Quantstamp Security Assessment Certificate

February 9th 2021 — Quantstamp Verified

Ideamarket

This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

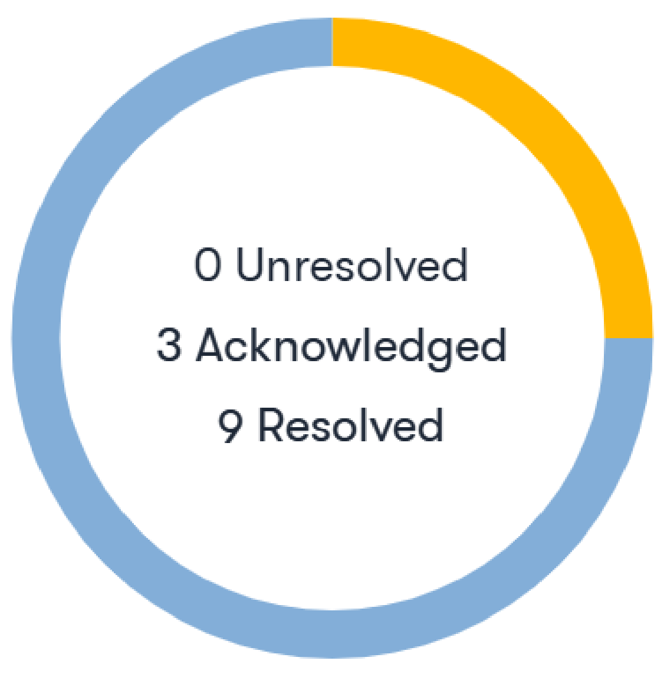
Type	Decentralized Reputation System
Auditors	Ed Zulkoski, Senior Security Engineer Fayçal Lalidji, Security Auditor Kacper Bąk, Senior Research Engineer
Timeline	2021-01-11 through 2021-02-09
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Online Documentation
Documentation Quality	High
Test Quality	High
Source Code	

Repository	Commit
ideamarket-contracts	4f7f0ce

Goals

- Can funds be indefinitely locked or stolen?
- Can the bonding curves be financially gamed?
- Do the smart contracts adhere to provided specifications?

Total Issues	**12** (9 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**2** (2 Resolved)
Low Risk Issues	**5** (4 Resolved)
Informational Risk Issues	**5** (3 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

During the audit, several issues of varying severity were found. Of note, we recommend ensuring that code interfacing with external DeFi protocols behaves as expected. The documentation quality was generally high throughout the project, and the test suite covers a high percentage of the code. We recommend addressing all below issues before deployment.
Update: The Ideamarket team has fixed or acknowledged all concerns as of commit a347315.

ID	Description	Severity	Status
QSP-1	Possible Incorrect Uniswap Pair	^ Medium	Fixed
QSP-2	Unclaimed COMP Tokens	^ Medium	Fixed
QSP-3	Unchecked function arguments	⌄ Low	Fixed
QSP-4	Inconsistent use of `accrueInterest`	⌄ Low	Fixed
QSP-5	Privileged Roles and Ownership	⌄ Low	Acknowledged
QSP-6	Unchecked external function call	⌄ Low	Fixed
QSP-7	Incorrect while-loop condition	⌄ Low	Fixed
QSP-8	Clone-and-Own	○ Informational	Acknowledged
QSP-9	Unlocked Pragma	○ Informational	Fixed
QSP-10	Allowance Double-Spend Exploit	○ Informational	Mitigated
QSP-11	Dependence on external DeFi protocols	○ Informational	Acknowledged
QSP-12	Incorrect Token Transfer Logic	○ Informational	Fixed

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.6.14
- Mythril v0.22.16

Steps taken to run the tools:

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

Findings

QSP-1 Possible Incorrect Uniswap Pair

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `MultiAction.sol`

Description: In `getPathInternal` and `convertInternal` the path generation logic only handles direct pairs, meaning that if the pair does not exist the transaction will throw. Knowing that most pairs in Uniswap uses ETH as a second asset, if a user select any other asset than ETH their transaction may throw if the pair does not exist. Refer to the Uniswap V2 documentation for more details.

Recommendation: A path with an array length equal to three can be used if ETH is not one of the input/output assets if the pair does not exist.

QSP-2 Unclaimed COMP Tokens

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `InterestManagerCompound.sol`

Description: Following compound documentation users should claim the accrued COMP tokens by calling a specific claim function through the compound comptroller. `InterestManagerCompound` implements a function that transfer the COMP tokens to the token recipient but does not implement any function that calls the comptroller to claim the generated tokens.

Recommendation: Add a function to claim the accrued tokens.

QSP-3 Unchecked function arguments

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Ownable.sol`, `IdeaTokenFactory.sol`, `IdeaTokenExchange.sol`, `InterestManagerCompound.sol`, `IdeaTokenVault.sol`, `MultiAction.sol`, `MinimalProxy.sol`, `DSPause.sol`

Description: The following functions do not ensure that certain arguments have non-zero values, which may lead to either incorrect contract initialization, or accidental misuse of functions.

1. `Ownable.setOwnerInternal` does not check if `newOwner` is non-zero.
2. `IdeaTokenFactory.initialize` does not check if address arguments are non-zero.
3. `IdeaTokenFactory.addMarket` does not check that `nameVerifier` is non-zero and likely `priceRise` or `baseCost` should be non-zero.
4. `IdeaTokenFactory.setTradingFee` and `IdeaTokenFactory.setPlatformFee` do not ensure that the corresponding fee rates are less than `FEE_SCALE` (10000). Further, it should be ensured that `tradingFeeRate + platformFeeRate <= 10000`.
5. `IdeaTokenFactory.setNameVerifier` does not check that `nameVerifier` is non-zero.
6. `IdeaTokenExchange.initialize` and `setAuthorizer` do not check if address arguments are non-zero.
7. `InterestManagerCompound.initialize` does not check if address arguments are non-zero.
8. `IdeaTokenVault.initialize` does not check if `ideaTokenFactory` is non-zero.
9. `MultiAction.constructor` does not check that address arguments are non-zero.
10. `MinimalProxy.constructor` does not check that `implementation` is non-zero.
11. `DSPause.constructor` and `DSPause.setOwner` do not check that `owner` is non-zero.

Recommendation: Add `require` statements to each function accordingly.
Update from the Ideamarket team: Fixed except for `priceRise` in `IdeaTokenFactory.addMarket`. Ideamarket might in the future add markets with a constant price (`priceRise = 0`). The missing check in this case is intended.

QSP-4 Inconsistent use of `accrueInterest`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `IdeaTokenExchange.sol`

Description: The function `withdrawTradingFee` does not invoke `accrueInterest`, unlike other functions that redeem from Compound.

Recommendation: Clarify if this is intentional, or add the call to `accreInterest` in `withdrawTradingFee`.

QSP-5 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `IdeaToken.sol`, `ProxyAdmin.sol`, `IdeaTokenExchange.sol`, `IdeaTokenFactory.sol`, `InterestManagerCompound.sol`

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

1. The owner of `IdeaToken` can mint arbitrarily.
2. The owner of `ProxyAdmin` can change the contract logic arbitrarily.
3. The owner of `IdeaTokenExchange` can invoke `setTokenOwner` and `setPlatformOwner` arbitrarily.
4. The owner of `IdeaTokenFactory` can set the trading and platform fees arbitrarily, and also change any name verifiers.
5. The owner of `InterestManagerCompound` may invoke the `redeem*` functions at any point.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner. Limit privileged roles where possible.
Update from the Ideamarket team: The contract system has been designed with upgradeability in mind. All points listed are intentionally designed the way they currently are:

1. The owner of `IdeaToken` can mint arbitrarily: The owner of `IdeaToken` is `IdeaTokenExchange` which mints and burns when tokens are sold and bought. Arbitrary access to mint would need a contract code change which is protected by the `Timelock`.
2. The owner of `ProxyAdmin` can change the contract logic arbitrarily: Having the option to upgrade contract code allows `Ideamarket` to effectively add new features in the future. Contract code changes are protected by the `Timelock`.
3. The owner of `IdeaTokenExchange` can invoke `setTokenOwner` and `setPlatformOwner` arbitrarily: Both functions are used to store the address of the owner of a listing (for example a Twitter account) or platform (for example Twitter) on-chain. As this data is not available on-chain, `Ideamarket` runs a verification service which handles first-time verification of accounts in the system (via the authorizer address). After an owner has initially been set, the authorizer is not allowed to change the owner address anymore. `Ideamarket` still has the possibility to change the owner of the listing, however now the `Timelock` applies (see below). This has been built in as `Ideamarket` will also target crypto newcomers, like publishers which want to use `Ideamarket` to create an income stream but are inexperienced with using Ethereum. We expect these newcomers to often lose access to their private keys. In this case `Ideamarket` can, after thorough verification, restore access to the listing.
4. The owner of `IdeaTokenFactory` can set the trading and platform fees arbitrarily, and also change name verifiers: The trading fees are `Ideamarket`'s main income and thus need a way to be dynamically adjusted. `Ideamarket` might for example decide to decrease the trading fee for a certain market and increase the platform fee by the same amount, thus directing more fees towards the listed platform. Name verifiers can be changed in case a name verifier is not acting correctly. For example, a name verifier might disallow a valid name due to a bug in its code. In this case the name verifier can be updated with a correct implementation Again the `Timelock` applies, see explanation below.
5. The owner of `InterestManagerCompound` may invoke the redeem* functions at any point: The owner of `InterestManagerCompound` is `IdeaTokenExchange` which invests/redeems Dai when `IdeaTokens` are bought or sold. Arbitrary access to the above function would need a contract code change which is protected by the `Timelock`.

All changes made to the system, including contract code changes, need to go through the `Timelock` (`DSPause`) which assures that upcoming changes are publicly visible as queued on-chain for a certain time until they can be executed. Additionally, the access to the `Timelock` is protected by a 2-of-2 Gnosis Safe Multisig controlled by the `Ideamarket` team.

QSP-6 Unchecked external function call

Severity: Low Risk

Status: Fixed

File(s) affected: `IdeaTokenExchange.sol`

Description: In the function `sellTokens`, the external call to `_dai.transfer` is not checked for success.

Recommendation: Wrap the external call to `transfer` in a `require` statement.

QSP-7 Incorrect while-loop condition

Severity: Low Risk

Status: Fixed

File(s) affected: `IdeaTokenVault.sol`

Description: In `getLockedEntries` if `maxEntries` is lower than the user's total number of entries, the function will always return an empty array. This is due to an incorrect check in the second while loop: `len` will be equal to `maxEntries` after the execution of the first while loop, therefore the second loop will always be false since the loop execution requires `len` to be lower than `maxEntries`.

Recommendation: Remove the condiction (`len < maxEntries`) in the second while loop.

QSP-8 Clone-and-Own

Severity: Informational

Status: Acknowledged

File(s) affected: `ERC20.sol`, `proxy/*`, `DSPause.sol`

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability, or may include intentionally or unintentionally modified upstream libraries.

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries.
Update from the Ideamarket team: We have deliberately chosen the clone-and-own pattern for the listed contracts. While we agree that this method is not always the best one to use and that it has certain downsides, we believe that in our case it is preferable:

- Some of the contracts have been modified or extended. This would not have been possible by inheriting from the original upstream contracts as some of them have private state variables and functions which we needed to modify.
- Cloning removes the risk of unknowingly pulling upstream changes into the codebase. This is especially important when working with the Transparent Proxy pattern,

- where even a simple reordering of state variables can break existing on-chain contracts after an upgrade.

QSP-9 Unlocked Pragma

Severity: Informational

Status: Fixed

File(s) affected: `All contracts`

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".

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-10 Allowance Double-Spend Exploit

Severity: Informational

Status: Mitigated

File(s) affected: `ERC20.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

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 it was used 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-11 Dependence on external DeFi protocols

Severity: Informational

Status: Acknowledged

Description: The Ideamarket protocols relies upon several external protocols that must be trusted in order to function correctly, specifically, Uniswap V2, Compound, and any tokens used throughout.

Recommendation: While there is no specific exploit related to these dependencies, we recommend ensuring that users are aware of external dependencies and potential risks through added documentation.
Update from the Ideamarket team: We are aware of the dependence on external DeFi protocols which are mainly Compound for interest generation and MakerDAO for Dai. We will follow Quantstamp's recommendation to improve the user documentation.

QSP-12 Incorrect Token Transfer Logic

Severity: Informational

Status: Fixed

File(s) affected: `InterestManagerCompound.sol`

Description: The function `invest` does not use `ERC20.transferFrom` to handle the `dai` deposits; `dai` tokens must be sent first through an external contract or user call before `invest` can be called.

Recommendation: If `invest` is intended to be called only by the `IdeaTokenExchange`, add `onlyOwner` to avoid any possible issue regarding access restrictions.

Automated Analyses

Slither

Slither reported the following:

1. In `IdeaTokenExchange.getRawPriceForSellingTokens`, it is suggested that all multiplication operations happen before division to avoid any truncation issues. This relates to the operations:

```
        uint priceAtSupply = baseCost.add(priceRise.mul(updatedSupply).div(10**18));
        uint priceAtSupplyMinusAmount = baseCost.add(priceRise.mul(updatedSupply.sub(updatedAmount)).div(10**18));
        uint average = priceAtSupply.add(priceAtSupplyMinusAmount).div(2);
        return hatchPrice.add(average.mul(updatedAmount).div(10**18));
```

1. An analogous issue occurs in `IdeaTokenExchange.getRawCostForBuyingTokens`.
2. It warns of potential reentrancy issues in `IdeaTokenExchange.sellTokens`, however all externally called contracts are known, so we classify this as a false positive.
3. In `IdeaTokenExchange.sellTokens`, the return value from `_dai.transfer` is not checked.
4. In `IdeaTokenExchange.buyTokens`, the return value from `_interestManager.invest` is not checked.
5. In `IdeaTokenExchange.withdrawPlatformFee`, the return value from `_interestManager.redeem` is not checked.
6. In `IdeaTokenExchange.withdrawTradingFee`, the return value from `_interestManager.redeem` is not checked.

Mythril

Mythril did not report any issues.

Code Documentation

1. In `IDSPause.sol`, the `@title` on L5 incorrectly states "IIdeaToken".

Adherence to Best Practices

1. Favor using `uint256` instead of `uint`.

Test Results

Test Suite Results

```
core/IdeaToken
   ✓ admin is owner
   ✓ admin can mint tokens
   ✓ admin can burn tokens (65ms)
   ✓ normal user cannot mint tokens
   ✓ normal user cannot burn tokens (39ms)
   ✓ user can transfer tokens (68ms)
   ✓ user can approve other user (104ms)
   ✓ user can transfer other users tokens (54ms)

core/IdeaTokenExchange
   ✓ admin is owner
   ✓ can buy and sell 500 tokens with correct interest (1176ms)
   ✓ buy completely in hatch (148ms)
   ✓ buy full hatch (143ms)
   ✓ buy partially in hatch (151ms)
   ✓ buy completely outside hatch (310ms)
   ✓ sell completely in hatch (275ms)
   ✓ sell full hatch (276ms)
   ✓ sell partially in hatch (287ms)
   ✓ sell completely outside hatch (321ms)
   ✓ can fallback on buy (246ms)
   ✓ fail buy/sell - invalid token (38ms)
   ✓ fail buy/sell - max cost / minPrice (248ms)
   ✓ fail buy - not enough allowance (87ms)
   ✓ fail buy/sell - not enough tokens (241ms)
   ✓ can withdraw platform interest (367ms)
   ✓ no trading fee available
   ✓ no platform fee available (52ms)
   ✓ no platform interest available (56ms)
   ✓ no interest available (107ms)
   ✓ fail authorize interest withdrawer not authorized
   ✓ fail withdraw interest not authorized
   ✓ fail withdraw platform fee not authorized
   ✓ fail withdraw platform interest not authorized
   ✓ fail authorize platform fee withdrawer not authorized
   ✓ can set factory address on init (185ms)
   ✓ fail only owner can set factory address (177ms)
   ✓ fail cannot set factory address twice (197ms)
   ✓ admin can set authorizer
   ✓ fail user cannot set authorizer
   ✓ authorizer can set interest withdrawer
   ✓ interest withdrawer can set new interest withdrawer
   ✓ fail authorizer cannot set interest withdrawer twice
   ✓ admin can set interest withdrawer twice
   ✓ authorizer can set platform fee withdrawer
   ✓ platform fee withdrawer can set new platform fee withdrawer
   ✓ fail authorizer cannot set platform fee withdrawer twice
   ✓ admin can set platform fee withdrawer twice
   ✓ admin can disable fees for specific token
   ✓ fail user cannot disable fees for specific token
   ✓ correct costs when buying with fee disabled (65ms)
   ✓ correct prices when selling with fee disabled (227ms)

core/IdeaTokenFactory
   ✓ admin is owner
   ✓ can add market (89ms)
   ✓ fail add market with same name (67ms)
   ✓ checks parameters when adding market
   ✓ only admin can add market
   ✓ can add token (132ms)
   ✓ fail add token with invalid name (75ms)
   ✓ fail add token with same name twice (116ms)
   ✓ fail add token invalid market (110ms)
   ✓ can set trading fee (47ms)
   ✓ fail user sets trading fee (40ms)
   ✓ fail set trading fee invalid market (43ms)
   ✓ can set platform fee (58ms)
   ✓ fail user sets platform fee (47ms)
   ✓ fail set platform fee invalid market
   ✓ can set name verifier (51ms)
   ✓ fail user sets name verifier
   ✓ fail set name verifier invalid market (39ms)

core/IdeaTokenVault
   ✓ can lock and withdraw tokens (388ms)
   ✓ has correct locked entries (385ms)
   ✓ can lock with different durations (329ms)
   ✓ fail invalid token (155ms)
   ✓ fail invalid duration
   ✓ fail invalid amount
   ✓ fail invalid until
   ✓ fail too early
   ✓ fail not enough allowance
   ✓ fail not enough balance

core/InterestManagerCompound
   ✓ admin is owner
   ✓ can invest (82ms)
   ✓ fail invest too few dai
   ✓ can redeem (94ms)
   ✓ fail redeem not admin (64ms)
   ✓ can withdraw COMP (93ms)
```

```
core/MultiAction
  ✓ can buy/sell tokens ETH (539ms)
  ✓ can buy/sell tokens WETH (613ms)
  ✓ can buy/sell tokens SOME (560ms)
  ✓ can buy/sell tokens 3-hop (597ms)
  ✓ can buy and fallback (287ms)
  ✓ can buy and lock ETH (292ms)
  ✓ can buy and lock DAI (225ms)
  ✓ can buy and lock DAI with fallback (312ms)
  ✓ can add and buy (270ms)
  ✓ can add and buy and lock (303ms)
  ✓ can convert add and buy (316ms)
  ✓ can convert add and buy and fallback (470ms)
  ✓ can convert add and buy and lock (1524ms)
  ✓ fail buy cost too high (60ms)
  ✓ fail sell price too low (346ms)
  ✓ fail directly send ETH

nameVerifiers/DomainNoSubdomainNameVerifier
  ✓ (empty)
  ✓ test.com
  ✓ abcdefghijklmnopqrstuvwxyz_1234567-89.com (84ms)
  ✓ test.com (with trailing whitespace)
  ✓ test.com (with leading whitespace)
  ✓ test.com (with leading and trailing whitespace)
  ✓ test (no dot and TLD)
  ✓ test. (no TLD) (54ms)
  ✓ . (dot only)
  ✓ .com (no domain)
  ✓ test..com (double dots)
  ✓ example.test.com (subdomain)
  ✓ test!.com (invalid character)

nameVerifiers/MirrorNameVerifier
  ✓ (empty)
  ✓ vitalik
  ✓ vi-talik
  ✓ v-i-t-a-l-i-k
  ✓ Vitalik
  ✓ -vitalik
  ✓ vitalik-
  ✓ -vitalik-
  ✓ VITALIK
  ✓ 12vitalik34
  ✓ (max length) (50ms)
  ✓ (too long)
  ✓ {unallowed ascii char} (1774ms)
  ✓ {allowed ascii char} (287ms)

nameVerifiers/SubstackNameVerifier
  ✓ (empty)
  ✓ vitalik
  ✓ vi-talik
  ✓ v-i-t-a-l-i-k
  ✓ Vitalik
  ✓ -vitalik
  ✓ vitalik-
  ✓ -vitalik-
  ✓ VITALIK
  ✓ 12vitalik34
  ✓ (max length) (41ms)
  ✓ (too long)
  ✓ {unallowed ascii char} (1757ms)
  ✓ {allowed ascii char} (256ms)

nameVerifiers/TwitterHandleNameVerifier
  ✓ (empty)
  ✓ @jack
  ✓ @a
  ✓ @aaaaaaaaaaaaaaa
  ✓ @abcdefghijklmno
  ✓ @pqrstuvwxyz
  ✓ @ABCDEFGHIJKLMNO
  ✓ @PQRSTUVWXYZ
  ✓ @123456789_
  ✓ (empty)
  ✓ @ (@ only)
  ✓ @aaaaaaaaaaaaaaaa (17 chars)
  ✓ @{unallowed ascii char} (1640ms)
  ✓ @{allowed ascii char} (261ms)

spells/AddMarketSpell
  ✓ can add new market (209ms)

spells/ChangeLogicSpell
  ✓ can change logic (144ms)

spells/SetPlatformFeeSpell
  ✓ can set platform fee (194ms)

spells/SetPlatformOwnerSpell
  ✓ can set new platform owner (48ms)

spells/SetTimelockAdminSpell
  ✓ can set new admin (46ms)

spells/SetTimelockDelaySpell
  ✓ can set new delay (43ms)

spells/SetTokenOwnerSpell
  ✓ can set new token owner (40ms)

spells/SetTradingFeeSpell
  ✓ can set trading fee (189ms)

timelock/DSPause
  ✓ admin and user cannot set owner
  ✓ admin and user cannot set delay
  ✓ admin can plot and drop
  ✓ admin can plot and exec (158ms)
  ✓ user cannot plot
  ✓ user cannot drop
  ✓ user cannot exec
  ✓ cannot exec unplotted
  ✓ cannot exec premature
  ✓ cannot disregard delay

timelock/DSPauseProxy
  ✓ fail unauthorized exec
  ✓ fail delegatecall error (46ms)


175 passing (2m)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
compound/	100	100	100	100	
ICToken.sol	100	100	100	100	
IComptroller.sol	**100**	**100**	**100**	**100**	
core/	98.63	81.32	98.73	98.64	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IdeaToken.sol	100	100	100	100	
IdeaTokenExchange.sol	99.4	93.1	100	99.41	133
IdeaTokenFactory.sol	100	83.33	100	100	
IdeaTokenVault.sol	100	86.67	100	100	
InterestManagerCompound.sol	86.84	45	92.86	86.84	… ,99,100,101
MultiAction.sol	100	78	100	100	
core/interfaces/	100	100	100	100	
IIdeaToken.sol	100	100	100	100	
IIdeaTokenExchange.sol	100	100	100	100	
IIdeaTokenFactory.sol	100	100	100	100	
IIdeaTokenVault.sol	100	100	100	100	
IInterestManager.sol	100	100	100	100	
core/nameVerifiers/	98	96.15	100	98	
DomainNoSubdomainNameVerifier.sol	100	100	100	100	
IIdeaTokenNameVerifier.sol	100	100	100	100	
MirrorNameVerifier.sol	100	100	100	100	
SubstackNameVerifier.sol	100	100	100	100	
TwitterHandleNameVerifier.sol	90	83.33	100	90	27
erc20/	81.58	50	70.59	81.58	
ERC20.sol	81.58	50	70.59	81.58	… 183,184,281
spells/	100	100	100	100	
AddMarketSpell.sol	100	100	100	100	
ChangeLogicSpell.sol	100	100	100	100	
SetPlatformFeeSpell.sol	100	100	100	100	
SetPlatformOwnerSpell.sol	100	100	100	100	
SetTimelockAdminSpell.sol	100	100	100	100	
SetTimelockDelaySpell.sol	100	100	100	100	
SetTokenOwnerSpell.sol	100	100	100	100	
SetTradingFeeSpell.sol	100	100	100	100	
test/	100	100	100	100	
TestComptroller.sol	100	100	100	100	
timelock/	100	79.17	100	100	
DSPause.sol	100	75	100	100	
DSPauseProxy.sol	100	100	100	100	
IDSPause.sol	100	100	100	100	
uniswap/	100	100	100	100	
IUniswapV2Factory.sol	100	100	100	100	
IUniswapV2Router01.sol	100	100	100	100	
IUniswapV2Router02.sol	100	100	100	100	
util/	100	66.67	100	100	
MinimalProxy.sol	100	50	100	100	
Ownable.sol	100	75	100	100	
weth/	100	100	100	100	
IWETH.sol	100	100	100	100	
All files	**97.54**	**80.8**	**95.35**	**97.57**	

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

```
8a7931b3bac0b669ada449c60cc32ae9746d69bb93784d50d5d6f91eb5cc64f6 ./contracts/weth/IWETH.sol

39fa9d194d73b0dd5a06f3bb17d0607ed8bd01527a691786e36abb6f6ba5a91c ./contracts/compound/ICToken.sol

e829ab90a544cfbef4bee1b84c7dbb2267fba5d0676a2b3952c5fe586561f723 ./contracts/compound/IComptroller.sol

7d5bc69e3f8beeb7d827877f5db87bde7deffd745120c787e187cf0b5ddc418b ./contracts/spells/SetTimelockDelaySpell.sol

89e57d731a6dd6f44aa215b5594ef79d3858d1cd7793f0398a75d8078ee8f6a0 ./contracts/spells/SetPlatformOwnerSpell.sol

b8b36478d9871c06fc65aa517f385b694ca75fd26a52b003f2d01b2b9a2939e0 ./contracts/spells/SetTradingFeeSpell.sol

c351f2e82472578fe3c8c07acab4a0a92e8ac6adec5617d629d3dcfe3991e65b ./contracts/spells/SetTimelockAdminSpell.sol

825b310c753fd05b93a9844decc0dd3d1295f39fa03cc52d6a84af59e3466586 ./contracts/spells/ChangeLogicSpell.sol

0218b756baf0c27802a352bfaf5f71c36a1c488dbf1376e7ba4cda15caa421f2 ./contracts/spells/SetTokenOwnerSpell.sol

5e516bf03c8f1175c54773eabecb66033ddd6fb49ef52a8c3db027235233ea87 ./contracts/spells/AddMarketSpell.sol

00c5a1c124b58dca2b702ead0421e818f8b31d20f1cd40ad3eb41674b7528705 ./contracts/spells/SetPlatformFeeSpell.sol

046bf40ac0446439f033cd6002314999910ff65a89fee8a4e38cc53a277336a3 ./contracts/erc20/ERC20.sol

6e2f47ef45bd7973879b2ed989e46b4e430b6b5415a2e02005a3d5295a022936 ./contracts/util/Initializable.sol

bac982496a39bffccce725750111101aea7759ae10ca9dcd7e59d32d76cd740a ./contracts/util/MinimalProxy.sol

31c3612f85c6f5e47b853b29926fb39434326c67d170089e6e54abe5f3b59aab ./contracts/util/Ownable.sol

a29e0aefb63716a4243f026f96bd290748ebc4736fe94dcd76eaedd4b939ce4b ./contracts/test/TestUniswapV2Router02.sol

b48a9fa8910ba5033c1d9009e495d2955072edf4d6dd99aa3e39197eca28dbff ./contracts/test/ITestUniswapV2ERC20.sol

08937c2981cfb6a25bb82083b213878e926757a42485fd714897d9b7b9afbc92 ./contracts/test/TestCDai.sol

589a2b44ce07337199bf31b8662c45b92874fae338767db72bb949f7a768a2bc ./contracts/test/TestComptroller.sol

4d0a75d92e09668fd27206df14d9dce133856562bbff006854de608c4d289c4b ./contracts/test/TestUniswapV2Pair.sol

0b9aeeec94247af2baeea176a404ef32b3de462735403e9b4cddb3ff6a7ed16d ./contracts/test/TestTransferHelper.sol

b7c1ae6b7b7a0518350c70e9043550597a2cfce6db9ca868328cc1bbb2724bcf ./contracts/test/UQ112x112.sol

0fe9ba5a6b91fcd999b025ec1749a4699e93fc5656570635e9507dac48f3489d ./contracts/test/TestWETH.sol

7344a505403682368d3093dae9021d0d7fdfa4706e53f052b18a1d906f2384bf ./contracts/test/TestUniswapV2Factory.sol

c4b6ec5144bdb7042dfc53929f4aefbbab5221825fb84a9714f19f8a3eed7af6 ./contracts/test/Math.sol

000bdc13fce3c1482e72f287584b53edb0f2de55c53f1fa289878b27cb740b50 ./contracts/test/ITestUniswapV2Factory.sol

6ca2f67c659cde24d890105a643f1abb31a5d28dca17e13bad5e1da4d0a23eb5 ./contracts/test/ITestUniswapV2Pair.sol

50fcab09ae7bfc1723c79a725fceabd7b3786db779eea8d93164fca0100ec1e8 ./contracts/test/TestERC20.sol

20d18e6f5bbb85e9d1704e223390f1723e77ecd1f072752b5b1d725feeaae721 ./contracts/test/ITestUniswapV2Callee.sol

2f7493bc37548e31de0dcd4e5df5b53d0c1d1b7438490376a3f2f3de12d905e2 ./contracts/test/TestUniswapV2Library.sol

3a3eb4e3d55e399db3190c838e176af392dc71288d9ac9030ff9d2c9e7b21f69 ./contracts/test/TestUniswapV2ERC20.sol

e76389ff74c898efdc6104d7861f0a56922030cc25701292f606192d86d65e21 ./contracts/proxy/AdminUpgradeabilityProxy.sol

0743fadd63af8b52c5a52e3e1e0d3417772bd8794da222cf93ee19686be76dc1 ./contracts/proxy/ProxyAdmin.sol

aa5b0064faf705747aa703cccb4adb12abb1ac2f86ef21fc5dd5cca2fedda363 ./contracts/proxy/Proxy.sol

18185bf89af52cc4d4b79a50852b3da4cdc90f671a37cfe0c99b3ce971d06cf0 ./contracts/proxy/UpgradeabilityProxy.sol

e4a2528ef2f63b3bd5099cc8e5fa271f96aefa76c0f15b0fd9f0c921911aecda ./contracts/core/MultiAction.sol

4e3729376c550abee152f2a2cceb4e4fc1ba7c2f8042b4a33e9866d958c19671 ./contracts/core/IdeaToken.sol

d4f36329d4f7d95aee75efbacd152ea0505643f76e5efe6a8682ea614ab886bb ./contracts/core/InterestManagerCompound.sol

7392aae47f3eedd8dda6efd8b202edbf5c7b815985e61a238009a25a4407ddc5 ./contracts/core/IdeaTokenVault.sol

aab851e6b41a61727b16a7b290a25d3721c403096bae743d5e5018b7d071bb5a ./contracts/core/IdeaTokenFactory.sol

5257aa00ab28482e0bc6a3a2b80f8377ce14f4cb6c7cd94470aabd0a10eb96fe ./contracts/core/IdeaTokenExchange.sol

258496e06e20576f1b2d045783a6b75eda6e99571c3ada81a695ce1ecb847bd4 ./contracts/core/interfaces/IIdeaTokenFactory.sol

3ee96d7a586bcdd01e6aaf7c1ef5a877dcb2f029f96eea10dc892a9ddd5b59b6 ./contracts/core/interfaces/IIdeaToken.sol

2a3295cc1df784bd6e8f10c261312071d861edaabe0d8d78749681aada7a16b0 ./contracts/core/interfaces/IIdeaTokenExchange.sol

2703533630c850f4dd50aba2f971363dfb958e056fc7a5ea7f545f425139eb5b ./contracts/core/interfaces/IIdeaTokenVault.sol

903b05632edbcb3469027922253154140702 17ed3a417b0766b0c76828138d19 ./contracts/core/interfaces/IInterestManager.sol

9174d91d4d07137326796d6a8b5de1ae84ab24c7f0636813bb396d018c497167 ./contracts/core/nameVerifiers/DomainNoSubdomainNameVerifier.sol

c55fe2165d1fbe05c17531273d83ac1cf262cf91c3d4239e3022bd782c649f1b ./contracts/core/nameVerifiers/MirrorNameVerifier.sol

bad7aedaa0ae33ae5f83fd311d9484a7b06ec4376d5a70df85195b28ac28aa27 ./contracts/core/nameVerifiers/SubstackNameVerifier.sol

d70f8ee076f70ebfe03491d6d322ba5a1f02ff44101df9230594f5522faaa6cf ./contracts/core/nameVerifiers/TwitterHandleNameVerifier.sol
```

```
895452dddb78e0cc13ecdc9964d762e27a1b441cbcdf8abc5c288d404956d36a ./contracts/core/nameVerifiers/IIdeaTokenNameVerifier.sol
fe233d21760315e56993d27fab5c67f2ebdb44983d9397fb470a1b9bcb566ca0 ./contracts/timelock/DSPauseProxy.sol
5da80c2fcde0537585267b9a4df4c4e47266f16bc1030cf99bd0ec37c2e429c6 ./contracts/timelock/IDSPause.sol
622d1a495555fb6bf85dc6acc157462dd1fe3123c2950e509f4eee5b4d95cc35 ./contracts/timelock/DSPause.sol
4906a17a33ae04d3c5e2484a2abf560757635fff6e9def01fd01325e0c56b2a5 ./contracts/uniswap/IUniswapV2Router01.sol
8c46984762fe779e33aac5dcd000c49d9e1fcd06dec919bde64343a8a1c4fbc1 ./contracts/uniswap/IUniswapV2Router02.sol
b96c90a8a3f53579a41f1c7b226e9d60d1c8fe3685120244c73bb50d2112705e ./contracts/uniswap/IUniswapV2Factory.sol
```

Tests

```
dd9c1dc254135c92440e2ad9ffd50478843cbdfe8bdc0c59b454a22a9ea43d0c ./test/contracts/nameVerifiers/TwitterHandleNameVerifier.test.js
ea6bdbddc7c5c330ce2baee655cf7e6f7055e68cc9c1d208de68e230a19159d1 ./test/contracts/nameVerifiers/DomainNoSubdomainNameVerifier.test.js
14cbed2985f9aa574ca83860a6c0f7ab84d739037ccad3f70d05a4f7a3e0a2c0 ./test/contracts/nameVerifiers/MirrorNameVerifier.test.js
33366eaa6613db80feaaf88f6e4f7972e3b6b627f59de4f36f0da925e951a3d7 ./test/contracts/nameVerifiers/SubstackNameVerifier.test.js
c7c9367b3e6920a0039aa3b1039c3469d7873c22031ac7fc1aa2ffcb80fe1877 ./test/contracts/spells/ChangeLogicSpell.test.js
ae0ea1d989c54e09dbd3fbcdf6d8faead0b0fe5654ae7c6c7193d0e28053a5a6 ./test/contracts/spells/SetTradingFeeSpell.test.js
4570f0558e2e04e4d2e281a1673ac601cee1babe48518c7cc0ba0b7f133b880c ./test/contracts/spells/SetTokenOwnerSpell.test.js
675613a5cea8d3ca53792eaf534332981cb898ee87aea19a0505d807767d0893 ./test/contracts/spells/SetPlatformOwnerSpell.test.js
40f6d8af71871da163edb69f13410b44d5b1eae19b3a796c54804a8d9cb6bec6 ./test/contracts/spells/SetTimelockAdminSpell.test.js
67e5098071855aa63598485dd723058d70157095f489a6401a7fd16d1847d6ff ./test/contracts/spells/SetTimelockDelaySpell.test.js
ff0edf81a241f1aed17b152efd550e8c1f12c7156bcfb270843e9d9e74097d8d ./test/contracts/spells/AddMarketSpell.test.js
dad40252e177043a00a6f878fe7c5a3fc03deb08ec421fd3b5706113acc4a3aa ./test/contracts/spells/SetPlatformFeeSpell.test.js
11c33a84076b891206d36d90f4efbb724f03d4c686289e74bf5cafbb66ca54f4 ./test/contracts/core/IdeaTokenExchange.test.js
aa20419db30641a41a5c606fe54f8e65c9fb680fac04f9089d3d4f58529202b5 ./test/contracts/core/InterestManagerCompound.test.js
32bdecbecf9fb5a26fee680079792623524788 2e68433544db06c9c788333672 ./test/contracts/core/IdeaToken.test.js
970f7564dd5bde951f3f7ce6ef5a92cc9e0a97ee1120e317828485be5c30e7cf ./test/contracts/core/IdeaTokenFactory.test.js
4082849c14b481299b91dc586fcec73323110a0fd54df139a523312c3f8ca990 ./test/contracts/core/IdeaTokenVault.test.js
542a4ac96e3a0eb308a4123c75170b0998d705e68df19701158c3a7f1a0f147e ./test/contracts/core/MultiAction.test.js
e16d2284a1d94cff7fe38689adaf9183e3871baed4fe25ff6554aea98392e03b ./test/contracts/timelock/DSPauseProxy.test.js
a960f8ad6f58ca6421679719e8a36447149a5d6917c6b69f22cef014f6338152 ./test/contracts/timelock/DSPause.test.js
```

Changelog

- 2021-01-26 - Initial report
- 2021-02-08 - Updated report based on commit `a347315`

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.