Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED · SECURITY CERTIFICATE

August 18th 2021 — Quantstamp Verified

IDEX

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

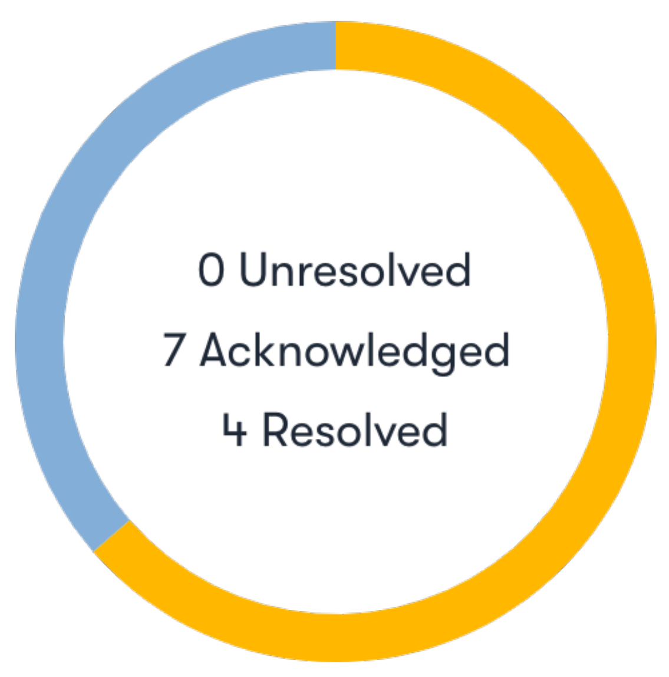
Type	Asset Exchange Protocol
Auditors	Kacper Bąk, Senior Research Engineer Jake Bunce, Research Engineer Joseph Xu, Technical R&D Advisor
Timeline	2021-06-30 through 2021-10-13
EVM	Berlin
Languages	Solidity, Typescript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	IDEX: Silverton Contract Audit Documentation
Documentation Quality	High
Test Quality	High
Source Code	

Repository	Commit
idex-contracts-silverton	ebd8066
idex-farm	1e6fb53

Goals

- Can funds get locked up in the contract?
- Does the implementation deviate from the specification?

Total Issues	**11** (4 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**9** (3 Resolved)
Informational Risk Issues	**2** (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

Quantstamp has conducted an audit of the IDEX Silverton and associated farming contracts. Quantstamp has identified 11 issues, all of which are of either Low or Informational severity. The IDEX team has responded to many of these issues - resolving 4 issues and acknowledging 5 issues at this point.

It is important to note that the reviewed contracts: 1) are assumed to be deployed on Polygon main chain (an L2 solution) only, and 2) we have not reviewed the off-chain part. The latter plays an important role in the project and is proprietary.
Update: the team addressed issues as of commit `403e501`.

ID	Description	Severity	Status
QSP-1	Censorship	Low	Acknowledged
QSP-2	Inflationary and deflationary tokens are incompatible with the protocol	Low	Acknowledged
QSP-3	Small trades and withdrawals may be unprofitable and may drain IDEX funds	Low	Mitigated
QSP-4	No validation if arguments are non-zero	Low	Acknowledged
QSP-5	Loss of precision due to 8 decimal precision	Low	Acknowledged
QSP-6	Possible replay attacks	Low	Mitigated
QSP-7	Privileged Roles and Ownership	Low	Acknowledged
QSP-8	Token decimals and symbols are not guaranteed to be correct with `registerToken()`	Low	Mitigated
QSP-9	Extra pip buffer on the pool price check against the resting sell price	Low	Acknowledged
QSP-10	Sandwich attacks	Informational	Mitigated
QSP-11	User access roles are split between `Exchange.sol` and `Owned.sol`	Informational	Acknowledged

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 v0.8.0

Steps taken to run the tools:

> Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .`

Findings

QSP-1 Censorship

Severity: Low Risk

Status: Acknowledged

File(s) affected: `Exchange.sol`

Description: It possible for IDEX to censor transactions since only IDEX can settle on chain the orders that were submitted off-chain through the order book.

Recommendation: We recommend informing users.

QSP-2 Inflationary and deflationary tokens are incompatible with the protocol

Severity: Low Risk

Status: Acknowledged

File(s) affected: `AssetTransfers.sol`

Description: Transfers of inflationary and deflationary tokens via `transferFrom()` and `transferTo()` will fail because of the checked conditions.

Recommendation: Document this assumption and vet any token contracts before adding them to the platform.

QSP-3 Small trades and withdrawals may be unprofitable and may drain IDEX funds

Severity: Low Risk

Status: Mitigated

Description: Small trades and withdrawals may be unprofitable and may drain IDEX funds because trades need to be settled on chain. It may be especially problematic if the gas price increases rapidly.

Recommendation: Calculate how much liquidity is needed to keep the exchange running for a given time period. Perhaps reject trades that are highly unprofitable. Consider using L2.

Update: The team informed us that the off-chain components of IDEX implement minimum transaction sizes for maker orders, taker orders, withdrawals, and liquidity additions and removals. These minimums prevent unprofitable executions within the enforced fee maximums. Minimums are adjusted over time to account for large changes in gas variation. It's always possible for users to withdraw dust amounts of funds below the withdrawal minimum at their gas expense using a wallet exit.

QSP-4 No validation if arguments are non-zero

Severity: Low Risk

Status: Acknowledged

File(s) affected: `IDEXMigrator.sol`, `IDEXFarm.sol`

Description: Arguments are not checked to be non-zero in`IDEXMigrator.constructor()` and `IDEXFarm.constructor()`.

Recommendation: Add relevant checks.

Update: The team informed us that the IDEX Farm repo is intended to adhere to the original fork's code as much as possible. This approach makes it easy for the community to diff the logic against well-known and trusted production contracts. There are numerous areas where additional checks and best practices may be applied but are eschewed to limit changes.

QSP-5 Loss of precision due to 8 decimal precision

Severity: Low Risk

Status: Acknowledged

Description: The system normalizes assets to 8 decimal points. Operations on assets that operate on more than 8 decimal points may result in the loss of precision and some funds lost.

Recommendation: Inform users.

Update: Documentation covers it here.

QSP-6 Possible replay attacks

Severity: Low Risk

Status: Mitigated

File(s) affected: `Hashing.sol`

Description: Order hashes do not contain contract address and chain Id. A malicious operator could reuse orders across different contracts or chains to perform replay attacks.

Recommendation: Add contract address and chain Id to the hashed orders.

Update: The team informed us that order hashes include a `signatureHashVersion` field which is unique to each IDEX blockchain deployment. 3 is the signature hash version for Silverton on Polygon.

QSP-7 Privileged Roles and Ownership

Severity: Low Risk

Status: Acknowledged

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. Specifically, all upgrades are controlled by an Admin address.

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. Furthermore, consider moving to a DAO to control upgrades of functionality, or make this clear to the users of the service.

Update: The governance documentation covers ownership structure and privileges in detail.

QSP-8 Token decimals and symbols are not guaranteed to be correct with `registerToken()`

Severity: Low Risk

Status: Mitigated

File(s) affected: `AssetRegistryAdmin.sol`

Description: The asset decimals and symbols are specified as an argument and never checked with the ERC-20 asset using `tokenAddress.decimals()` or `tokenAddress.symbol()` within the function `AssetRegistryAdmin.registerToken()`. The confirmation function `AssetRegistryAdmin.confirmTokenRegistration()` matches the new arguments against the previous arguments, so it also does not confirm these parameters with the token contracts themselves. The lack of check on decimals is the more problematic issue.

Recommendation: Although decimals and symbols are optional in ERC-20, but it's a good idea to never add tokens that do not allow confirmation on these two variables in the first place. Furthermore, add relevant checks.

Update: The team explained that the `decimals` accessor is optional in the ERC-20 standard so an explicit check against the contract decimals may limit listable tokens on the exchange. Instead, the design relies on a two-step process of calling `registerToken()` then `confirmTokenRegistration()` to ensure broad token support while minimizing the possibility of incorrect input.

QSP-9 Extra pip buffer on the pool price check against the resting sell price

Severity: Low Risk

Status: Acknowledged

File(s) affected: `HybridTradeValidations.sol`

Description: In `contracts/libraries/HybridTradeValidations.sol`, L97-106 performs a pool price validation check by comparing the pool price against the resting sell price. There appears to be 2 pips worth of buffer as opposed to a 1 pip buffer described in the comments.

```
    // Price of pool must not be better (higher) than resting sell price
    require(
      Validations.calculateImpliedQuoteQuantityInPips(
        baseAssetReserveInPips,
        makerOrder.limitPriceInPips
        // Allow 1 pip buffer for integer rounding
      ) +
        1 >=
        quoteAssetReserveInPips - 1,
      'Pool marginal sell price exceeded'
    );
```

In a previous commit (`c26dfb7`), the snippet responsible for this check (L74-80) was as follows:

```
    require(
      Validations.getImpliedQuoteQuantityInPips(
        baseAssetReserveInPips,
        makerOrder.limitPriceInPips
        // Allow 1 pip buffer for integer rounding
      ) >= quoteAssetReserveInPips - 1,
      'Pool marginal sell price exceeded'
    );
```

It appears that the 1 pip buffer in `quoteAssetReserveInPips - 1` was moved to the other side of the inequality in the latest commit, but the original buffer was not removed from the original side of the inequality.

Recommendation: Double-check the price validation to ensure that the inequality condition is properly specified.

QSP-10 Sandwich attacks

Severity: Informational

Status: Mitigated

File(s) affected: `Exchange.sol`

Description: A common attack in DeFi is the sandwich attack. Upon observing a trade of asset X for asset Y, an attacker frontruns the victim trade by also buying asset Y, lets the victim execute the trade, and then executes another trade after the victim by trading back the amount gained in the first trade. Intuitively, one uses the knowledge that someone's going to buy an asset, and that this trade will increase its price, to make a profit. The attacker's plan is to buy this asset cheap, let the victim buy at an increased price, and then sell the received amount again at a higher price afterwards.

Recommendation: Sandwich attacks are hard to prevent reliably without a user-defined minimum output amount or an oracle-computed price. It's recommended to keep the trade order size low relative to the pool's liquidity to make such attacks economically less attractive, or only use pools for highly liquid tokens.

Update: The team informed us that Silverton's hybrid on-chain, off-chain design eliminates the possibility of sandwich attacks. The off-chain matching engine sequences settlements without the possibility of front-running, and only IDEX is authorized to submit settlement transactions to the Exchange contract.

QSP-11 User access roles are split between `Exchange.sol` and `Owned.sol`

Severity: Informational

Status: Acknowledged

File(s) affected: `Exchange.sol`, `Owned.sol`

Description: Both the `Owned` mixin and `Exchange` are used to define and manage user roles, with `onlyMigrator` and `onlyDispatcher` defined in `Exchange` and `onlyOwner` and `onlyAdmin` defined in `Owned`. Same applies in `LiquidityProviderToken`.

Recommendation: If possible, use a consistent approach for user access preferably through a widely used and audited library to manage the different roles.

Automated Analyses

Slither

Slither failed to analyze contracts due to unresolved imports.

Code Documentation

1. In `Exchange.sol`, L204-205, there are docstrings relating to the event `executeOrderBookTrade`, but this has been removed in commit `ebd8066`. **Update:** fixed.

Test Results

Test Suite Results

All tests executed successfully. Furthermore, we consider the test suite to be comprehensive.

```
Contract: Exchange (tokens)
  registerToken
     ✓ should work (2547ms)
     ✓ should revert when token has too many decimals (2947ms)
     ✓ should revert for ETH address (2296ms)
     ✓ should revert for blank symbol (2386ms)
     ✓ should revert when already finalized (2629ms)
  confirmTokenRegistration
     ✓ should work (2465ms)
     ✓ should revert for unknown token address (2543ms)
     ✓ should revert when already finalized (2758ms)
     ✓ should revert when symbols do not match (2465ms)
     ✓ should revert when decimals do not match (2560ms)
  addTokenSymbol
     ✓ should work (2629ms)
     ✓ should revert for unregistered token (2468ms)
     ✓ should revert for unconfirmed token (2579ms)
     ✓ should revert for reserved ETH symbol (2675ms)
     ✓ should revert for ETH address (2753ms)
  loadAssetBySymbol
     ✓ should work for ETH (2625ms)
     ✓ should work for registered token (2974ms)
     ✓ should revert when no token registered for symbol (3049ms)
     ✓ should revert when no token registered for symbol prior to timestamp (3203ms)
  assetUnitsToPips
     ✓ should succeed (140ms)
     ✓ should truncate fractions of a pip (61ms)
     ✓ should revert on uint64 overflow
     ✓ should revert when token has too many decimals
  pipsToAssetUnits
     ✓ should succeed (146ms)
     ✓ should revert when token has too many decimals

Contract: Custodian
  deploy
     ✓ should work (84ms)
     ✓ should revert for invalid exchange address (87ms)
     ✓ should revert for non-contract exchange address (84ms)
     ✓ should revert for invalid governance address (90ms)
     ✓ should revert for non-contract governance address (106ms)
  receive
     ✓ should work when sent from exchange address (47ms)
     ✓ should revert when not sent from exchange address (38ms)
  setExchange
     ✓ should work when sent from governance address (430ms)
     ✓ should revert for invalid address (115ms)
     ✓ should revert for non-contract address (103ms)
     ✓ should revert when not sent from governance address (96ms)
  setGovernance
     ✓ should work when sent from governance address (246ms)
     ✓ should revert for invalid address (119ms)
     ✓ should revert for non-contract address (101ms)
     ✓ should revert when not sent from governance address (97ms)
  withdraw
     ✓ should work when sent from exchange (103ms)
     ✓ should revert withdrawing ETH not deposited (67ms)
     ✓ should revert withdrawing tokens not deposited (147ms)
     ✓ should revert when not sent from exchange (55ms)

Contract: Exchange (deposits)
   ✓ should revert when receiving ETH directly (338ms)
   ✓ should migrate balance on deposit (2822ms)
  setDepositIndex
     ✓ revert when called twice (367ms)
     ✓ revert when called with invalid value (367ms)
  depositEther
     ✓ should work for minimum quantity (2826ms)
     ✓ should revert below minimum quantity (2547ms)
     ✓ should revert when depositIndex is unset (324ms)
  depositTokenBySymbol
     ✓ should work for minimum quantity (3231ms)
     ✓ should revert for ETH (3080ms)
     ✓ should revert for exited wallet (3221ms)
     ✓ should revert when token quantity above wallet balance (3081ms)
     ✓ should revert for unknown token (2544ms)
  depositTokenByAddress
     ✓ should work for minimum quantity (3296ms)
     ✓ should work for minimum quantity with non-compliant token (3061ms)
     ✓ should revert for ETH (3067ms)
     ✓ should revert for unknown token (2488ms)
     ✓ should revert when token skims from transfer (2788ms)

Contract: Exchange (tunable parameters)
   ✓ should revert when receiving ETH directly (295ms)
  constructor
     ✓ should work (254ms)
     ✓ should revert for invalid balance migration source (221ms)
  loadBalanceInAssetUnitsByAddress
     ✓ should revert for invalid wallet (2248ms)
  loadBalanceInPipsByAddress
     ✓ should revert for invalid wallet (1918ms)
  loadBalanceInPipsBySymbol
     ✓ should revert for invalid wallet (1916ms)
  loadBalanceInAssetUnitsBySymbol
     ✓ should revert for invalid wallet (1911ms)
  loadLiquidityPoolByAssetAddresses
     ✓ should revert when no pool exists (1867ms)
  loadLiquidityMigrator
     ✓ should work (1989ms)
  cleanupWalletBalance
     ✓ should work for valid address (2090ms)
     ✓ should revert if not called by current Exchange (1942ms)
  setAdmin
     ✓ should work for valid address (254ms)
     ✓ should revert for empty address (261ms)
     ✓ should revert for setting same address as current (2057ms)
     ✓ should revert when not called by owner (254ms)
  removeAdmin
     ✓ should work (1969ms)
  setCustodian
     ✓ should work for valid address (1917ms)
     ✓ should revert for empty address (261ms)
     ✓ should revert after first call (1933ms)
  setChainPropagationPeriod
```

```
      ✓ should work for value in bounds (2079ms)
      ✓ should revert for value out of bounds (1950ms)
    setDispatcher
      ✓ should work for valid address (1916ms)
      ✓ should revert for empty address (257ms)
      ✓ should revert for setting same address as current (2033ms)
    removeDispatcher
      ✓ should set wallet to zero (2013ms)
    setFeeWallet
      ✓ should work for valid address (2104ms)
      ✓ should revert for empty address (247ms)
      ✓ should revert for setting same address as current (2004ms)
    setMigrator
      ✓ should work for valid address (2031ms)
      ✓ should revert for invalid address (274ms)
      ✓ should revert for setting same address as current (2040ms)

  Contract: Exchange (exits)
    exitWallet
      ✓ should work for non-exited wallet (2439ms)
      ✓ should revert for wallet already exited (2400ms)
    withdrawExit
      ✓ should work for ETH (2498ms)
      ✓ should revert for wallet not exited (2360ms)
      ✓ should revert for wallet exit not finalized (2525ms)
      ✓ should revert for asset with no balance (2505ms)
    clearWalletExit
      ✓ should work for non-exited wallet (2444ms)
      ✓ should revert for wallet not exited (2357ms)

  Contract: Governance
    ✓ should deploy (82ms)
    setAdmin
      ✓ should work for valid address (119ms)
      ✓ should revert for empty address (125ms)
    setCustodian
      ✓ should work for valid address (2290ms)
      ✓ should revert for empty address (128ms)
      ✓ should revert for non-contract address (117ms)
      ✓ should revert after first call (2393ms)
      ✓ should revert when not called by admin (2400ms)
    initiateExchangeUpgrade
      ✓ should work for valid contract address (2800ms)
      ✓ should revert for invalid contract address (2450ms)
      ✓ should revert for non-contract address (135ms)
      ✓ should revert for same Exchange address (2552ms)
      ✓ should revert when upgrade already in progress (2755ms)
    cancelExchangeUpgrade
      ✓ should work when in progress (2603ms)
      ✓ should revert when no upgrade in progress (2011ms)
    finalizeExchangeUpgrade
      ✓ should work when in progress and addresses match (2361ms)
      ✓ should revert when no upgrade in progress (1894ms)
      ✓ should revert on address mismatch (2125ms)
      ✓ should revert when block threshold not reached (2189ms)
    initiateGovernanceUpgrade
      ✓ should work for valid contract address (2030ms)
      ✓ should revert for invalid contract address (1913ms)
      ✓ should revert for non-contract address (104ms)
      ✓ should revert for same Governance address (2051ms)
      ✓ should revert when upgrade already in progress (2277ms)
    cancelGovernanceUpgrade
      ✓ should work when in progress (2233ms)
      ✓ should revert when no upgrade in progress (2007ms)
    finalizeGovernanceUpgrade
      ✓ should work when in progress and addresses match (2635ms)
      ✓ should revert when no upgrade in progress (2278ms)
      ✓ should revert on address mismatch (2619ms)
      ✓ should revert when called before block threshold reached (2500ms)

  Contract: Exchange (invalidations)
    invalidateOrderNonce
      ✓ should work on initial call (2408ms)
      ✓ should work on subsequent call with a later timestamp (2417ms)
      ✓ should revert for nonce with timestamp too far in the future (2459ms)
      ✓ should revert on subsequent call with same timestamp (2362ms)
      ✓ should revert on subsequent call before block threshold of previous (2503ms)
      ✓ should revert for non-V1 UUID (2287ms)

  Contract: Exchange (trades)
    executeOrderBookTrade
      ✓ should work for matching limit orders (3662ms)
      ✓ should work near max pip value (4504ms)
      ✓ should revert above max pip value (4041ms)
      ✓ should work for matching limit orders with 2 decimal base asset (3921ms)
      ✓ should work for matching limit orders without exact price match (3823ms)
      ✓ should work for matching stop limit orders (3867ms)
      ✓ should work for matching stop market orders (4055ms)
      ✓ should work for matching maker limit and taker market order on quote terms (4104ms)
      ✓ should work for partial fill of matching limit orders (4141ms)
      ✓ should work for partial fill of matching maker limit and taker market order in quote terms (3744ms)
      ✓ should revert for self-trade (3228ms)
      ✓ should revert for limit order with quoteOrderQuantity (3671ms)
      ✓ should revert when fill base net and fee do not sum to gross (4079ms)
      ✓ should revert when fill quote net and fee do not sum to gross (4112ms)
      ✓ should revert for limit order overfill (4586ms)
      ✓ should revert for market order overfill on quote terms (4324ms)
      ✓ should revert when not called by dispatcher (2942ms)
      ✓ should revert for exited buy wallet (2964ms)
      ✓ should revert for exited sell wallet (3047ms)
      ✓ should revert for invalidated buy nonce (3275ms)
      ✓ should revert for invalidated sell nonce (3443ms)
      ✓ should revert for unconfirmed base asset (2622ms)
      ✓ should revert for invalid signatureHashVersion (3037ms)
      ✓ should revert for invalid signature (wrong wallet) (3445ms)
      ✓ should revert for invalid signature (quote order quantity switched) (3217ms)
      ✓ should revert for excessive taker fee (3539ms)
      ✓ should revert for excessive maker fee (3609ms)
      ✓ should revert for zero base quantity (2998ms)
      ✓ should revert for zero quote quantity (3398ms)
      ✓ should revert when buy limit price exceeded (3106ms)
      ✓ should revert when sell limit price exceeded (3189ms)
      ✓ should revert when base and quote assets are the same (3060ms)
      ✓ should revert when maker fee asset not in trade pair (3628ms)
      ✓ should revert when taker fee asset not in trade pair (3502ms)
      ✓ should revert when maker and taker fee assets are the same (3014ms)
      ✓ should revert on double fill (4219ms)

  Contract: Exchange (liquidity pools)
    migrateLiquidityPool
      ✓ should work (4831ms)
      ✓ should work on an existing pool (4662ms)
      ✓ should revert when not called by migrator (3228ms)
      ✓ should revert when pool already exists (3944ms)
      ✓ should revert when liquidity too low (4358ms)
      ✓ should revert when base quantity too low (4575ms)
      ✓ should revert when quote quantity too low (4522ms)
      ✓ should revert when max reserve ratio exceeded (4642ms)
    createLiquidityPool
      ✓ should work (3180ms)
      ✓ should revert when pool already exists (2826ms)
    reverseLiquidityPoolAssets
      ✓ should work (3221ms)
    addLiquidity
      ✓ should work with no fees (6194ms)
      ✓ should work with fees (6040ms)
      ✓ should work for pool with zero reserves (4680ms)
      ✓ should revert when already initiated (5386ms)
      ✓ should revert when wallet exited (5349ms)
      ✓ should revert past deadline (5221ms)
    addLiquidityETH
      ✓ should work with no fees (5852ms)
      ✓ should work with fees (5998ms)
      ✓ should credit balances when to is Custodian (5517ms)
      ✓ should revert when wallet exited (4845ms)
      ✓ should revert when already initiated (5124ms)
      ✓ should revert past deadline (5044ms)
    executeAddLiquidity
      ✓ should work when initiated off-chain (6648ms)
      ✓ should work without price check when reserves below minimum (6445ms)
      ✓ should revert duplicate initiated off-chain (6262ms)
      ✓ should revert when pool price changed (6266ms)
      ✓ should revert when wallet exited (6039ms)
      ✓ should revert invalid signature initiated off-chain (6110ms)
      ✓ should revert when not initiated on-chain (5492ms)
      ✓ should revert on asset address mismatch (6167ms)
      ✓ should revert when minimum base not met (6050ms)
      ✓ should revert when desired base exceeded (5556ms)
      ✓ should revert when minimum quote not met (5726ms)
      ✓ should revert when desired quote exceeded (5609ms)
      ✓ should revert for excessive base fee (5966ms)
      ✓ should revert for excessive quote fee (5859ms)
      ✓ should revert for invalid liquidity (5112ms)
      ✓ should revert for invalid signatureHashVersion (4733ms)
    removeLiquidity
      ✓ should work with no fees (7814ms)
      ✓ should work with fees (6162ms)
      ✓ should credit balances when to is Custodian (5774ms)
      ✓ should revert when wallet exited (4810ms)
      ✓ should revert when already initiated (5363ms)
      ✓ should revert past deadline (5118ms)
    removeLiquidityETH
      ✓ should work with no fees (6902ms)
      ✓ should work with fees (7583ms)
      ✓ should revert when wallet exited (5131ms)
      ✓ should revert when already initiated (5869ms)
      ✓ should revert after deadline (5975ms)
    executeRemoveLiquidity
      ✓ should work when initiated off-chain (7574ms)
      ✓ should work without signature after wallet exit when initiated off-chain (7790ms)
      ✓ should work for fixured dataset 1 (5563ms)
      ✓ should work for fixured dataset 2 (5244ms)
      ✓ should work for fixured dataset 3 (5198ms)
      ✓ should work for fixured dataset 4 (5219ms)
      ✓ should work for fixured dataset 5 (5579ms)
      ✓ should revert duplicate initiated off-chain (7153ms)
      ✓ should revert invalid signature initiated off-chain (6698ms)
      ✓ should revert when not initiated on-chain (6226ms)
      ✓ should revert on asset address mismatch (6578ms)
      ✓ should revert when gross is zero (6550ms)
      ✓ should revert when minimum base not met (6603ms)
      ✓ should revert when minimum quote not met (6818ms)
      ✓ should revert on invalid asset (6589ms)
      ✓ should revert on excessive base fee (5769ms)
      ✓ should revert on excessive quote fee (5755ms)
      ✓ should revert on invalid liquidity amount (5674ms)
      ✓ should revert on invalid base amount (5733ms)
      ✓ should revert on invalid quote amount (5866ms)
      ✓ should revert for invalid signatureHashVersion (5652ms)
    removeLiquidityExit
      ✓ should work (6160ms)
      ✓ should revert when wallet exit not finalized (4055ms)
    skim
      ✓ should work (4928ms)
      ✓ should revert for invalid token address (4146ms)

  Contract: Exchange (liquidity pools)
    executePoolTrade
      ✓ should work with no fees for taker buy (4275ms)
      ✓ should work with fees for taker buy (4387ms)
      ✓ should work with no fees for taker sell (4572ms)
      ✓ should revert for non-zero price correction (4595ms)
      ✓ should revert for unbalanced input fees (5448ms)
      ✓ should revert for excessive input fees (5290ms)
      ✓ should revert for base reserve below min (5632ms)
      ✓ should revert for quote reserve below min (6281ms)
      ✓ should revert for invalidated nonce (5547ms)
      ✓ should revert for exited wallet (4787ms)
      ✓ should revert for decreasing constant product (5758ms)
      ✓ should revert for duplicate trade pair (5000ms)
      ✓ should revert for symbol address mismatch (5205ms)
      ✓ should revert when base quantity is zero (5151ms)
      ✓ should revert when quote quantity is zero (5118ms)
      ✓ should revert when buy limit price exceeded (5161ms)
      ✓ should revert when sell limit price exceeded (5096ms)
      ✓ should revert for excessive output adjustment (4979ms)
      ✓ should revert for excessive gas fee (5092ms)
      ✓ should revert when net quote plus taker fee not equal to gross (5339ms)
      ✓ should revert when net base plus taker fee not equal to gross (5775ms)
      ✓ should revert when order signature invalid (5462ms)
    executeHybridTrade
      ✓ should work with no fees (5276ms)
      ✓ should work with fees (5299ms)
      ✓ should work for taker sell order (5457ms)
      ✓ should work for taker sell order with price correction (6098ms)
      ✓ should work for taker sell order with price correction greater than gross output (5424ms)
      ✓ should revert for quote out with price correction (5572ms)
      ✓ should revert for taker buy with price correction (5200ms)
      ✓ should revert for excessive maker fee (4986ms)
      ✓ should revert for unbalanced orderbook base fees (4860ms)
      ✓ should revert for unbalanced orderbook quote fees (4777ms)
      ✓ should revert for excessive output adjustment (5180ms)
      ✓ should revert for excessive taker fee (5137ms)
      ✓ should revert for taker sell order with excessive price correction (5671ms)
      ✓ should revert for non-zero pool gas fee (4914ms)
      ✓ should revert when pool marginal buy price exceeded (6415ms)
      ✓ should revert when pool marginal sell price exceeded (5750ms)
      ✓ should revert for exited buy wallet (4558ms)
      ✓ should revert for exited sell wallet (4715ms)
      ✓ should revert for self-trade (4333ms)
      ✓ should revert for mismatched trades (4548ms)

  Contract: Exchange (liquidity provider token)
    constructor
      ✓ should work (301ms)
      ✓ should revert for zero Custodian address (299ms)
      ✓ should revert when base and quote address are the same (87ms)
      ✓ should revert when base is not ETH or a contract (81ms)
      ✓ should revert when quote is not ETH or a contract (87ms)
    onlyExchange
      ✓ should restrict burn, mint, and reverseAssets (335ms)

  Contract: Math
    sqrt
      ✓ should work for small values (73ms)
      ✓ should work for max uint64 value (68ms)

  Contract: Exchange (trades)
    executeOrderBookTrade
      ✓ should revert when buy order base asset is mismatched with trade (3394ms)
      ✓ should revert when sell order base asset is mismatched with trade (3340ms)

  Contract: UUID
    getTimestampInMsFromUuidV1
      ✓ should work for current timestamp (70ms)
      ✓ should work for 0 (63ms)
      ✓ should revert for wrong UUID version (63ms)
      ✓ should revert for timestamp before Unix epoch (62ms)

  Contract: Exchange (withdrawals)
    withdraw
      ✓ should work by symbol for ETH (2319ms)
      ✓ should work by address for ETH (2407ms)
      ✓ should work by symbol for token (3032ms)
      ✓ should work by symbol for non-compliant token (2864ms)
      ✓ should deduct fee (3101ms)
      ✓ should revert for unknown token (2325ms)
      ✓ should revert when token skims from transfer (2895ms)
      ✓ should revert for invalid signature (2294ms)
      ✓ should revert for exited wallet (2114ms)
      ✓ should revert for excessive fee (2269ms)
      ✓ should revert for double withdrawal (2306ms)


  314 passing (18m)
```

Code Coverage

The code features an excellent coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	94.03	97.06	93.9	94.44	
Custodian.sol	100	100	100	100	
Exchange.sol	100	100	100	100	
FaucetToken.sol	**0**	**0**	**0**	**0**	**… 44,48,49,51**
Governance.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
LiquidityProviderToken.sol	100	100	100	100	
Owned.sol	100	100	100	100	
contracts/libraries/	100	100	100	100	
AssetRegistry.sol	100	100	100	100	
AssetTransfers.sol	100	100	100	100	
AssetUnitConversions.sol	100	100	100	100	
BalanceTracking.sol	100	100	100	100	
Constants.sol	100	100	100	100	
Depositing.sol	100	100	100	100	
Enums.sol	100	100	100	100	
Hashing.sol	100	100	100	100	
HybridTradeHelpers.sol	100	100	100	100	
HybridTradeValidations.sol	100	100	100	100	
Interfaces.sol	100	100	100	100	
LiquidityChangeExecutionHelpers.sol	100	100	100	100	
LiquidityChangeExecutionValidations.sol	100	100	100	100	
LiquidityPoolAdmin.sol	100	100	100	100	
LiquidityPoolHelpers.sol	100	100	100	100	
LiquidityPools.sol	100	100	100	100	
Math.sol	100	100	100	100	
NonceInvalidations.sol	100	100	100	100	
OrderBookTradeValidations.sol	100	100	100	100	
PoolTradeHelpers.sol	100	100	100	100	
PoolTradeValidations.sol	100	100	100	100	
Structs.sol	100	100	100	100	
Trading.sol	100	100	100	100	
UUID.sol	100	100	100	100	
Validations.sol	100	100	100	100	
Withdrawing.sol	100	100	100	100	
contracts/test/	100	100	100	100	
AssetsMock.sol	100	100	100	100	
BalanceMigrationSourceMock.sol	100	100	100	100	
CleanupExchange.sol	100	100	100	100	
ExchangeLPTokenMock.sol	100	100	100	100	
ExchangeWithdrawMock.sol	100	100	100	100	
GovernanceMock.sol	100	100	100	100	
MathMock.sol	100	100	100	100	
NonCompliantToken.sol	100	100	100	100	
SkimmingTestToken.sol	100	100	100	100	
TestToken.sol	100	100	100	100	
UUIDMock.sol	100	100	100	100	
All files	**98.51**	**99.12**	**97.79**	**98.56**	

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

```
ccb1e7da933def12583f54cd5e5abef17d87d315ebacca4f211692dc24c784f6 ./IDEXFarm.sol
38ce51ae5bf5de7c31f6e1be0f039ea180cae880ae01f3eed5be88ccae279962 ./IDEXFarm_flat.sol
e834c7c225092cdf8515a3628aed1b21f56042384ad92312f1da8941e81317f1 ./Migrations.sol
7b0f1406c1ca78bd7e41a80daf89a4c13ad10e7a026eae15ad5964e44a4897b7 ./IDEXMigrator.sol
97c7480e2ec2f35f8f797ae2e9d8861a86f13b2eeb57f39c5956a0c8dea2cbe5 ./IUniswapV2Pair.sol
adc089e28c02fec9cf10c0ef37b3ba32e98149b179ff39f37b2e05e8f45cbfc9 ./contracts/Exchange.sol
bdde450451ecee3bad79250ca69834a6ca5782c6fe960979a88031da42bc47ea ./contracts/Owned.sol
89bfb4539224d162e500a2ea9ff84a778eed3ad90db537a5bb57e9f778f3ecf6 ./contracts/Migrations.sol
dcf2ca013d0ce825d50837507c9036cc7af12227a4e37571475410c83dc0afd0 ./contracts/Governance.sol
460a62e077e7d9127347b873ad50b8e066603b387508a2c80e3eb334623372a2 ./contracts/Custodian.sol
e2c3f78822530e14e2c70857afcd30264327956afd2bc3f200623674db7b44c5 ./contracts/LiquidityProviderToken.sol
bbb0638bf41f3183052c4c440527369a5b36860920cca19ce2ddd330444d812d ./contracts/FaucetToken.sol
4d485c9ba33126fec3fffb78f57d05ee10480b207ba18b75932216041c7e470a ./contracts/libraries/HybridTradeHelpers.sol
10dbc1a1efabd79786e1369743c573fb4a3a497630c0f6ed137d70cb8950b640 ./contracts/libraries/Depositing.sol
22541600fae41c9d5bf7dd3c82927c87c0f9805c403a110b7428953963180144 ./contracts/libraries/Structs.sol
33be935ffec846a962e01f53ba1bf5278ce692b993190329c70e76cea55dd770 ./contracts/libraries/OrderBookTradeValidations.sol
f9276c25e7107ba7cf2c04afa247d36ba4510410307ede94238274ba6e43aecf ./contracts/libraries/Hashing.sol
e676af54f8a3d82267fa3a2912fe22781185a12fe60e20a74d137ae9ff41dd97 ./contracts/libraries/PoolTradeValidations.sol
e5017ba4600f63b32e55e1ac2f76d0b3fc2519816ba023830635a1a2bfc211b2 ./contracts/libraries/LiquidityChangeExecutionHelpers.sol
8c31a322327128589804f9642ed1b74b8135d01888031343d1f3be66d0f96a31 ./contracts/libraries/Trading.sol
dfff8ed4cf749dc3d1a18c1bb098538138ee3f9817e3fb24eef70b4e6ea190b6 ./contracts/libraries/Enums.sol
5d8a32cf1227267653832874d42ea47e31b6dfcfa650660b867237c81ca57846 ./contracts/libraries/HybridTradeValidations.sol
72a9b289e99086f3fce8b11227cfebc8f5191dc48db01720d63515663c01e27d ./contracts/libraries/UUID.sol
110d0f82f27b792ef2ffb8a8c97e8aaebb6bc39acbf93c8168a47a41135b1e76 ./contracts/libraries/LiquidityPools.sol
01274c424f4f67b599a31343a35a521966399232605dcfc000ccc5a599688ccc ./contracts/libraries/Withdrawing.sol
b646de15255b3ae15556676e1375bee4ef3d0f9977983aacb5d8a746eb933f88 ./contracts/libraries/AssetRegistry.sol
f5a7a547123b0434a13adf6166ddca1d1e2ea3eb11f6d877d3347d35a991d128 ./contracts/libraries/Validations.sol
2d6acc75938f2ff4e7b0b8e1301255abc25885f23a955c482550ca690cef8b94 ./contracts/libraries/AssetTransfers.sol
859d01898216029af0b8fcba415c94457304c8495329678b69b9c839d9ea9c37 ./contracts/libraries/Constants.sol
68ef7b74f814d4bd22f630eced4fcad51b58ac03a18f616c8d7f747986b9e289 ./contracts/libraries/AssetUnitConversions.sol
29dddb421da57dd924af5d09a072a42b1476365179c70d87480d9be5794c6fb5 ./contracts/libraries/Math.sol
6f5dbee4070536a3c31cd8ba158179e7ab10c20c1fb815115423740cbe20e506 ./contracts/libraries/PoolTradeHelpers.sol
49001326c9f1e974623e18d399fe21c05cee06f6f2a6a2e782c156867331a52e ./contracts/libraries/LiquidityPoolHelpers.sol
c14ec3efd9e8f87a31f73758c87b1a8858d77e2827910747e534e7bc0c12c589 ./contracts/libraries/LiquidityPoolAdmin.sol
724cdc2d49d2d2c4d1375ea074238b31c89752251a8ce5fcc66b6b38c73be910 ./contracts/libraries/BalanceTracking.sol
eca22eea1d76650304065da7be68a9d3707dd64f1770db5f2952805fb38d7a0a ./contracts/libraries/LiquidityChangeExecutionValidations.sol
e2bf59cba18ce12fc023de0ca1cb308d67b15e9a2e8aa5c889dfe9db9126994c ./contracts/libraries/NonceInvalidations.sol
4a01b0742c8972a762ea64853420e3ab23a1131c66a075b0dacea05b8e7365da ./contracts/libraries/Interfaces.sol
171dfb1ffcd063ecbe46b5ff0e90b997cdcdc82ca86fe50600c98dd66613bd8d ./contracts/test/UUIDMock.sol
66f610cf45cc4d3b9271faaa7c7a9afeb9bd433f8e58abaa0bd54423af3b7606 ./contracts/test/ExchangeWithdrawMock.sol
cf07b5c9976c9cfb2668146ec1d5ddc424b5ade5691305b01eb28c8a4dc38386 ./contracts/test/SkimmingTestToken.sol
1acf793d46c8a47f794d108a88d4b5af99b0fec025a2fe158667cf33e0dc78d2 ./contracts/test/NonCompliantToken.sol
00e125ab644e3471df44a532c234c8ebd1798e2a7d4661aa9cb7f9d316cd5149 ./contracts/test/CleanupExchange.sol
5415b4052000c78f51d575720ad5ddd97bb6505f269c1ac1911235cc810139c2 ./contracts/test/BalanceMigrationSourceMock.sol
9eade7b3822cfb583141eb655e320e225fea31cac70ff5b0561191eccba9b756 ./contracts/test/ExchangeLPTokenMock.sol
ee1d7e4f33479f988366b3ff4bd7b4aac3c2af90bc5c7b4bbf4cc55bd92e1ae6 ./contracts/test/TestToken.sol
141ed322ff8043bbc745ca4e47ac643e56f3b98a2ce302b1533532dc840d670c ./contracts/test/GovernanceMock.sol
846e9804b536d6a8f15b1d6bd3ab61e00daf5640bfccc7d496f49160a4e5cec0 ./contracts/test/MathMock.sol
ed9ada392502f15eab0724d6e0f9524533a40281d28733686bf5ac8f729d9284 ./contracts/test/AssetsMock.sol
```

Tests

```
7a00945e631f16afb2d1edf746b91c91f429bd4536420638ff4bc1ffd12a2074 ./tests/.eslintrc.js
621813b054847a557c2120247b2d3899a49b29b2f5e87f2b43b3a62a5513af46 ./tests/lib/index.test.ts
c49531020d66840cf02c5f91c511a4675707a47be3c407ed6d52d9f0f08d984b ./tests/contracts/trade-block-time.test.ts
81fd83afa4033f29a1189619fa53f5ea3b321ccc7c69d8a9c0ee6c6c84d7c8c7 ./tests/contracts/liquidity-pools.test.ts
5bf2217bd06960c8339d1f00e4cd44e3dee3b91272f7fb0206bb751f28434a8c ./tests/contracts/custodian.test.ts
676e39da83df13880dc0e4e3506d7fe66defb72f5615f52b95f5e34132166558 ./tests/contracts/helpers.ts
```

```
e9209d0c410444ef152eefd7fedef89856d4c0ccd5eba247d573156d5d62f4f0 ./tests/contracts/invalidate.test.ts
92c88afcef3631691fd91cbdf9df834868706d5c742788f7460769150f5179b0 ./tests/contracts/liquidity-pool-trades.test.ts
01d3adaf28b42075466a1e4f91e6dcdb9be554aa74008bdaf6f574e6a9317fa2 ./tests/contracts/uuid.test.ts
1cc3216139b46b2e07a790e8b10134668291b1da59abdd7b0ddfc46ed770aa88 ./tests/contracts/exit.test.ts
6d51370e25ddfba1b6d28ffac678fdae919147f3cdeb7fd77ae65a13e4676252 ./tests/contracts/deposit.test.ts
6afb9ccb99e01d629a9ca7dd0ed8a02881643a7d31ad17a29bff57049878c985 ./tests/contracts/trade.test.ts
88ad78327f36a3a1badafb4e614978114b71418acca8ac189a6ee7e9b6654916 ./tests/contracts/.eslintrc.js
c9d61e4fa58e2552f21db98a99ccba4f4f85001f5db24366d2504e0181963fa0 ./tests/contracts/liquidity-provider-token.test.ts
d92591041e6147f121cb0e0d8826b405ac321203a67cccc1d422cb4104747c75 ./tests/contracts/exchange.test.ts
a9e39b97b37fcc6a0b7805d20760ee648fea5d6f3141e37de63c855d120e174b ./tests/contracts/math.test.ts
33b7ad79770f5c863ff1f4992d25c3afa805ec844099b7f5f76c40af6265b911 ./tests/contracts/governance.test.ts
25bbf8d23a78e104a3dd6368c028e2d25d1bdbc1420383e325db35e441412db2 ./tests/contracts/assets.test.ts
bf86e56a241d2c42a55600d28cb4a427acb1d2aa177908b49edd2acb288ee343 ./tests/contracts/withdraw.test.ts
```

Changelog

- 2021-08-18 - Initial report
- 2021-10-13 - Revised report based on commit `403e501`.

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.