Quantstamp Security Assessment Certificate

May 10th 2021 — Quantstamp Verified

MCDEX

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

Executive Summary

Type	Decentralized Exchange
Auditors	Jan Gorzny, Blockchain Researcher Joseph Xu, Technical R&D Advisor Poming Lee, Research Engineer
Timeline	2021-02-25 through 2021-04-20
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Reference Document Speficiation PDF
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
mai-protocol-v3	50fb550

Total Issues	**19** (13 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	**3** (2 Resolved)
Low Risk Issues	**13** (9 Resolved)
Informational Risk Issues	0 (0 Resolved)
Undetermined Risk Issues	**2** (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

The protocol is a complicated system with several functionalities. On the whole, the system appears to do what it is intended to do, but Quantstamp has identified several issues which have since been resolved or acknowledged. The issues range from high-severity concerns to inconsistencies with documentation and specification. Tests back up the idea that the code is functions as intended, but the issues found suggest that additional testing would be helpful for future development and maintenance.
Please note that the `governance`, `test`, and `reader` directories were not in scope for this audit and as such, have not been reviewed.

ID	Description	Severity	Status
QSP-1	Dangerous External Calls From `Broker.sol`	High	Fixed
QSP-2	Trade Price Can Be Zero	Medium	Fixed
QSP-3	No Backup Oracle Nor Protection From Erroneous Price Data	Medium	Acknowledged
QSP-4	Trading Privilege Check Returns Incorrect Results	Medium	Fixed
QSP-5	Ambiguous Liquidity Pool Ownership And Operator Privilege Control	Low	Fixed
QSP-6	Contract Stores Its Own Balance For Calculation Instead Of Utilising Real-Time Token Contract Value	Low	Acknowledged
QSP-7	Possible Reward Truncation	Low	Acknowledged
QSP-8	`calculateCashToReturn` Does Not Exclude Perpetuals	Low	Acknowledged
QSP-9	`rebalance` Does Not Have a Return Value	Low	Fixed
QSP-10	Unused Functions	Low	Acknowledged
QSP-11	Incorrect Constants	Low	Fixed
QSP-12	Incorrect Data Representation	Low	Fixed
QSP-13	Omitted Event	Low	Fixed
QSP-14	Missing Argument	Low	Fixed
QSP-15	Gas Usage / `for` Loop Concerns	Low	Mitigated
QSP-16	Missing Input Validation	Low	Fixed
QSP-17	Integer Overflow / Underflow	Low	Mitigated
QSP-18	Possibly Incorrect Revert Condition In `validateBaseParameters`	Undetermined	Fixed
QSP-19	End User Can Pretend To Be A Broker Or Relayer	Undetermined	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.

Findings

QSP-1 Dangerous External Calls From `Broker.sol`

Severity: *High Risk*

Status: Fixed

File(s) affected: `contracts\broker\Broker.sol`

Description: The function `callFunction`, can be called by any user to call any external contracts. This enables a user to have privilege of the `Broker.sol` contract and could be used as a tool to conduct a complex attack.

Recommendation: Consider limiting the target calling contracts of this function, or the `msg.sender` of this function. Otherwise, state this risk explicitly to your public document.

Update: The related files have been removed, so this issue is no longer relevant.

QSP-2 Trade Price Can Be Zero

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts\module\TradeModule.sol`

Description: Within the function `validatePrice`, the trade price can be set to zero. This could lead to devastating results.

Recommendation: Exclude this condition by requiring this value be non-zero.

Update: A check that the trading price must be greater than zero has been added.

QSP-3 No Backup Oracle Nor Protection From Erroneous Price Data

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `contracts\module\PerpetualModule.sol`, `contracts\interface\IOracle.sol`

Description: According to `updatePrice` in `contracts\module\PerpetualModule.sol` and `contracts\interface\IOracle.sol`, the oracles that the current system uses do not have any backup. The system only collects price data of each `path` from one oracle so there is no backup oracle. The system could fail to work correctly when any of the oracles is operating abnormally or being manipulated.

Recommendation: Add more than one oracle for each `path` in order to increase the security level. Also, consider adding some sanity checks to the collected price data.

Update: The code in `PerpetualModule.sol` for price reading logic will be left unchanged. Currently the price check will be done in `updatePrice` (non-zero). The team will add more validations in the wrapper layer of oracle once we have more data suppliers.

QSP-4 Trading Privilege Check Returns Incorrect Results

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts\modules\LiquidityPoolModule.sol`

Description: The sequence of the arguments in `isAuthorized` for calling `IAccessControl.isGranted` on `L814` is incorrect. The sequence of `grantee` and `grantor` is switched.

Recommendation: Switch the sequence of `grantee` and `grantor`.

Update: `grantor` has been renamed to `grantee`, fixing the issue.

QSP-5 Ambiguous Liquidity Pool Ownership And Operator Privilege Control

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\factory\Tracer.sol`

Description: Within the function `setLiquidityPoolOwnership`, there is a permission check so that it can only be called by a liquidity pool instance. However, there seems to be a difference in being the "owner" of a liquidity pool vs the "operator" of a liquidity pool.
The operator privilege is transfered via `transferOperator` and `transferOperator`, but this does not automatically transfer liquidity pool ownership.

Recommendation: Clarify the intended privileges.

Update: The function `setLiquidityPoolOwnership` has been renamed to `registerOperatorOfLiquidityPool`, resolving this issue.

QSP-6 Contract Stores Its Own Balance For Calculation Instead Of Utilising Real-Time Token Contract Value

Severity: *Low Risk*

Status: Acknowledged

Description: For `account.cash` and `collateralToken`, the system stores its own balance data for calculation instead of utilising real-time token contract data. This could lead to faulty calculation when non-standard tokens (e.g., those which may be deflationary, involve rebasing, or have wrong return values from `transfer`) are used.

Recommendation: Please make sure to use only ether and standard ERC20 tokens (which are inspected manually or otherwise approved to conform to the system requirements) in the system, or do not use this method to compute balances.

Update: The team will add a check on the front end to inform users.

QSP-7 Possible Reward Truncation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts\governance\RewardDistribution.sol`

Description: `L61` in `setRewardRate` may truncate unfinished reward distribution.

Recommendation: If this cannot be avoided, ensure that this is leads to negligible losses and make the issue known to users. Consider adding a test case to confirm that truncation only occurs in acceptable cases, if any.

Update: The team will add a case for reward truncation and will add notifications to inform users if necessary.

QSP-8 `calculateCashToReturn` Does Not Exclude Perpetuals

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts\modules\AMMModule.sol`

Description: The function `calculateCashToReturn` assumes that the context is fully prepared without excluding any perpetuals.

Recommendation: Considering adding a check for `context.indexPrice = 0 && context.position = 0` to avoid calculation using the wrong context.

Update: The team has acknowledged the issue.

QSP-9 `rebalance` Does Not Have a Return Value

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\modules\LiquidityPoolModule.sol`

Description: As in the title, `rebalance` does not have a return value; this may lead to undesirable results or wasted computation.

Recommendation: Consider adding a return value that matches the movement of collateral from perpetual to pool to allow checking for whether `rebalance` actually took place or not.

Update: The team added a signed return value to indicates the rebalanced amount.

QSP-10 Unused Functions

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts\modules\LiquidityPoolModule.sol`

Description: Functions `transferFromUserToPool` and `transferFromPoolToUser` are unused. There are also comments indicating risks of incorrect calculation if used in `addLiquidity` or `removeLiquidity`.

Recommendation: Consider removing these functions.

Update: The team has acknowledged the issue.

QSP-11 Incorrect Constants

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\modules\PerpetualModule.sol`

Description: There is an error in `L759-761` of `validateBaseParameters`: `INDEX_OPEN_SLIPPAGE_FACTOR` is used instead of `INDEX_MAINTENANCE_MARGIN_RATE` to validate the liquidation penalty parameter. Having said that, both constants are set to 1 so the result is the same.

Recommendation: Check that the values of the constants are what they should be.

Update: `INDEX_OPEN_SLIPPAGE_FACTOR` has been renamed to `INDEX_MAINTENANCE_MARGIN_RATE`, resolving the issue.

QSP-12 Incorrect Data Representation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\modules\TradeModule.sol`

Description: There is a difference in the representation of `int256 deltaPosition` between `liquidateByAMM` and `liquidateByTrader`, which may cause confusion in the future. Specifically, `deltaPosition` in `liquidateByAMM` represents the position to be taken on by the AMM, which is the trader's current position. In contrast, `deltaPosition` represents the change in the trader's position in `liquidateByTrader`.

Recommendation: To properly account the two different `deltaPosition`, `L260` in `postLiquidate` should have the last argument as `deltaPosition.neg` and `L324` `liquidatedAmount` should equal `deltaPosition` without `.neg`. The `postLiquidate` argument won't cause problems though, since the function ends up using `deltaPosition.abs` throughout.

Update: From the team: "The `deltaPosition` represents the opposite meaning in the two functions, so we tried to unify the perspectives to improve the readability of `liquidateByTrader` function." Therefore this issue is resolved.

QSP-13 Omitted Event

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\LibraryEvents.sol`

Description: `SetOracle` event from `module/PerpetualModule.sol` is not included.

Recommendation: Include the event.

Update: The `SetOracle` event has been added.

QSP-14 Missing Argument

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\LibraryEvents.sol`

Description: `OperatorCheckIn` event should contain `address indexed operator` as an argument.

Recommendation: Add the argument.

Update: The argument has been added to `OperatorCheckIn`.

QSP-15 Gas Usage / `for` Loop Concerns

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `contracts\broker\Broker.sol`, `contracts\module\AMMModule.sol`, `contracts\module\LiquidityPoolModule.sol`, `contracts\oracle\router\OracleRouter.sol`, `contracts\symbolService\SymbolService.sol`

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.
There are several functions which use a `for` loop in the listed contracts.

Recommendation: Ensure that these loops will not exceed any limits on computation time.

Update: The team added a limitation to the max perpetual count in one liquidity pool, and added a test file `LoopTest.test.ts` to show the max gas consumption before deploying.

QSP-16 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\Governance.sol`, `contracts\broker\Broker.sol`, `contracts\factory\PoolCreator.sol`, `contracts\symbolService\SymbolService.sol`

Description: Several functions do not validate that input arguments are values are trivially incorrect, such as addresses possibly being zero. The following list may be incomplete.

1. `Governance.sol`: `transferOperator` does not check that input address is non-zero.
2. `Broker.sol`: `_transfer`, `callFunction` do not check that input addresses are non-zero.
3. `PoolCreator.sol`: Constructor, `_createLiquidityPoolWith` do not check that addresses are non-zero.
4. `SymbolService.sol`: `isWhitelistedFactory` does not check if input address is non-zero.

Recommendation: Add `require` statements to check if these arguments are not these potentially problematic cases.

Update: The team added `isContract` and non-zero address check to listed functions.

QSP-17 Integer Overflow / Underflow

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `contracts\broker\Broker.sol`, `contracts\libraries\EnumerableMapExt.sol`, `contracts\symbolService\SymbolService.sol`

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 `uint8` variables, meaning 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 this project, some contracts do not use SafeMath everywhere to protect against this issue.

1. `Broker.sol`: `SafeMath` not used on `L321`, `L179`.
2. `EnumerableMapExt.sol`: `SafeMath` not used in several places.
3. `SymbolService.sol`: `SafeMath` not used on `L133`.

Recommendation: Use `SafeMath` in these contracts.

Update: In `Broker.sol` the function was removed, the issue was fixed in `SymbolService`, and the issue was acknowledged in `EnumerableMapExt`.

QSP-18 Possibly Incorrect Revert Condition In `validateBaseParameters`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `contracts\module\PerpetualModule.sol`

Description: In `L759-762`: the revert message is inconsistent with the revert condition.

Recommendation: Change the messages or fix the revert conditions.

Update: This issue is resolved, as it is a duplicate of a previous issue which has been fixed.

QSP-19 End User Can Pretend To Be A Broker Or Relayer

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `contracts\Perpetual.sol`

Description: In the function `brokerTrade`, a user can interact with `Perpetual.sol` directly by giving it a manipulated `OrderData` that seems to be sent by a broker, from a relayer.

Recommendation: We have no recommendation at this time.

Update: From the team: "It is ok for a user to be the broker. The broker is designed for the third party who supplies the service to relay sign transactions and charge for gas fee. Calling the method directly won't make extra profit or causing harms."

Adherence to Specification

1. `contracts\module\AMMModule.sol`: `function ammOpenPosition`: the first term in the `max{}` of the upper part of formula 10, and the first term in the `min{}` of the lower part of formula 12 in `AMM.pdf`, were not implemented.
2. `contracts\module\AMMModule.sol`: `function ammClosePosition`: the first term in the `max{}` of the lower part of formula 10, and the first term in the `min{}` of the upper part of formula 12 in `AMM.pdf`, were not implemented.
3. `contracts\module\AMMModule.sol`: the comment "If price is better (for trader) than best price, change price to best price" should be "If price is better (for the AMM) than best price, change price to best price" instead.
4. `contracts\module\AMMModule.sol`: `queryTradeWithAMM`: `L68-L70`: `L68` calculates `bestPrice` and applies it to the determination of `deltaCash` on `L70`. This design does not adhere to the specification. Please double check to make sure this is intended and is correctly designed.

Update: All issues above have been clarified and are considered resolved.

Code Documentation

Inline comments are perhaps better than average, but not always clear. For example, the comments in `liquidateByAMM` of `TradeModule.sol` do not clearly match the code. We also found the following specific places documentation could be improved.

1. `contracts\Readme.md`: "Pool create also manages global variables for all created liquidity pools." should contain the text "creator".
2. `contracts\Readme.md`: "Contracts of pool creator is located in contracts/LiquidityPool.sol." should contain the text "of liquidity pool".
3. `contracts\Readme.md`: "Contracts of pool creator is located in contracts/Perpetual.sol." should contain the text "of perpetual".
4. `contracts\Readme.md`: "Contracts of pool creator is located in contracts/module/AMMModule.sol." should contain the text "of AMM".
5. `contracts\Readme.md`: "Contracts of pool creator is located in contracts/governance." should contain the text "of governance".
6. `contracts\Getter.sol`: `L299` there is a typo: `amoun`.

Adherence to Best Practices

1. `contracts\Type.sol`: For variables in `struct PerpetualStorage` and should always be `>=0`, such as `syncFundingInterval`, consider just making them `uint256` instead to avoid edge cases. **Update**: Acknowledged.
2. For variables that should always be `>=0`, such as price data (e.g., index price, mark price), consider just making them `uint256` instead of `int256` to avoid edge cases. **Update**: Acknowledged.
3. `contracts\module\LiquidityPoolModule.sol`: `L476`, updating the price before checking if the oracle is terminated doesn't make sense. **Update**: Acknowledged.
4. `contracts\module\LiquidityPoolModule.sol`: `L30-L34` have been repeated. **Update**: Fixed
5. `contracts\modules\LiquidityPoolModule.sol`: Incorrect comment on `L322` for `updatePerpetualRiskParameter`. This function does not take max or min values for risk parameters. **Update**: Fixed.
6. `contracts\modules\PerpetualModule.sol`: Typo on `L43` (and elsewhere) - `INDEX_HARF_SPREAD`. **Update**: Fixed.
7. `contracts\module\PerpetualModule.sol`: when `position == 0`, shouldn't the `perpetual.fundingRate` be set to zero? **Update**: Resolved - not an issue.
8. `contracts\module\CollateralModule.sol`: `L88` consider checking if the balance in the ERC20 contract is changed correctly. **Update**: Fixed.

9. `contracts\module\CollateralModule.sol`: `L118` consider checking if the balance in the ERC20 contract is changed correctly. **Update**: Fixed.

10. `contracts\module\TradeModule.sol`: `L72`: consider checking and reverting if an oracle is already terminated here. **Update**: Acknowledged.

11. `contracts\modules\TradeModule.sol`: Redundant calculation of `int 256 totalFee` on `L152` (previously done on `L144`). **Update**: Fixed.

12. `contracts\module\AMMModule.sol`: `L250` shouldn't also update the `bestPrice` if `deltaCash` is changed? **Update**: Acknowledged.

13. `contracts\Perpetual.sol`: `function liquidateByTrader`: consider checking if the `msg.sender` is a `trader` before performing any further operations. **Update**: Acknowledged.

14. `contracts\Perpetual.sol`: Typo on `L58`: `Constant.PRIVILEGE_DEPOS[TI]`. **Update**: Fixed.

15. `contracts\factory\Variables.sol`: `L86-90`: comments for `getMCBToken` are incorrect. The function gets the address of the MCB token contract as opposed to the symbol service. **Update**: Fixed.

16. A TODO is unfinished in `contracts\factory\Variables.sol`: `L93`. **Update**: Fixed.

17. `contracts\symbolService\SymbolService.sol`: `L68` when a factory is removed from the whitelist, all the `liquidityPool`s created by it cannot pass `modifier onlyWhitelisted` anymore. Please check if it is intended. **Update**: Acknowledged.

18. `contracts\broker\Broker.sol`: `L130`: a `order.relayer` can cancel all the orders it relayed. Please make sure if this is intended. **Update**: Acknowledged.

19. `contracts\governance\GovernorAlpha.sol`: `L229-230`: comments do not match the implementation. Critical functions require more share tokens to reach quorum but do not require more share tokens to be proposed. **Update**: Fixed.

20. `contracts\governance\RewardDistribution.sol`: Redundant call to `earned` on `L130` when `rewards[account] = earned(account);` is already computed in the modifier. **Update**: Acknowledged.

21. `contracts\governance\RewardDistribution.sol`: Incorrect comment on `lastTimeRewardApplicable`. The variable `periodFinish` is expressed in block number as opposed to in timestamp. **Update**: Fixed.

22. `contracts\interface\IAccessControll.sol`: Typo on the interface name with an extra 'l' at the end. **Update**: Fixed.

23. `contracts\libraries\Constant.sol`: Typo on `L10` and `L14` - `PRIVILEGE_DEPOS[TI]`. **Update**: Fixed.

24. `contracts\libraries\Utils.sol`: The comment on `extractSign` should indicate the case for zero (1 means the number is non-negative). **Update**: Fixed (the function has been removed).

25. `contracts\LiquidityPool.sol`: Incorrect comment on `L138-139` for `donateLiquidity`. The LP does not receive share tokens in this case. **Update**: Fixed.

Test Results

Test Suite Results

Some test output has been removed for brevity. Two test files (`SymbolService.test.ts` and `Trade2.test.ts`) were skipped in order to run tests.

```
AccessControl
   ✓ privileges (470ms)

AMM
  isAMMSafe
     ✓ init - ok (54ms)
     ✓ flat - ok (38ms)
     ✓ short - ok
     ✓ short - fail
     ✓ long - ok
     ✓ long - fail
  getPoolMargin
     ✓ success-0 (40ms)
     ✓ success-1
     ✓ success-2
     ✓ success-3
     ✓ success-4
     ✓ success-5
     ✓ short unsafe (38ms)
     ✓ long unsafe
  getDeltaCash
     ✓ 0 -> +5
     ✓ 0 -> -5
  safePosition
     ✓ init
     ✓ short, infinite max position2, choose max position1
     ✓ short, choose max position1 (38ms)
     ✓ short, choose max position2
     ✓ long, choose max position3 (39ms)
     ✓ zero index price
  trade - success
     ✓ open 0 -> -141.421, near pos2 limit (45ms)
     ✓ open 0 -> -0.1, effected by spread (42ms)
     ✓ open -10 -> -141.067, near pos2 limit (43ms)
     ✓ open -10 -> -10.1, effected by spread
     ✓ open 0 -> 100, near pos2 limit (38ms)
     ✓ open 0 -> 0.1, effected by spread (42ms)
     ✓ open 10 -> 100, near pos2 limit (42ms)
     ✓ open 10 -> 10.1, effected by spread (42ms)
     ✓ close -10 -> -9, normal
     ✓ open -10 -> -9.9, effected by spread (38ms)
     ✓ close -10 -> 0, to zero
     ✓ close 10 -> 9, normal
     ✓ close 10 -> 9.9, effected by spread
     ✓ close 10 -> 0
     ✓ close unsafe -10 -> -9, normal
     ✓ close unsafe -10 -> -9.9, small
     ✓ close unsafe 10 -> 9, normal
     ✓ close unsafe 10 -> 9, small
     ✓ close negative price, clip to index*(1-discount)
     ✓ open 0 -> -141.422, partialFill
     ✓ open -10 -> -141.068, pos2 too large, partialFill (38ms)
     ✓ open -10 already unsafe, partialFill
     ✓ open 0 -> 100.001, partialFill
     ✓ open 10 -> 100.001, partialFill (41ms)
     ✓ open 10 already unsafe, partialFill
  trade - fail
     ✓ emergency
     ✓ zero trade amount
     ✓ poolMargin = 0
     ✓ open 0 -> -141.422, pos2 too large (45ms)
     ✓ open -10 -> -141.068, pos2 too large
     ✓ open -10 already unsafe
     ✓ open 0 -> 100.001
     ✓ open 10 -> 100.001
     ✓ open 10 already unsafe
  mint share
     ✓ init
     ✓ before safe, after safe
     ✓ short, before unsafe, after unsafe
     ✓ short, before unsafe, after safe
     ✓ long, before unsafe, after unsafe
     ✓ long, before unsafe, after safe
     ✓ poolMargin = 0 && totalShare != 0
  redeem share
     ✓ poolMargin = 0
     ✓ no position
```

```
      ✓ no position, remove all
      ✓ short
      ✓ long
      ✓ state != NORMAL
      ✓ short, before unsafe
      ✓ long, before unsafe
      ✓ short, after unsafe
      ✓ long, after unsafe
      ✓ long, after negative price
      ✓ long, after exceed leverage
      ✓ zero index
      ✓ zero supply of share token

Order
   ✓ normal

Broker
   ✓ broker (302ms)
   ✓ broker - cancel (210ms)
   ✓ broker - cancel by another signer (236ms)
   ✓ broker - fee (361ms)

LiquidityPool
   ✓ versionControl (171ms)
   ✓ createLiquidityPoolWith (307ms)
   ✓ tracer (304ms)
   ✓ tracer - 2 (586ms)

Funding
  updateFundingState
     ✓ state != NORMAL (259ms)
     ✓ init (259ms)
     ✓ current time = last funding time (247ms)
     ✓ normal (265ms)
     ✓ not sync (253ms)
     ✓ two sync (258ms)
  updateFundingRate
     ✓ state != NORMAL (237ms)
     ✓ init (242ms)
     ✓ unsafe (250ms)
     ✓ normal (246ms)
     ✓ exceed limit (242ms)
     ✓ margin < 0 (224ms)

Getter
   ✓ main (1043ms)

Governance
   ✓ checkIn (43ms)
   ✓ operatorship (181ms)
   ✓ forceToSetEmergencyState (126ms)
   ✓ setOracle (82ms)
   ✓ setEmergencyState - 1 (141ms)
   ✓ setEmergencyState - 2 (152ms)
   ✓ setPerpetualBaseParameter (50ms)
   ✓ setPerpetualBaseParameter - exception (209ms)
   ✓ updatePerpetualRiskParameter (38ms)
   ✓ setPerpetualRiskParameter (67ms)
   ✓ updatePerpetualRiskParameter - exception (108ms)
   ✓ updatePerpetualRiskParameter (75ms)
   ✓ setLiquidityPoolParameter (40ms)

GovernorAlpha.test
   ✓ exceptions (39ms)
   ✓ validateProposer (155ms)
   ✓ params (46ms)
   ✓ quorum (40ms)
   ✓ quorum critical (116ms)
   ✓ quorum critical - mixed (41ms)
   ✓ quorum critical - mixed (45ms)
   ✓ pass (467ms)
   ✓ rejected (518ms)
   ✓ lock (791ms)
   ✓ lock - 2 (528ms)
   ✓ lock - 3 (375ms)
   ✓ lock - 3 (871ms)
   ✓ lock - 4 (805ms)

integration
   ✓ main (2615ms)
   ✓ main - eth (1404ms)
   ✓ main - 6 decimals (2508ms)
   ✓ main (1797ms)
   ✓ liquidate (1096ms)
   ✓ accesscontrol (1042ms)

integration2
   ✓ normal case (1184ms)
   ✓ deposit more than balance
   ✓ deposit when not NORMAL
   ✓ add liquidity more than balance (38ms)
   ✓ add liquidity when not running
   ✓ trade when not authorized (127ms)
   ✓ trade when market closed (138ms)
   ✓ trade when market terminated (134ms)
   ✓ trade when invalid close-only amount (240ms)
   ✓ trade when invalid limit price (188ms)
   ✓ trade when trader unsafe (339ms)

LibMath
  mostSignificantBit
     ✓ normal
  sqrt
     ✓ small
     ✓ normal

LibSafeMathExt
  mul
     ✓ uint256 half up
     ✓ int256 half up
     ✓ int256 ceil
     ✓ int256 floor
  div
     ✓ uint256 half up
     ✓ int256 half up
     ✓ int256 ceil
     ✓ int256 floor
  frac
     ✓ uint256 half up
     ✓ int256 half up
     ✓ int256 ceil
     ✓ int256 down
  others
     ✓ uint256 max
     ✓ uint256 min
     ✓ int256 max
     ✓ int256 min
     ✓ abs
     ✓ neg

Liquidate
  basic
     ✓ liquidateByAMM (184ms)
     ✓ liquidateByAMM - bankrupt (138ms)
     ✓ liquidateByAMM - vault fee (186ms)
     ✓ liquidateByAMM - vault fee / bankrupt (147ms)
     ✓ liquidateByAMM - vault fee / not bankrupt (149ms)

LiquidityPool
  2 liquidityPool group
     ✓ updatePrice (154ms)
     ✓ getAvailablePoolCash (134ms)
     ✓ rebalance (304ms)
     ✓ rebalance - 2 (184ms)
     ✓ isAMMMaintenanceMarginSafe (219ms)
  operator
     ✓ transferOperator (135ms)
  trader
     ✓ donateInsuranceFund (93ms)
     ✓ deposit (69ms)
     ✓ withdraw (98ms)
     ✓ clear (316ms)
     ✓ clear - 2 (384ms)
     ✓ clear - 3 (398ms)
     ✓ settle (385ms)

LiquidityPool2
  liquidity
     ✓ init (184ms)
     ✓ before safe, after safe (182ms)
     ✓ short, before unsafe, after unsafe (173ms)
     ✓ short, before unsafe, after safe (182ms)
     ✓ long, before unsafe, after unsafe (179ms)
     ✓ long, before unsafe, after safe (180ms)
     ✓ invalid margin to add (112ms)
     ✓ poolMargin = 0 && totalShare != 0 (136ms)
```

```
      ✓ donate (70ms)
    remove liquidity
      ✓ poolMargin = 0 (181ms)
      ✓ no position (191ms)
      ✓ no position, remove all (200ms)
      ✓ short (196ms)
      ✓ long (197ms)
      ✓ zero share to remove (170ms)
      ✓ insufficient share balance (170ms)
      ✓ short, before unsafe (178ms)
      ✓ long, before unsafe (176ms)
      ✓ short, after unsafe (178ms)
      ✓ long, after unsafe (194ms)
      ✓ long, after negative price (179ms)
      ✓ long, after exceed leverage (185ms)

  LiquidityPool3
     ✓ createPerpetual - address (98ms)
     ✓ createPerpetual - fastCreation disable (164ms)
     ✓ createPerpetual - fastCreation enabled (236ms)

  LoopTest
     ✓ main (8495ms)

  Integration
     ✓ mint / redeem (62ms)
     ✓ lock

  MarginModule
    Getters
      ✓ +getInitialMargin (53ms)
      ✓ -getInitialMargin (56ms)
      ✓ +getMaintenanceMargin (54ms)
      ✓ -getMaintenanceMargin (55ms)
      ✓ +margin (61ms)
      ✓ -margin (57ms)
      ✓ +position (49ms)
      ✓ -position (44ms)
      ✓ getAvailableCash + funding (45ms)
      ✓ getAvailableCash - funding 1 (47ms)
      ✓ getAvailableCash - funding 2 (45ms)
      ✓ +isInitialMarginSafe yes (59ms)
      ✓ -isInitialMarginSafe yes (57ms)
      ✓ +isInitialMarginSafe no (54ms)
      ✓ -isInitialMarginSafe no (50ms)
      ✓ +isMaintenanceMarginSafe yes (52ms)
      ✓ -isMaintenanceMarginSafe yes (53ms)
      ✓ +isMaintenanceMarginSafe no (56ms)
      ✓ -isMaintenanceMarginSafe no (52ms)
      ✓ isEmptyAccount (55ms)
      ✓ isEmptyAccount - 1 (54ms)
      ✓ isEmptyAccount - 2 (56ms)
    Setters
      ✓ setMarginAccount (119ms)

  Minging
     ✓ notifyRewardAmount (283ms)
     ✓ setRewardRate (54ms)
     ✓ earned (250ms)
     ✓ rewardPerToken (203ms)
     ✓ rewardPerToken - 2 (291ms)
     ✓ rewardPerToken - reward tuncation (590ms)

  OracleRouter
     ✓ hash
     ✓ normal
     ✓ closed / terminated (73ms)

  Order
     ✓ signature
     ✓ validateOrder (92ms)
     ✓ decompress
     ✓ signer

  Perpetual
     ✓ getMarkPrice && getIndexPrice (76ms)
     ✓ getRebalanceMargin (131ms)
     ✓ setNormalState (80ms)
     ✓ setEmergencyState (82ms)
     ✓ setClearedState (78ms)
     ✓ donateInsuranceFund (49ms)
     ✓ deposit (55ms)
     ✓ withdraw (99ms)
     ✓ withdraw - market closed (86ms)
     ✓ clear (160ms)
     ✓ clear - 2 (284ms)
     ✓ getNextActiveAccount (114ms)
     ✓ settle (296ms)
     ✓ updateInsuranceFund (90ms)

  Perpetual2
    erc20
      ✓ donateInsuranceFund (88ms)
      ✓ deposit (119ms)
      ✓ withdraw (191ms)
      ✓ trade - 1 (250ms)
      ✓ trade - 2 (343ms)
      ✓ settle (388ms)

  PoolCreator
     ✓ main (557ms)
     ✓ implementations (1108ms)

  Reader
     ✓ getAccountStorage
     ✓ getLiquidityPoolStorage (90ms)
     ✓ zero price (116ms)

  TradeModule1
    basic
      ✓ getFees (90ms)
      ✓ getFees - rebate (150ms)
      ✓ getFees - open (83ms)
      ✓ validatePrice
     postTrade
       ✓ hasOpenedPosition (41ms)
       ✓ postTrade - 1 (86ms)
       ✓ postTrade - 2 (72ms)
       ✓ postTrade - 3 (73ms)
     trade
       ✓ sell (149ms)
       ✓ buy without cross 0 (145ms)
       ✓ buy cross 0 (157ms)

  TradeModule3
    basic
      ✓ regular (174ms)
      ✓ close (281ms)
      ✓ close - but no fee (446ms)
      ✓ market (199ms)

  upgrade
     ✓ main (752ms)

  normal
     ✓ 1 oracle, vanilla (77ms)
     ✓ 1 oracle, inverse (67ms)
     ✓ 2 oracles, vanilla (72ms)
     ✓ 2 oracles, inverse (74ms)
     ✓ 3 oracles (75ms)


  293 passing (2m)
```

Code Coverage

Two test files (`SymbolService.test.ts` and `Trade2.test.ts`) were skipped in order to compute test coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	92.44	69.05	90.24	92.74	
Getter.sol	80	100	62.5	80	… 308,313,314
Governance.sol	100	87.5	100	100	
LibraryEvents.sol	100	100	100	100	
LiquidityPool.sol	85.71	62.5	85.71	85.71	142,143
Perpetual.sol	100	64.58	100	100	
Storage.sol	100	75	100	100	
Type.sol	100	100	100	100	
contracts/broker/	80.33	70.83	90	79.03	
Broker.sol	80.33	70.83	90	79.03	… 186,188,189
contracts/factory/	94.81	60.26	96.15	95.17	
AccessControl.sol	100	100	100	100	
PoolCreator.sol	100	50	100	100	
Tracer.sol	93.75	58.33	92.86	93.94	140,181
UpgradeableProxy.sol	100	50	100	100	
Variables.sol	73.68	28.57	87.5	73.68	51,52,53,55,56
VersionControl.sol	100	78.57	100	100	
contracts/libraries/	87.42	80.3	87.23	87.73	
BitwiseMath.sol	100	100	100	100	
Constant.sol	100	100	100	100	
EnumerableMapExt.sol	46.15	37.5	57.14	44.44	… 124,126,127
Math.sol	100	100	100	100	
OrderData.sol	91.3	50	81.82	92	93,102
SafeMathExt.sol	100	100	100	100	
Signature.sol	77.78	25	100	88.89	37
Utils.sol	92.59	78.57	83.33	92.59	50,88
Validator.sol	100	100	100	100	
contracts/module/	98.04	82.84	98.32	98.05	
AMMModule.sol	99.28	88.89	100	99.29	313
CollateralModule.sol	100	72.73	100	100	
LiquidityPoolModule.sol	95.88	73	95.12	95.94	… 759,760,806
MarginAccountModule.sol	100	100	100	100	
OrderModule.sol	80.95	67.65	100	80	42,48,117,123
PerpetualModule.sol	100	92.06	100	100	
TradeModule.sol	99.07	84	100	99.07	321
contracts/oracle/router/	100	85	100	100	
OracleRouter.sol	100	87.5	100	100	
OracleRouterCreator.sol	100	75	100	100	
contracts/symbolService/	62.5	29.17	70	63.41	
SymbolService.sol	62.5	29.17	70	63.41	… 162,166,167
All files	**93.93**	**76.28**	**93.77**	**93.97**	

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

64f4535176087d4b3ef273ea72dc3047ac94ccde2cb3c5a9bde782cdc0c77afa ./contracts/Getter.sol

4fa8caa504f8a6b5c6927ab22da4a80473406cca6a376cfe46c40ebe5a100626 ./contracts/Governance.sol

68f070a8c0f65372597b01201c96c52587a91b1b45e3e60e5a843960f2eeef15 ./contracts/LibraryEvents.sol

a748f11e1f2762170ae1ab49961655f1f9390939541477799e4fa1bdb1cb1434 ./contracts/LiquidityPool.sol

7dd3c87a3ca64e8b1511cf506e95828e4ba757dffed77a1ac828caba246d1c5d ./contracts/Perpetual.sol

59e1afc580c1b3195eb1b85f3997f54b036ddb2f6957b00ad1a09820c90d79f6 ./contracts/Storage.sol

1dabd55f5aab85aed37bbb90e63c350e2db1eeaaab532b521e055790bb723b36 ./contracts/Type.sol

6f9e964687cba1b7cc9bb6b325e46d89093f2699f659390effc069a14571277d ./contracts/thirdparty/weth/WETH9.sol

dd1db3483c094444526a35130f72b05fe046d618c72055f7ba4ea634e4f53d0c ./contracts/thirdparty/cloneFactory/CloneFactory.sol

a28fcbea41d41bab3f68baa89e9576ad2b5b7c063bf8aee3f6b0426e157fa1a3 ./contracts/test/BlackHole.sol

955ab27daf3227b6dcb7c976d3e47ecee5c4cb477bbb51c0b1417cb6b13ee811 ./contracts/test/CustomERC20.sol

5d9299fdc8aa00a1a6e8347d0ff9ec5f36b20d13a7e7a426f38055cad9378b95 ./contracts/test/LiquidityPoolAdmin.sol

eb9d23a0db1fbced0d761c9f09dfbf70c31039439fb639f9aeca15e2a6654099 ./contracts/test/MockAMMModule.sol

56000fdb9ce677e2bffa291d51d7d017e5516d29028b99562fac2ac444b0baae ./contracts/test/MockLiquidityPool.sol

1493a543f8f978acc9b239dd68262a425bd17ed701e8a0765db828846f89ab15 ./contracts/test/TestAccessControl.sol

6629957e35b3b48e5b35c869897971aba0cca8b7df323615296af58f967ecdc1 ./contracts/test/TestAMM.sol

22ae833f11ab2a2e331edd271e9845e165aa43ff7f03c02c23fff2a173c9daeb ./contracts/test/TestBitwiseMath.sol

8c967278f71d41e234d304755e2995b21ddb5a9eb9cf654451c45f193e1cdff5 ./contracts/test/TestCalc.sol

fedb87a8bbf6568bc90e22d9526e915376f462c4bfdcb691506e6b211c694beb ./contracts/test/TestGovernance.sol

9418d916ce850a2608da2055e1193ae22940031557b2708cd8d2bfd69d939bc6 ./contracts/test/TestGovernanceToken.sol

3072d001798d5af9d3ee407adfd77950d906ca36d28ec4ab795ffd76b839cc9b ./contracts/test/TestGovernor.sol

89e5bbddd7c802552cf3f50bf252721c6f0a6b2b731905c63d778c998c53b3d1 ./contracts/test/TestHelper.sol

a1205887ae50e80bcdf7ce912e35a0151776ecc6cfda3d82a6b93c619906a050 ./contracts/test/TestLibMath.sol

a300c74057ce1ca519d87a6b4cd42744061b0a2ff65565133cef5aafedc91ec2 ./contracts/test/TestLibSafeMathExt.sol

2ad602aa8a065eb90bec80bb14688aa89631198f90fa8ed7099c93a56c16b35e ./contracts/test/TestLiquidityPool.sol

f113621580a0704a0423f194155a52ad54d347257829a981ad28d36c37d38354 ./contracts/test/TestMarginAccount.sol

1091a01b1b77b28f79914225fa522ad42d24d235e5b9e5519d7ba5eed08f1efd ./contracts/test/TestOrder.sol

788f225952b1b9c574ddf77729d03613502e95ab8dca238317a7249c3f4163ed ./contracts/test/TestPerpetual.sol

bafc562f86f7bf84c2102c1b18ade56e643db7e8789067ea18384873c676a302 ./contracts/test/TestShareToken.sol

71b810d61e80773a52fd46846ba9bd5a9d0170c80589751ce1ab03dc49f460e7 ./contracts/test/TestSymbolService.sol

c4aea62ddb87bd325ba157d3a7bc765a8b5fa71896c09b44b16e895c5771f145 ./contracts/test/TestTracer.sol

d00994efaf8de635d046e55e43bc407ac7a830809f89dfbba004dbf6541b6f96 ./contracts/test/TestTrade.sol

42955e5b2d20f040a0d36be877c792d49576c09aeab4483800b56a1d937aa5c2 ./contracts/symbolService/SymbolService.sol

4fa040fa2180c5738de1e878204734364e827e9041ab9a557b2595f2053f1608 ./contracts/reader/Reader.sol

610ba5ad374ee7aa3ca93bd95c9dd63a5bb1de3a5163dc341cfe5329899067ef ./contracts/oracle/router/OracleRouter.sol

1d8368419e5fc39a85425b33df5372b047d103bd104a6e0ea446ba80c4293f0c ./contracts/oracle/router/OracleRouterCreator.sol

b9f6ad2a8c90bd6ec75b57f2ed01a407eb171fe77eb42d39aff85f115ff97e9e ./contracts/oracle/mock/OracleWrapper.sol

167ea834b5cca7a40c279610868c0951edc8745726174a2ff2f53fcc7853971e ./contracts/module/AMMModule.sol

095971f935d26454e9f5a4b7cc9324f0065d02196448d1981511459a68ced964 ./contracts/module/CollateralModule.sol

1403709d76ddf5e6eb4f668b94d4ef41657210def60f180d5598948e3d6dfe44 ./contracts/module/LiquidityPoolModule.sol

3d1f465dc3947d29c7ef61fc449cc0b35f72bcb7f218b8c9d430635c67a88f2f ./contracts/module/MarginAccountModule.sol

a0815ac82800038e3294cb6ecc3d9d7b13f6bb4e28ccbb4a127d8192ac1344f4 ./contracts/module/OrderModule.sol

b2b6c0245bcc129623c910d0eccd48502e325915089803c97ad64b83e83355f9 ./contracts/module/PerpetualModule.sol

e4ecc71be376ba40b27d097daa7740f158c93d2b88c4b3ad671558aca2ab6b34 ./contracts/module/TradeModule.sol

e311c0e6d5c0eeca7825a30f56f2be6083a39a1b021ae7bfb7a8dfbc990bded0 ./contracts/libraries/BitwiseMath.sol

0fe1f049d39a7797875c39e6ebfa984a8072dbb06d004386559bd2c6be2cb4ca ./contracts/libraries/Constant.sol

49ba9d35990e008da327e76b9f672ff921499451fdd5909d3aa8981a0d15e326 ./contracts/libraries/EnumerableMapExt.sol

610508798201ea947c3528df16646a86558f345ef8c443a9a9550d7e4b06d61a ./contracts/libraries/Math.sol

c72d915e71a25585fe0825a13f80f5d8237877dd813f36188df96ed28a2ea945 ./contracts/libraries/OrderData.sol

2547f52810a3b62c7477ae3ad558e3ab905162fc758085fb354fc36abebb9677 ./contracts/libraries/SafeMathExt.sol

ff4d11c4e8e3570f0b9cc0c61aa4fc3de716db532d68e7286cb8037eec3bb260 ./contracts/libraries/Signature.sol

2548253acedaec74c467a66c837d64559ca50cb96cc897d6ac1374ed1e8ca0a4 ./contracts/libraries/Utils.sol

4f3a0fa95822d3b8a922cbc0973e38526ac40467b814c2eeaa0142b521dff206 ./contracts/libraries/Validator.sol

4e442d768bb70e9bec90358680e2877fadfb164d3b268a31fa458abba49d94d0 ./contracts/interface/IAccessControl.sol

338c9d5977c5ca46ce6cbd013ebcd9d5d3210298aa48c7898dcc51d1effa4a04 ./contracts/interface/IDecimals.sol

0c6461d804fed4563495d39a0cda324143c06a9c51262fded322e59ee74c1df1 ./contracts/interface/IGovernor.sol

ba112f0e2b141f9b45d93f8954b701b74e0740572ed65f3958d69c78ae24e1b2 ./contracts/interface/ILiquidityPool.sol

bd523b660a76bf68ca16a6e8be5fcd994f960b128a10401f3a7b07ea9dd01be3 ./contracts/interface/ILiquidityPoolGovernance.sol

aef939beebb43522e972bf8bc7fcb39e620687a8af9b4148261f17ff86bcedc2 ./contracts/interface/IOracle.sol

87a1a58f141ff15ebf293be6b277910c9d045a1189497e549d26fab057aa185a ./contracts/interface/IPoolCreator.sol

39fd25567d0d3704764982fcb3b9ce01df0ba76f4fcba1b5ae784b063c7735a1 ./contracts/interface/IRelayRecipient.sol

97a5adc825ea35070ea429d41bc99a9f68eb1e6c1ab5abca872b59ab666da6cd ./contracts/interface/IShareToken.sol

75dafa1d791e070eea91e44036277ef0c77111a3bc0c52bfc2d6f635273765d8 ./contracts/interface/ISymbolService.sol

3e2bd64cddc1e03c4fa096c6b699613ebaaa9e27b2f3aa3f5b4aa3ed5e701d84 ./contracts/interface/IWETH.sol

585b83625b8ca6cf6998513e33433b36b8a6bf344fcfdd68b00b0ec03f57706d ./contracts/governance/GovernorAlpha.sol

4779284013e8b4b30bde7c26bf4431725ad07e6cb0d8eb8df8d6026ba0e87801 ./contracts/governance/LpGovernor.sol

9be8065476fe71c17d1aceb4189119ce7af54b252550a5bf9f558e2037f69a17 ./contracts/governance/RewardDistribution.sol

ed323d49a088924d6d257e3aa7a3f661217267b352ed4a87a5ae71f6fb025d59 ./contracts/factory/AccessControl.sol

ae302a02814b2d794960ac7d11664c53ed213823151fb684ca4b09b381188de1 ./contracts/factory/PoolCreator.sol

c6dc940fd7f5b38f35095f91b8b106bd33e613fcf6309678fd27247da1620ea2 ./contracts/factory/Tracer.sol

f7a93746d911f7387dd728edde53a178a8bb7f52cbb349f6510ffdcdb796fdb8 ./contracts/factory/UpgradeableProxy.sol

8705a5c0a613c27e1046ebd5a85902cfefb762fd73d5a4a1c8c6028d755c9980 ./contracts/factory/Variables.sol

0286268535b81aaa1000334261d8a3c93e40d264474df56a40b8000df46496bd ./contracts/factory/VersionControl.sol

adeb914ef49be8d785b26ca15a8c97097f3626352b51493eaef3c9368b09ae3d ./contracts/broker/Broker.sol

Tests

77ff35c8a9e9e5c93fc63d0ffd5834ef288a70b5bee4408d35c290d776c68d33 ./test/AccessControl.test.ts

40e017a61d10bab4a29bab01cfaca65ef90072b3a86ad023ee5f15b553e8a12a ./test/AMM.test.ts

616cce5677adc8a566b126e4e5ec06d61461d80282edb09ab73df7ce27171b09 ./test/BitwiseMath.test.ts

b577d138805a1d6a06471ac8e572a135d8f4adb1cfe1c6fea51fe023044c9358 ./test/Broker.test.ts

70eac95f8cdfc4d2d0b8316194991a57490ee6b734d18338d907c4587ac2f4c3 ./test/Creator.test.ts

8a0966ccaf7399d112b7f34d301bd29150145cbb7c842f886b7d87f062587dbc ./test/Funding.test.ts

4a0c4ceaae8e49448ee38aa1cc8414baea9dcd8cd57fbf2176921983f99bc683 ./test/Getter.test.ts

f8032b3b4dbefe2d8d810980990cad071fbd3620a775fd3c6133156684653ff6 ./test/Governance.test.ts

edf9a86eccd2bbb96c82c76424f22ada500a968ebfb4008cac58652197508afe ./test/GovernorAlpha.test.ts

e022a1e6e86cedec3e4242470e088c60cd4bb2774e197957164a0f9b081d8e5c ./test/helper.ts

30b7a0768bc3f2cb989b3e053cce346de19132812bbd832da8fa97f90676a095 ./test/Integration.test.ts

d9adc9a99411aeba7ada64545cf3e385606b9b41f750ba59c1e57d85de3370e4 ./test/Integration2.test.ts

ac4e9abed731cecfba074f87bb90e2879502ae2c9326049aaa082d63cb480679 ./test/LibMath.test.ts

97bb0f310456dcc4793edd268a2f9345b1d5b97c1ba4661708fdd0883e8c53e4 ./test/LibSafeMathExt.test.ts

c8788d96db48695cdd771e2f6ad547d4a3b344ffdebd4a61d2c93eb88ef458fb ./test/Liquidate.test.ts

efc0caec0db75e9591228d46636eccbe63530d6a092c14b6aed7b1bb9c015970 ./test/LiquidityPool.test.ts

450821eed6aeab8ae4e78772f4ca008904bfd01524249fb9b3f5b6b135160c33 ./test/LiquidityPool2.test.ts

547536ac5e2f33b79e1fa1bf5ead6a788a7f308a23c17e0333a5a337fde08bf2 ./test/LiquidityPool3.test.ts

5aae586992d58b4d5667e66b59d94dccf9ec2f6522781f69806da496f18b7673 ./test/LoopTest.test.ts

33e85b83bb97bcb00ac4e4c2edb272204af20860c85c7fba28d419bbda1f77eb ./test/LpGovernor.test.ts

524e98b30d47cb1751e7ac7fd2b24bd5afe620feba5478a94936a6349decc9a1 ./test/MarginAccount.test.ts

a98b42103afe7d42bfef6891f42852fab404cf1c07df0fe396aa8ea5456d662a ./test/Mining.test.ts

02affcc33912e121cfc0e8952489c749640db10758f70809898e9cf16986f271 ./test/OracleRouter.test.ts

b9a7ed6f4f266910e74a091e8ecfda0998d3106bfc48241d6fda309a982e4194 ./test/Order.test.ts

2536897e60420388b75828c5a9f67be2385704230b169f9b7a04c9155bf5e93e ./test/Perpetual.test.ts

809948936794046e67da13169f08de33252f2543bfb4b62fb62a4a3733d70c00 ./test/Perpetual2.test.ts

2572b0a13ed2891b511641157277e4ec2b2b38231f84faa9a3c945c400346c1e ./test/PoolCreator.test.ts

34ceba549fd9abc2395869a684fb8db7f7a0741276a9e1f7d940dbeaad4ef920 ./test/Reader.test.ts

7b2b09f4855d075038038bc0a50987a49e3809957e1e489ffd1334e786016f5e ./test/SymbolService.test.ts

340c2277c7aeeeb834c35363c1b4f5634a6b9115f5a67593acd932a046e0efbb ./test/Trade1.test.ts

afddc32784b50dc2e0067a426429142e301beb8b4e1300559bd9a1bc47b87fda ./test/Trade2.test.ts

49829f345bb3a6f778a9371a8e76dec4800a643d5dcbbceec5d45586710a2c8b ./test/Trade3.test.ts

ed4fedfae9964ce75484ac907d74838cf1dda486fd944d2697298d40a6dc2466 ./test/Upgrade.test.ts

Changelog

- 2021-04-12 - Initial report [e4bfcf1]
- 2021-04-20 - Revised report [50fb550]

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.