Idle Governance

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

Executive Summary

Type: Yield Farming and Governance

Auditors: Poming Lee, Research Engineer
Kevin Feng, Blockchain Researcher
Ed Zulkoski, Senior Security Engineer

Timeline: 2020-10-12 through 2020-10-26

EVM: Muir Glacier

Languages: Solidity, Javascript


Specification: README.md
https://developers.idle.finance/

Source Code: Repository, Commit
idle-governance 01S88bb
idle-governance 029654

Goals:
- Do functions have proper access control logic?
- Are there centralized components of the system which users should be aware?
- Do the contracts adhere to best practices?

Total Issues: 8 (6 Resolved)
High Risk Issues: 0 (0 Resolved)
Medium Risk Issues: 1 (1 Resolved)
Low Risk Issues: 2 (1 Resolved)
Informational Risk Issues: 4 (3 Resolved)
Undetermined Risk Issues: 1 (1 Resolved)

- High Risk: The issue puts a large number of users’ sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client’s reputation or serious financial implications for client and users.

- Medium Risk: The issue puts a subset of users’ sensitive information at risk, would be detrimental for the client’s reputation if exploited, or is reasonably likely to lead to moderate financial impact.

- Low Risk: The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client’s business circumstances.

- Informational: The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.

- Undetermined: The impact of the issue is uncertain.

- Unresolved: Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.

- Acknowledged: The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

- Resolved: Adjusted program implementation, requirements or constraints to eliminate the risk.

- Mitigated: Implemented actions to minimize the impact or likelihood of the risk.
Summary of Findings

During auditing, we found eight potential issues of various levels of severity: one medium-severity issue, two low-severity issues, four informational-level findings, and one undermined finding. We also made five best practices recommendations.

Disclaimer: Please be aware that Quantstamp was requested and had audited these files: PriceOracle.sol, Idle.sol, and IdleController.sol; the whole system was not audited by us.

update-2020-10-26: All findings were either fixed or acknowledged.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Severity</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSP-1</td>
<td>Unintended Revert in Function claim Idle</td>
<td>Medium</td>
<td>Fixed</td>
</tr>
<tr>
<td>QSP-2</td>
<td>Gas Usage / for Loop Concerns</td>
<td>Low</td>
<td>Acknowledged</td>
</tr>
<tr>
<td>QSP-3</td>
<td>_moveDelegates() May Not Behave Correctly After Token Transfers</td>
<td>Low</td>
<td>Fixed</td>
</tr>
<tr>
<td>QSP-4</td>
<td>Missing Address Sanitization</td>
<td>Informational</td>
<td>Fixed</td>
</tr>
<tr>
<td>QSP-5</td>
<td>Privileged Roles</td>
<td>Informational</td>
<td>Acknowledged</td>
</tr>
<tr>
<td>QSP-6</td>
<td>delegateBySig() Should Validate the v and s Parameters</td>
<td>Informational</td>
<td>Fixed</td>
</tr>
<tr>
<td>QSP-7</td>
<td>Possible Truncation in Calculating APR Precision</td>
<td>Informational</td>
<td>Fixed</td>
</tr>
<tr>
<td>QSP-8</td>
<td>Integer Overflow / Underflow</td>
<td>Undetermined</td>
<td>Fixed</td>
</tr>
</tbody>
</table>

Quantstamp Audit Breakdown

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

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.

2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

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

Setup

Tool Setup:

- Truffle v5.1.33
- SolidityCoverage v0.7.11
- Mythril v0.22.10
- Slither v0.6.12
Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
3. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
4. Installed the Mythril tool from Pypi: `pip3 install mythril`
5. Ran the Mythril tool on each contract: `myth a path/to/contract`
6. Installed the Slither tool: `pip install slither-analyzer`
7. Ran Slither from the project's root directory: `slither .`

**Findings**

**QSP-1 Unintended Revert in Function claimIdle**

**Severity:** Medium Risk

**Status:** Acknowledged

**Description:** In `contracts/IdleController.sol`, the function `claimIdle` will loop through all the combinations of `holders` and `idleTokens` and pass into the function `distributeIdle`. However, in the function `distributeIdle` on L119 will revert all the transactions that pass in any `supplier` != `idleToken`.

**Recommendation:** Remove the inner for loop in the function `claimIdle`, or remove L119 directly. The way of fixing this issue should be done based on the functionality that the idle team actually seeks to achieve.

**QSP-2 Gas Usage / Loop Concerns**

**Severity:** Low Risk

**Status:** Acknowledged

**Description:** Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a for loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely. It is best to break such loops into individual functions as possible. In particular, for `contracts/IdleController.sol`, if many markets are added, `refreshIdleSpeedsInternal()` may run into issues.

**Update-2020-10-26:** Idle Team added a `_resetMarkets` method which should completely remove all markets in case of emergency. Gas analysis was also conducted successfully.

**Recommendation:** We recommend 1) performing gas analysis to ensure that each loop-function will not run into gas limitations, particularly for large inputs, and 2) adding a function that can reduce the number of markets in the allMarkets for emergency use.

**QSP-3 _moveDelegates() May Not Behave Correctly After Token Transfers**

**Severity:** Low Risk

**Status:** Fixed

**Description:** In `contracts/Idle.sol`, the function `_moveDelegates()` invokes `_moveDelegates()` with the delegator’s full balance instead of remaining undelegated balance. This can cause users to lose delegation ability if additional Idle tokens are acquired without minting (i.e., via transfers).

Consider the following scenarios:

1. Alice has 10 Idle, which is delegated to Bob.
2. Alice acquires 1 additional Idle from a transfer. In general, if Alice’s balance is ever more than the number of tokens minted toward her account (due to transfers), she will not be able to delegate. This can be mitigated by Alice by simply transferring the excess tokens out of her account, however this scenario may not be clear to end-users from a UX-perspective.

**Recommendation:** It is not clear if this functionality is as intended. If so, no changes are needed, but user documentation should exist describing the scenario above. If the scenario above is undesirable, `_moveDelegates()` should be invoked in `transfer()` as well. Note however that with this approach, votes can be more easily "bought" by acquiring Idle tokens on exchanges.

**QSP-4 Missing Address Sanitization**

**Severity:** Informational

**Status:** Fixed

**Description:** For `contracts/IdleController.sol`, the values inside the `priceOracle` input parameter is not checked to be different from `0x0` inside the `_setPriceOracle` function.

**Recommendation:** Add a `require` statement that checks that the value of the `priceOracle` is different from `0x0`.

**QSP-5 Privileged Roles**

**Severity:** Informational

**Status:** Acknowledged

**Description:** (a) For `contracts/PriceOracle.sol`, a potentially malicious owner (if the private key was leaked) can change the fee contract addresses/block length to give incorrect price oracles that can affect the token. While privileged roles for the Idle token are addressed in https://developers.idle.finance/advanced/admin-powers, it is not addressed for the price oracle. (b) For `contracts/IdleController.sol`, the admin can manipulate the rate of tokens in which each market receives by adding/removing idle markets to the list.

**Update-2020-10-26:** Idle Team stated that the owner of the `PriceOracle.sol` and the `IdleController.sol` contracts will be the TimeLock contract (i.e., the governance itself) directly on deploy.

**Recommendation:** These privileged operations and their potential consequences should be clearly communicated to (non-technical) end-users via publicly available documentation.
QSP-6 \texttt{delegateBySig()} Should Validate the \texttt{v} and \texttt{s} Parameters

Severity: Informational

Status: Fixed

Description: For \texttt{contracts/Idle.sol, delegateBySig()} should validate the \texttt{v} and \texttt{s} parameters as in \texttt{ECDSA.sol} (See: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/2bb6e6af48757c470092b32d7a08a64P7V/contracts/cryptography/ECDSA.sol#L46).

QSP-7 Possible Truncation in Calculating APR Precision

Severity: Informational

Status: Fixed

Description: For \texttt{contracts/PriceOracle.sol, in function getCompApr} the computed pair will always return a result that has the last two digits as 0, in \texttt{L58} due to division before multiplication \texttt{div(cTokenNAV).mul(100)}. Recommendation: If it is not intended for the last two digits to be 0, then it is recommended to perform \texttt{mul(100).div(cTokenNAV)} instead for a more precise calculation.

QSP-8 Integer Overflow / Underflow

Severity: Undetermined

Status: Fixed

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute. Integer overflow and underflow may cause many unexpected kinds of behavior and was the core reason for the batchOverflow attack. Here's an example with unsigned integers with a range of 0..255. function under_over_flow() public { uint8 num_players = 0; num_players = num_players - 1; // 0 - 1 now equals 255! if (num_players == 255) { emit LogUnderflow(); // underflow occurred } uint8 jackpot = 255; jackpot = jackpot + 1; // 255 + 1 now equals 0! if (jackpot == 0) { emit LogOverflow(); // overflow occurred } } In particular, for \texttt{contracts/PriceOracle.sol, there is a potential underflow on L78 \ldots 10**(18-tokenDecimals) \ldots} It is recommended to use SafeMath for this operation.

Automated Analyses

Mythril

Mythril reported no issues.

Slither

- Slither warns of several potential reentrancy issues, however as the associated external calls were to trusted contracts (either Idle contracts or underlying protocols), we classified these as false positives.
- Slither detects that there are "divided-before-multiply" operations in \texttt{L58} of \texttt{contracts/PriceOracle.sol} as mentioned in our Finding section. Re-ordering these operations may improve precision.
- Slither detects that \texttt{comptrollerImplementation} in \texttt{contracts/IdleControllerStorage.sol} is never initialized as mentioned in our Best Practice section.

Adherence to Specification

The code adheres to the specification provided, as well as the inline documentation.

Code Documentation

The code is generally well-documented. We suggest several improvements:

- [false-positive] For \texttt{contracts/PriceOracle.sol, in L64 (and possibly L67 also), the documentation says to scale it to 1e18 but instead multiples by 10e10}
- For \texttt{contracts/PriceOracle.sol, the constant 100 in L58 is not documented.}
- Consider adding code comments to the functions that currently are not commented, to increase the maintainability of the code.

Adherence to Best Practices

The code does not fully adhere to best practices. In particular:

- For \texttt{contracts/PriceOracle.sol, consider using constants for the key part of the assignment for L28-L29 and L32-L37 like what is done in L26 and L27, in order to make the code more readable.}
- [false-positive] L48 in \texttt{contracts/IdleController.sol, comptrollerImplementation} was never set.
- For \texttt{contracts/PriceOracle.sol, the ChainLink price feeds are hardcoded in the constructor, for better coding practices it is better to pass this in as an initial value during the deployment of the contract.}
- For \texttt{contracts/Idle.sol, should use uint256 instead of uint.}
- [false-positive] For \texttt{contracts/IdleController.sol, Comp has the public convenience function claimComp(address holder) which claims for all markets. Is there a reason this was omitted?}
Compiling your contracts...

> Everything is up to date, there is nothing to compile.

Contract: EarlyRewards
✓ set params on constructor (74ms)
✓ setRewards rewards if passed have different length (103ms)
✓ setRewards rewards if passed have different length (27ms)
✓ setRewards rewards if passed have different length (108ms)
✓ setRewards rewards if passed have different length (64ms)
✓ recipient can claim (104ms)
✓ claim reverts if msg.sender is not authorized (108ms)
✓ emergencyWithdrawal can only be called by owner (78ms)
✓ emergencyWithdrawal can only be called by owner (78ms)
✓ emergencyWithdrawal can only be called by owner (78ms)

Contract: IdleController
✓ refreshIdleSpeeds with IDLE equally splitted (150ms)
✓ refreshIdleSpeeds with different supply (256ms)
✓ refreshIdleSpeeds with different price (240ms)
✓ refreshIdleSpeeds with different price (239ms)
✓ refreshIdleSpeeds with different price (287ms)
✓ refreshIdleSpeeds with different price (287ms)
✓ _resetMarkets (269ms)

Contract: PriceOracle
✓ allows onlyOwner to setBlocksPerYear (62ms)
✓ allows onlyOwner to updateFeedETH (63ms)
✓ getPriceUSD when there no price feed (64ms)
✓ getPriceUSD when there is USD price feed (64ms)
✓ getPriceUSD when there is USD price feed but ETH price feed is present (72ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getPriceUSD when there is no price feed for token (69ms)
✓ getUnderlyingPrice (103ms)
✓ getCompApr (88ms)

29 passing (18s)
## Code Coverage

The branch, statement and function coverage of `contracts/Idle.sol` and `contracts/IdleController.sol` are low. This indicates that much of the functionality of the protocol is not executed during tests. We strongly recommend that the branch coverage be brought to 100% as it is crucial to execute all functionality in order to verify that no functional bugs exist in the code.

### update-2020-10-26

Idle Team states that they have added only tests for differences with respect to the audited version of the code of Compound. This is why for `Idle.sol` and `IdleController.sol` tests, given that they had the majority of the code copied from Compound, don't have that much coverage.

<table>
<thead>
<tr>
<th>File</th>
<th>% Stmts</th>
<th>% Branch</th>
<th>% Funcs</th>
<th>% Lines</th>
<th>Uncovered Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>contracts/</td>
<td>27.18</td>
<td>15.2</td>
<td>29.25</td>
<td>27.47</td>
<td></td>
</tr>
<tr>
<td>ERC20Permit.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>EarlyRewards.sol</td>
<td>100</td>
<td>83.33</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>EcosystemFund.sol</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GovernorAlpha.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Idle.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IdleController.sol</td>
<td>66.94</td>
<td>37.04</td>
<td>70</td>
<td>66.94</td>
<td></td>
</tr>
<tr>
<td>IdleControllerStorage.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>PriceOracle.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Reservoir.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Timelock.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Unitroller.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Vester.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VesterFactory.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>contracts/interfaces/</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>CERC20.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>ChainLinkOracle.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Comptroller.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IdleToken.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>contracts/lib/</td>
<td>18.75</td>
<td>12</td>
<td>26</td>
<td>18.75</td>
<td></td>
</tr>
<tr>
<td>CarefulMath.sol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Exponential.sol</td>
<td>22.83</td>
<td>15.79</td>
<td>28.89</td>
<td>22.83</td>
<td></td>
</tr>
<tr>
<td>contracts/mocks/</td>
<td>90.48</td>
<td>100</td>
<td>90</td>
<td>90.48</td>
<td></td>
</tr>
<tr>
<td>CERC20Mock.sol</td>
<td>71.43</td>
<td>100</td>
<td>60</td>
<td>71.43</td>
<td>18.21</td>
</tr>
<tr>
<td>ChainLinkOracleMock.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>ComptrollerMock.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IdleTokenMock.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>PriceOracleMock.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>TokenMock.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>All files</td>
<td>27.79</td>
<td>14.67</td>
<td>35.23</td>
<td>28.03</td>
<td></td>
</tr>
</tbody>
</table>
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 (but is not necessarily) be an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

22f1b43d1da1e7a435ab0079a7d9f20b58a548d92a4b5f5397ba1712e07 ./contracts/EarlyRewards.sol
98e2c87a8edc5e83767a37f7b0e815944a40f41fe0a7a94e388ba4edf3d ./contracts/EcosystemFund.sol
25282897f6b5ecb122cc6b2e9c2d7f29f9d0b4859c4cd1d56633c8db5b693 ./contracts/ERC20Permit.sol
7a4ecd117e4af54d84f8e0a2745bf4d706a3eb8b4d68f0957e652b5d4b14 ./contracts/GovernorAlpha.sol
bd72f5442b82d3963f5cf97fca0a8884a2ab0bd100049a86ad3becc0f71f ./contracts/Idole.sol
6935bf1f9ece9a5e9582f2b6879f7697f0221e6d6f27917c6aa514e6a4d5a9i56 ./contracts/IdleController.sol
cf1b5b51e6d8eb782a2e8939d6c5f59b58e4f150d705f5ae8d763d6964 ./contracts/IdleControllerStorage.sol
af517e1c7773dd6121f4b79a55f96f1070f6d1f9b0a59879cc97b0c38a262b2 ./contracts/Migrations.sol
52382736ef1fc5179087c1aad4fd82406b5d62d9e4972e76a9bab1b28088f7d2 ./contracts/PriceOracle.sol
fd89f9ed026ccba5b2bc8f1f9db09e15e8c682fa58023a3cc1d98e2944166f9392 ./contracts/Reservoir.sol
d96fd3d5dcb78f6a26d3c63c06c69a2f0f9ebec823714906d6569c1a6e6c1cl ./contracts/Timelock.sol
0d1b9949cc926a592d1807d252777e4a50bb5d51ad66a404589e9a27a652d7 ./contracts/Unitroller.sol
9438313d1a242d6eb9c44f43fed2c4f7d2d441de4a01dab358f8e142f1898634 ./contracts/Vester.sol
90696a484f3f8e4a16af4c4a787b78e85f593878788c4580d3bae4291a1 ./contracts/VesterFactory.sol
59129f2b602fc8b4a419fdec2b0f1666786ccfca32c9a5007a878d929e78429 ./contracts/mocks/ERC20Mock.sol
799d938edd021582392378f07e91e33c129fbbceefb35816f52b031634686ff ./contracts/mocks/ChainLinkOracleMock.sol
5f407197c1d1ad19ab360e9f19f59f9035a5d71aebc8c5b06c292c8b6a532a99 ./contracts/mocks/ComptrollerMock.sol
ef0ca415b737386149a1662e18f3e446cb09f9504e708f52b32d72970788a5 ./contracts/mocks/IdleTokenMock.sol
5f0cbdbd4d2db5c5e3a56d657a6bb1710e1d39f3ba5a6ab56e244987d184629 ./contracts/mocks/PriceOracleMock.sol
418f7885175751712e4e3c2541786c73d01a1de045277ab9e083a5960de4 ./contracts/mocks/TokenMock.sol
757c55f81f470204a6edaaed790d6e1a47ecf935edff8ed0d9f11e4e98f23f1ab ./contracts/mocks/TokenMock.sol
709747f9eb09b81953f73d5a2b66a34a35c883ba16b953eace2669c7b5a9134 ./contracts/lib/CarefulMath.sol
1517f9b593a9b86167b8a2293774a248354e422b6b09b192f2c1199f8b8 ./contracts/lib/Exponential.sol
2a65b8ca195868a6cd2d921e44171d0fd5b3aca8ff53ed675f3dbdb4d0e9ae ./contracts/interfaces/CERC20.sol
6585a26353cc8674927494cc75ce7cc5aa4c2575a9280195a63f3aa3829f49e11 ./contracts/interfaces/ChainLinkOracle.sol
04b6fd6ea0e947c7b18c2ca9be03d77827961712b28a2cb7d52d1d57e80b8d7 ./contracts/interfaces/Comptroller.sol
1e141add4cb51a4b8f4ee6515e168cc696e91792b20b233545ae242f42de8 ./contracts/interfaces/Idole.sol

Tests

feach25fac1c088eb3b5f75d71da3e32c58a922b10a0d91b708e37b4c9f1 . ./test/EarlyRewards.js
a746def6e3c49e66c13778ceef757c08968c3c832a45f3f818996ddc891f320 ./test/IdoleController.js
4b009ca4c99f87758e8d58d59807c9a6748c90874dd1da917601bfaf41a6f1 ./test/PriceOracle.js

Changelog

- 2020-10-20 - Initial report
- 2020-10-26 - Reaudit report
About Quantstamp

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

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

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

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

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.