0	0	… 183,193,203
All files	**0**	**0**	**0**	**0**	

Istanbul reports written to ./coverage/ and ./coverage.json solidity-coverage cleaning up, shutting down ganache server An unexpected error occurred:

{ Error: Cannot find module '@test/helpers/simpleDescribe' at Function.Module._resolveFilename (internal/modules/cjs/loader.js:636:15) at Function.Module._load (internal/modules/cjs/loader.js:562:25) at Module.require (internal/modules/cjs/loader.js:692:17) at require (internal/modules/cjs/helpers.js:25:18) at Object. (/root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/test/contracts/staking/addressAccrual.test.ts:43:40) at Module._compile (internal/modules/cjs/loader.js:778:30) at Module.m._compile (/root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/node_modules/ts-node/src/index.ts:1043:23) at Module._extensions..js (internal/modules/cjs/loader.js:789:10) at Object.require.extensions.(anonymous function) as .ts at Module.load (internal/modules/cjs/loader.js:653:32) at tryModuleLoad (internal/modules/cjs/loader.js:593:12) at Function.Module._load (internal/modules/cjs/loader.js:585:3) at Module.require (internal/modules/cjs/loader.js:692:17) at require (internal/modules/cjs/helpers.js:25:18) at /root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/node_modules/mocha/lib/mocha.js:349:36 at Array.forEach () at Mocha.loadFiles (/root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/node_modules/mocha/lib/mocha.js:346:14) at Mocha.run (/root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/node_modules/mocha/lib/mocha.js:1006:10) at Promise (/root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/node_modules/@nomiclabs/buidler/src/builtin-tasks/test.ts:55:15) at new Promise () at SimpleTaskDefinition.config_env_1.internalTask.addOptionalVariadicPositionalParam.setAction [as action] (/root/workspace/arc/contracts-d22c805dadcea053eb43f7acd0c4f14de0ead62b/node_modules/@nomiclab s/buidler/src/builtin-tasks/test.ts:54:26) at process._tickCallback (internal/process/next_tick.js:68:7) code: 'MODULE_NOT_FOUND' } error Command failed with exit code 1. info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.

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

```
10eddfebb2e61d245359b1ff3862e5475b1089e39b839e9be3a44be7a5b015d6 ./contracts/ArcProxy.sol
37e751e133a2f449e8023a7f5c6cad459b582269aa67a4faddefae63efb79c75 ./contracts/v1/CoreV1.sol
5a411f8f9c0fb913b9ddbcf66beddecd3865d5983c0ad985643e262a3dad0b65 ./contracts/v1/StateV1.sol
82f7fe8abe6c7d5e56bbd2e13fddc701b4535abf016b6a5885eaee9f947311b9 ./contracts/v1/StorageV1.sol
afc6dc928a64e15f38e57b07517a320389a8d5f9ddf59c226173402e59027f7f ./contracts/v1/TypesV1.sol
0bb3613431bfe6d58571a0428daa049d07030984a2906f99490eff609a0f10a2 ./contracts/token/ArcxToken.sol
974ca2d3e619db5e2c43a6b688af8bc9762cfd40b657379b6d30201a25801667 ./contracts/token/BaseERC20.sol
0cd8dbde5240fb7dd7e0fe3b521212810df77e2056c58919c462ca84b679ef9b ./contracts/token/SyntheticToken.sol
2c8b4ef70251547f156145bd498b2a9c23f68017965f798ba3669d52c5777a71 ./contracts/test/MockOracle.sol
9918fcb6af32656b9af08dd4ec1dab1420cbd019d19bd302b678717f1ffeb9b2 ./contracts/test/TestToken.sol
ba945b5eea12233f9fd798b17c0653b1b4992fd50c64802c81af649133bdf37b ./contracts/staking/Accrual.sol
202567031535a56e976654d0592d5944b70220a7a169382b4529d227e3f85c3c ./contracts/staking/AddressAccrual.sol
dd8a6a1c0942e65bcc562ad7b39e3c0a1edf543e6dfbeb41e51d945a19e6bf30 ./contracts/staking/RewardsDistributionRecipient.sol
3e208f819128364e745fead63bfb79671e510b8371ac3a37a26b74d028b2a96a ./contracts/staking/StakingRewards.sol
65dcf36b73ae2196c63da8db5546c6e04bf677e328f4eb796474a39913bda2f6 ./contracts/staking/StakingRewardsAccrual.sol
8d82bd5c3212b14cd3dc12b77017a2692daf71f9ba4c735d5cefc79fddb8a729 ./contracts/staking/StakingRewardsAccrualCapped.sol
a85e700a8600d1c7c5235e34a7ce4255739b34b6f8ea9d6f3d749b756f657499 ./contracts/staking/TokenStakingAccrual.sol
2d4979381edbe06e992d3f8c2ed2d7f1d0a53125f20cf29c1ce94e8dcc16b023 ./contracts/lib/Adminable.sol
feeca31ab248b925aa92303de0495f1dbfcbd064574cc7976c51db955a1a5e9f ./contracts/lib/Decimal.sol
2e73329e1c4e46378008d029e65914bb170a9fdab4a62edb742aeae122554725 ./contracts/lib/Math.sol
6377d519e5ed7c5c542c998df299e1762209d0efaff63183f2d3d479533fc418 ./contracts/lib/Storage.sol
cfc24cc17ecb55395bf52c8548dbd657cf7aaad85c0806d8aa8e9ab1b036ea13 ./contracts/interfaces/IChainLinkAggregator.sol
f20ec5d1fc11a2d83d6cd7b719aadfdf85eef4a5da5a28159d584953c97ae8e7 ./contracts/interfaces/IKYFV2.sol
0ee09f97b06c8c3603e71dc4719bcd74b882a1e8edd3a5ce1ef031ae479e7d3b ./contracts/interfaces/IMintableToken.sol
5338727f954b83da0cc9ec70748ac93f19eb52ebd562e2ce96556fdc59c0e86b ./contracts/interfaces/IOracle.sol
8bc46b72f6220a6f8204084b0f6a4caec6d59be4adf56485e78c2231b9ce174c ./contracts/interfaces/IStakingRewards.sol
661c938fb3a0a2b84bfe20f2afd45a31333f47d777483e3da84044bade8950f5 ./contracts/interfaces/ISyntheticToken.sol
59c33cff5572ce065875cd9d347127f3456443c1a0fedced7cbea1b897b28196 ./contracts/impl/ChainLinkOracle.sol
ad3e98056da63328d787255fec19db0af01222a11ff25f308ef4c53daff64eb0 ./contracts/global/KYF.sol
3b06ce02a5b42c5aef8af5a1703732b343333761c4ec89ebb5fc09fcc7a6defb ./contracts/global/KYFV2.sol
a6a088b020514d24d3499fccb550f3c74d4b4ee55dff2fb80757bc17a132ecbd ./contracts/global/SynthRegistry.sol
```

Tests

```
a9753819f23dc5dc079d27b65c5e93a1b8f92abf196549374794de5010e5a6f5 ./test/helpers/EVM.ts
37129cbba6791f4cd2ca7a95a7c23eb608f1b255ec890d3c766c1cf92b7b768e ./test/helpers/arcDescribe.ts
046bbdefecc77cb96478f7ca24121ad62c339184a2bfb72928f80d5d90b508da ./test/helpers/initializeArc.ts
90b52198d6626afc08a4795260e219171318588250130bf1d7b614f479c0fd66 ./test/helpers/simpleDescribe.ts
ee52e58bd21f64217d21f4f3aff0ad9f11248d67e0f70375d2c8907cba08108f ./test/deployment/deployment.test.ts
c37715048f15edd093f3d922544316192334b60b7d880f9dbcb4e930ad9dfb99 ./test/contracts/v1/arc.test.ts
83ab8e45827276f23a3be0c570b7cffceaff4cf1a3a40affd8248825f15814c9 ./test/contracts/v1/borrowPosition.test.ts
151a7a89ced840c039dcf838b06b62dfb087a59fb20b65dc1f6c0907ae867520 ./test/contracts/v1/integration.test.ts
8f65054d32f23ab8234019b72a253c4a5973d66aef7aa7af6c090ab6769e5838 ./test/contracts/v1/liquidatePosition.test.ts
0dc46c1f640f89085ce4b11e7911257da66760cf47e65e2312e00b866414cbef ./test/contracts/v1/openPosition.test.ts
9c8a271a36aac7b443b143ea94926f1c728e10dabb46d3af17c81e4ed7de5687 ./test/contracts/v1/repayPosition.test.ts
8560bf30dc1e860fbf90f34afa656006d50dc8e8d1dc3f8133d8211eac06d0b5 ./test/contracts/v1/state.test.ts
11a6e135617c660fcb5ee8df77120bdbab87d98a183e6b376db0b86ff82d7519 ./test/contracts/token/syntheticToken.test.ts
c9b2bac07c3efa799d6045617a04b7e04a43714141271309e784611d89aa5c4e ./test/contracts/staking/addressAccrual.test.ts
f5c8c41410a2a8794aad253dbe689d90d62dd8d01c16dacf6eaac5f45c288bc0 ./test/contracts/staking/integration.test.ts
24e69a26c0e2b50a32d27d276fb73c900d8fb1c2f19a7268f7e23b7ee13fa265 ./test/contracts/staking/stakingRewards.test.ts
b92a22daf4d17b4b06f32a0ec816bef9beb46dfb1185a21e6d8d229bba2bc947 ./test/contracts/staking/stakingRewardsCapped.test.ts
e4556fe1565a7ade743a789fb1ff945cbae79ef62e858c5f7eeccb7ac54d89ce ./test/contracts/staking/tokenAccrual.test.ts
```

Changelog

- 2020-08-31 - Initial report
- 2020-09-03 - Reaudit report
- 2020-09-08 - Final report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.