Quantstamp Security Assessment Certificate

September 3rd 2021 — Quantstamp Verified

Ondo Finance V2

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

Executive Summary

Type	Yield Aggregator Strategy
Auditors	Leonardo Passos, Senior Research Engineer Fayçal Lalidji, Security Auditor
Timeline	2021-08-23 through 2021-09-03
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Low
Test Quality	Low
Source Code	

Repository	Commit
protocol-dev	610a629

Total Issues	15	(0 Resolved)
High Risk Issues	0	(0 Resolved)
Medium Risk Issues	4	(0 Resolved)
Low Risk Issues	6	(0 Resolved)
Informational Risk Issues	4	(0 Resolved)
Undetermined Risk Issues	1	(0 Resolved)

15 Unresolved
0 Acknowledged
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

Through reviewing the code, we found **16 potential issues** of various levels of severity. We recommend addressing the findings prior to deploying the smart contracts to the main network.

ID	Description	Severity	Status
QSP-1	Mid-Term Lp Deposits Allow For Risk-Free Profits	^ Medium	Unresolved
QSP-2	Use of Magic Numbers Dependent on External Protocol Fees	^ Medium	Unresolved
QSP-3	Possible Truncation While Calculating Vault Deposits Shares	^ Medium	Unresolved
QSP-4	Possible Incorrect Path Reward Update	^ Medium	Unresolved
QSP-5	Privileged Roles and Ownership	˅ Low	Unresolved
QSP-6	Potential Division by an Extremely Small Value	˅ Low	Unresolved
QSP-7	Harvest Is Not Affected by Pausing	˅ Low	Unresolved
QSP-8	`updateRewardPath` Allows Circular Paths	˅ Low	Unresolved
QSP-9	Mising `isContract` Validation	˅ Low	Unresolved
QSP-10	Lp Removal Does Not Validate Target Address	˅ Low	Unresolved
QSP-11	Clone and Own (1)	○ Informational	Unresolved
QSP-12	Clone and Own (2)	○ Informational	Unresolved
QSP-13	Vault Shares Update Does Not Follow the Same Pattern	○ Informational	Unresolved
QSP-14	Possible Transaction Revert Due to Sandwich Attack	○ Informational	Unresolved
QSP-15	Incorrect Path Validation Logic	? Undetermined	Unresolved

Quantstamp Audit Breakdown

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

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 Mid-Term Lp Deposits Allow For Risk-Free Profits

Severity: Medium Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: An attacker can sandwich-attack every time `SushiStakingV2Strategy.harvest(...)` is called by mid-term depositing LP tokens before the harvest (receiving tranche tokens). The attacker proceeds to withdraw the tranche share tokens for an increased amount of LP tokens.

Exploit Scenario: 1. Attacker observes a pending harvest function by the strategist

1. Attacker deposits LP tokens using `AllPairVault.depositLp(...)`. The latter invokes `Strategy.addLp(...)`, which gets being resolved to `SushiStakingV2Strategy.addLp(...)`. The attacker receives tranche tokens representing a fair share of the current LP tokens of the strategy
2. Harvest happens, increasing the total LP tokens of the strategy
3. Attacker redeems their tranche tokens from step (2) by calling `AllPairVault.withdrawLp(...)` (calling `Strategy.removeLp(...)`) and receives more LP tokens than initially due to the increase in (3).

Recommendation: We recommend compounding the rewards before assigning the shares to a possible attacker.

QSP-2 Use of Magic Numbers Dependent on External Protocol Fees

Severity: Medium Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: The constants used in the `calculateSwapInAmount()` function depend on the fees charged by Uniswap and SushiSwap. Changes in fees by these protocols would lead to incorrect calculations.

Recommendation: Parameterize the contract s.t. one could make updates to the fee values as needed.

QSP-3 Possible Truncation While Calculating Vault Deposits Shares

Severity: Medium Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: The implemented staking mechanism in `SushiStakingV2Strategy` is highly optimized and helps avoid any extra computation. However, setting the values of the initial shares for a specific pool is important and should be analyzed carefully.
The algorithm is designed in a way to compute smaller shares every time that the LP rewards get higher, meaning that depositing the same LP tokens values for a specific vault will result in giving different shares values (while the reward is accruing).
However, if the initial deposit is low enough and if the LP rewards are significantly greater than the initial LP deposit, new deposits to the vault will see their computed share being truncated.
In a worst-case scenario, if for any reason higher LP rewards can be introduced by an attacker, the next deposits will be truncated following the value of rewards introduced. The attacker will profit by getting higher returns.

Recommendation: This issue can be solved by multiplying the initial amount by a constant multiplier in `depositIntoChef` L496 and L498. The value of the multiplier should be determined following the smallest possible investment, for example, if the smallest possible investment is 1 Wei, we would recommend a multiplier of `10**18`.

QSP-4 Possible Incorrect Path Reward Update

Severity: Medium Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: `updateRewardPath` requirement in L282 seems incorrect since in the case if reward token and end token are similar the required length is set to be higher than 1, meaning that resetting a path length equal to 1 is not allowed.

Recommendation: The team should confirm if this behavior was intended since the requirement looks like it has to be changed to check if the length is equal to one when both reward and end token are equal.

QSP-5 Privileged Roles and Ownership

Severity: Low Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: Many operations in `SushiStakingV2Strategy` rely on a strategist role.
A vault's strategist is responsible for correctly executing most of the vault and strategy actions and needs to be trusted.
Even when assuming full trust in strategists, `SushiStakingV2Strategy` would be at risk if strategists get compromised.

Recommendation: Document the trust assumptions for the strategist role and consider using a multi-sig defence.

QSP-6 Potential Division by an Extremely Small Value

Severity: Low Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: The `SushiStakingV2Strategy.depositIntoChef(...)` calls `_compound()` before reaching the block in L497-500, in which the following calculation s performed:

```
uint256 shares = (_amount * poolData.totalShares) / poolData.totalLp
```

If `poolData.totalLp` is an extremely small value (for example, because of some dust generated from calling `_compound()`), this may lead to an extremely large value being stored in the `shares` variable. A similar issue is present in `withdrawFromChef(...)`.

Recommendation: Consider writing several test cases in which these code blocks are reached to ensure intended behavior.

QSP-7 Harvest Is Not Affected by Pausing

Severity: Low Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: To allow reacting to a potential hack, it is always a good practice to allow pausing a contract. However, `SushiStakingV2Strategy.harvest(...)` can still be called when the contract is paused. Note that the latter calls `_compound`, which is itself not affected by any pauses. Hence, one can invoke `harvest(...)` while the strategy is paused. That should not be allowed, especially in the case of reacting to a hack.

Recommendation: Add the `whenNotPaused` modifier to `SushiStakingV2Strategy.harvest(...)`.

QSP-8 `updateRewardPath` Allows Circular Paths

Severity: Low Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: The `SushiStakingV2Strategy.updateRewardPath(...)` function allows paths with lengths greater than one, yet, with the reward and end token being the same (circular). It is just wasteful to trade a token for itself.
At the code level (L278--281), this appears as follows:

```
L278. require(
L279.    rewardToken != endToken || _pathFromReward.length > 1, <===
L280.   "Invalid path"
);
```

For `_pathFromReward.length > 1` to be evaluated, `rewardToken` must be the same as `endToken`, in which case the length is already greater than one. Otherwise, if `rewardToken == endToken`, the length should be exactly one, which is not the condition being checked.

Recommendation: Change L278--281 to:

```
require(
   rewardToken != endToken || _pathFromReward.length == 1,
   "Invalid path"
);
```

QSP-9 Mising `isContract` Validation

Severity: Low Risk

Status: Unresolved

File(s) affected: `AlchemixUserReward.sol`, `SushiStakingV2Strategy.sol`

Description: The constructor of `SushiStakingV2Strategy` does not validate whether the input addresses are indeed contracts; same with `SushiStakingV2Strategy.addPool(...)` and `AlchemixUserReward.constructor(...)`. Hence, incorrect setups could lead the contract to misbehave.

Recommendation: As mitigation, add a check requiring that all provided addresses that should refer to a contract pass the `Address.isContract(...)` check. Alternatively, make sure your deployment scripts assert that the given addresses refer to the contract addresses expected by `SushiStakingV2Strategy.constructor(...)`.

QSP-10 Lp Removal Does Not Validate Target Address

Severity: Low Risk

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: The `SushiStakingV2Strategy.removeLp(...)` does not validate its `to` argument. If `address(0)`, the removal essentially burn funds.

Recommendation: Add a `require` statement to ensure the `to` address is not `0x0`.

QSP-11 Clone and Own (1)

Severity: Informational

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: Function `calculateSwapInAmount(...)` appears to have been cloned from Alpha Homora and Zapper, but the inline code comment does not indicate the commit hash and file from which it was taken. If buggy, faulty behavior may be permanently set in unless one is able to trace the code and corresponding fixes that might surface.

Recommendation: Link the cloned code with its source so one can trace potential bugs and patches, periodically pulling the latter whenever available.

QSP-12 Clone and Own (2)

Severity: *Informational*

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: Contract `SushiStakingV2Strategy` shares a lot of commonality with `SushiStrategyLP`, but duplicated code appears in both contracts. Not only this adds extra maintenance, but also can lead to incorrect bug fixing if fixes are done in one strategy, but not the other.

Recommendation: Refactor duplicated code in `SushiStakingV2Strategy` and `SushiStrategyLP` in a base contract that gets inherited by both.

QSP-13 Vault Shares Update Does Not Follow the Same Pattern

Severity: *Informational*

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: `depositIntoChef()` does not accumulate the vault shares but rewrites them, using the input value. This scheme is not used any of the other functions that update the shares. Even if `depositIntoChef` can be called only once, developers should be careful with future contract updates.

Recommendation: We recommend to accumulates its value rather than resetting it.

QSP-14 Possible Transaction Revert Due to Sandwich Attack

Severity: *Informational*

Status: Unresolved

Description: Not including the minimum out values and setting its value to zero when executing an external call to Sushiswap might result in a failing transaction due to bots not recognizing that a transaction may fail later on. This issue depends on how the attack bots are implemented.

QSP-15 Incorrect Path Validation Logic

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `SushiStakingV2Strategy.sol`

Description: The`rewardPaths` array should have two paths. It seems one path should end in `token0` and the other in `token1` (or vice-versa). However, the implementation in `SushiStakingV2Strategy.addPool(...)`, both paths could end in `token0` or `token1`. It is unclear if the latter situation is indeed desired; our understanding is that **only** the first case should be allowed.

Recommendation: Clarify this issue with better documentation in the code. If the current implementation is incorrect, then require that both paths have distinct ends, one ending in `token0` and the other in `token1`.

Automated Analyses

Slither

Slither did not report any significant issues.

Adherence to Best Practices

- In `SushiStakingV2Strategy.sol`, make sure to add descriptive messages for the require statements. For exaample, `require(_router != address(0), "Invalid address")` could be restated as `require(_router != address(0), "Invalid router address")`.
- `SushiStakingV2Strategy.sol` make use of magic constants whose purpose is unclear to readers. Please provide supporting documentation.
- Unused storage variable `SushiStakingV2Strategy.xSushi`. Consider removing it.
- Unused function `SushiStakingV2Strategy.getSushiPath(...)`. Consider removing it.
- Message in `SushiStakingV2Strategy.sol` at L245 is misleading, as it could be the case that both the first element in the first and second paths are sushi. In the latter case, although the first path would be sushi, the error message would report the contrary.
- Function `SushiStakingV2Strategy._compound(...)` is too large. Consider refactoring it.
- It seems `SushiStakingV2Strategy.updateRewardPath(...)` does not need to be marked `nonReentrant`, as it does not contain any external calls.

Test Results

Test Suite Results

```
npx hardhat test
Creating Typechain artifacts in directory typechain for target ethers-v5
Successfully generated Typechain artifacts!


  masterchefv2 dual incentive pool
    mcv2 vault lifecycle
       ✓ get assets (17270ms)
       ✓ create vaults (2589ms)
       ✓ deposit assets and fast-forward (9608ms)
       ✓ invest ALCX/ETH vault (39481ms)
       ✓ invest ETH/ALCX vault (8186ms)
       ✓ harvest (322ms)
       ✓ harvest (392ms)
       ✓ harvest (395ms)
       ✓ harvest (254ms)
       ✓ harvest (370ms)
       ✓ harvest (315ms)
       ✓ redeem ALCX/ETH vault (3159ms)
       ✓ redeem ETH/ALCX vault (659ms)

  Claim
     ✓ Create vault (2652ms)
     ✓ Get some DAI
     ✓ Deposit on both sides (11363ms)
     ✓ Move to invest state (4894ms)
     ✓ Try to claim DAI
     ✓ Try to claim ETH
     ✓ Redeem funds (2593ms)
     ✓ Withdraw all funds (74ms)

  Create2
     ✓ create Vault (1800ms)

  Performance fees
    delayed Vault
       ✓ setup vault fixture (3577ms)
       ✓ create vault (1394ms)
       ✓ can only deposit after start time (1345ms)
       ✓ setup fee collector and performance fee
       ✓ deposits after start time (77ms)
       ✓ invest and redeem (194ms)

  Ondo
    metadata
       ✓ has a name
       ✓ has a symbol
    balanceOf
       ✓ grants to initial account
    delegateBySig
       ✓ reverts if the signatory is invalid (528ms)
       ✓ reverts if the nonce is bad
       ✓ reverts if the signature has expired
       ✓ delegates on behalf of the signatory
    numCheckpoints
       ✓ returns the number of checkpoints for a delegate (80ms)
    Ondo transfer disabled
       ✓ transfer reverts when transfers are disabled
       ✓ account with transfer role can transfer
       ✓ transferAllowed works correctly

  Registry
     ✓ grant roles
     ✓ register contracts

  Rescue functions
    pause and rescue asset and LP tokens
       ✓ doesn't allow rescuing tokens outside of pause mode
       ✓ pauses and freezes Vault functions (74ms)
       ✓ rescue tokens (57ms)
       ✓ stops the pause and restores functionality

  test reverts
     ✓ reverts (1723ms)

  RolloverVault
    basic rollover
       ✓ create rollover (2640ms)
       ✓ create rollover from existing Vault (3795ms)
       ✓ deposit senior asset (157ms)
       ✓ deposit exceeds senior user cap (50ms)
       ✓ deposit junior asset (180ms)
       ✓ adds another Vault to rollover tip (1304ms)
       ✓ migrate (182ms)
       ✓ single deposit (54ms)
       ✓ deposit senior asset (184ms)
       ✓ deposit exceeds senior user cap (47ms)
       ✓ deposit junior asset (186ms)
       ✓ adds another Vault to rollover tip (1245ms)
       ✓ migrate (299ms)
       ✓ deposit senior asset (155ms)
       ✓ deposit exceeds senior user cap (44ms)
       ✓ deposit junior asset (159ms)
       ✓ adds another Vault to rollover tip (1326ms)
       ✓ migrate (278ms)
       ✓ single deposit (51ms)
       ✓ deposit senior asset (227ms)
       ✓ deposit exceeds senior user cap (59ms)
       ✓ deposit junior asset (178ms)
       ✓ adds another Vault to rollover tip (1532ms)
       ✓ migrate (306ms)
       ✓ deposit senior asset (253ms)
       ✓ deposit exceeds senior user cap (50ms)
       ✓ deposit junior asset (163ms)
       ✓ adds another Vault to rollover tip (1298ms)
       ✓ migrate (283ms)
       ✓ doesn't send additional excess on deposit after claiming
       ✓ withdraw (94ms)
       ✓ deposit senior asset (177ms)
       ✓ deposit exceeds senior user cap (42ms)
       ✓ deposit junior asset (161ms)
       ✓ adds another Vault to rollover tip (1330ms)
       ✓ migrate (296ms)

  StakingPools
     ✓ should set correct state variables (523ms)
     ✓ should allow emergency withdraw (197ms)
     ✓ should give out ondos only after farming time (1093ms)
     ✓ should not distribute ONDOs if no one deposit (602ms)
     ✓ should distribute ondos properly for each staker (1302ms)
     ✓ should give proper ONDOs allocation to each pool (835ms)
     ✓ should stop giving bonus ONDOs after the bonus period ends (1079ms)
     ✓ should track minimumOndoRequiredBalance properly (903ms)

  SushiStrategyLP
     ✓ pool add and update reverts (18253ms)
    batchCreate
       ✓ create Vault: sell senior for leveraged junior returns (1255ms)
       ✓ create Vault: sell all junior to partially cover senior (1297ms)
       ✓ create Vault: sell some junior to cover senior (1371ms)
    batchDeposit
       ✓ deposit senior asset: sell senior for leveraged junior returns (92ms)
       ✓ deposit junior asset: sell senior for leveraged junior returns (87ms)
       ✓ deposit senior asset: sell all junior to partially cover senior (90ms)
       ✓ deposit junior asset: sell all junior to partially cover senior (87ms)
       ✓ deposit senior asset: sell some junior to cover senior (88ms)
       ✓ deposit junior asset: sell some junior to cover senior (88ms)
    increase time
       ✓ increase time
    batchInvest
       ✓ invest assets: sell senior for leveraged junior returns (6703ms)
       ✓ invest assets: sell all junior to partially cover senior (1192ms)
       ✓ invest assets: sell some junior to cover senior (229ms)
    increase time
       ✓ increase time
    batchMidDeposit
       ✓ deposit LP tokens mid-duration with signer 4: sell senior for leveraged junior returns (1199ms)
       ✓ deposit LP tokens mid-duration with signer 4: sell all junior to partially cover senior (548ms)
       ✓ deposit LP tokens mid-duration with signer 4: sell some junior to cover senior (525ms)
    batchMidDeposit
       ✓ deposit LP tokens mid-duration with signer 5: sell senior for leveraged junior returns (517ms)
       ✓ deposit LP tokens mid-duration with signer 5: sell all junior to partially cover senior (522ms)
       ✓ deposit LP tokens mid-duration with signer 5: sell some junior to cover senior (526ms)
    increase time
       ✓ increase time
    batchHarvest
       ✓ harvest rewards: sell senior for leveraged junior returns (166ms)
       ✓ harvest rewards: sell all junior to partially cover senior (368ms)
       ✓ harvest rewards: sell some junior to cover senior (167ms)
    batchMidWithdraw
       ✓ withdraw LP mid-duration with signer 4: sell senior for leveraged junior returns (109ms)
       ✓ withdraw LP mid-duration with signer 4: sell all junior to partially cover senior (101ms)
       ✓ withdraw LP mid-duration with signer 4: sell some junior to cover senior (112ms)
    batchMidDeposit
       ✓ deposit LP tokens mid-duration with signer 6: sell senior for leveraged junior returns (528ms)
       ✓ deposit LP tokens mid-duration with signer 6: sell all junior to partially cover senior (528ms)
       ✓ deposit LP tokens mid-duration with signer 6: sell some junior to cover senior (510ms)
    batchHarvest
       ✓ harvest rewards: sell senior for leveraged junior returns (161ms)
       ✓ harvest rewards: sell all junior to partially cover senior (170ms)
       ✓ harvest rewards: sell some junior to cover senior (165ms)
    increase time
       ✓ increase time
    batchClaim
       ✓ claim tranche tokens: sell senior for leveraged junior returns (148ms)
       ✓ claim tranche tokens: sell all junior to partially cover senior (144ms)
       ✓ claim tranche tokens: sell some junior to cover senior (88ms)
    redeem and withdrawal: sell senior for leveraged junior returns
       ✓ redeem LP after fee accrual (329ms)
       ✓ withdraw received amounts (291ms)
    redeem and withdrawal: sell all junior to partially cover senior
       ✓ redeem LP after fee accrual (311ms)
       ✓ withdraw received amounts (281ms)
    redeem and withdrawal: sell some junior to cover
       ✓ redeem LP after fee accrual (473ms)
       ✓ withdraw received amounts (302ms)
    final sanity check
       ✓ pool totallp and totalshares is zero
    works with sushi as token in pair being farmed
      batchCreate
         ✓ create Vault: sell senior for leveraged junior returns (1244ms)
         ✓ create Vault: sell all junior to partially cover senior (1321ms)
         ✓ create Vault: sell some junior to cover senior (1134ms)
      batchDeposit
         ✓ deposit senior asset: sell senior for leveraged junior returns (4697ms)
         ✓ deposit junior asset: sell senior for leveraged junior returns (90ms)
         ✓ deposit senior asset: sell all junior to partially cover senior (60ms)
         ✓ deposit junior asset: sell all junior to partially cover senior (94ms)
         ✓ deposit senior asset: sell some junior to cover senior (62ms)
         ✓ deposit junior asset: sell some junior to cover senior (89ms)
      increase time
         ✓ increase time
      batchInvest
         ✓ invest assets: sell senior for leveraged junior returns (1488ms)
         ✓ invest assets: sell all junior to partially cover senior (215ms)
         ✓ invest assets: sell some junior to cover senior (338ms)
      increase time
         ✓ increase time
      batchMidDeposit
         ✓ deposit LP tokens mid-duration with signer 4: sell senior for leveraged junior returns (535ms)
         ✓ deposit LP tokens mid-duration with signer 4: sell all junior to partially cover senior (497ms)
         ✓ deposit LP tokens mid-duration with signer 4: sell some junior to cover senior (497ms)
      batchMidDeposit
         ✓ deposit LP tokens mid-duration with signer 5: sell senior for leveraged junior returns (504ms)
         ✓ deposit LP tokens mid-duration with signer 5: sell all junior to partially cover senior (536ms)
         ✓ deposit LP tokens mid-duration with signer 5: sell some junior to cover senior (521ms)
      increase time
         ✓ increase time
      batchHarvest
         ✓ harvest rewards: sell senior for leveraged junior returns (246ms)
         ✓ harvest rewards: sell all junior to partially cover senior (235ms)
         ✓ harvest rewards: sell some junior to cover senior (143ms)
      batchMidWithdraw
         ✓ withdraw LP mid-duration with signer 4: sell senior for leveraged junior returns (111ms)
         ✓ withdraw LP mid-duration with signer 4: sell all junior to partially cover senior (114ms)
         ✓ withdraw LP mid-duration with signer 4: sell some junior to cover senior (100ms)
      batchMidDeposit
         ✓ deposit LP tokens mid-duration with signer 6: sell senior for leveraged junior returns (522ms)
         ✓ deposit LP tokens mid-duration with signer 6: sell all junior to partially cover senior (500ms)
         ✓ deposit LP tokens mid-duration with signer 6: sell some junior to cover senior (498ms)
      batchHarvest
         ✓ harvest rewards: sell senior for leveraged junior returns (143ms)
         ✓ harvest rewards: sell all junior to partially cover senior (138ms)
         ✓ harvest rewards: sell some junior to cover senior (141ms)
      increase time
         ✓ increase time
      batchClaim
         ✓ claim tranche tokens: sell senior for leveraged junior returns (85ms)
         ✓ claim tranche tokens: sell all junior to partially cover senior (81ms)
         ✓ claim tranche tokens: sell some junior to cover senior (91ms)
      redeem and withdraw
         ✓ redeem LP after fee accrual (3694ms)

  alchemix
    alchemix vault lifecycle
       ✓ get assets (239ms)
       ✓ add pool
       ✓ create vaults (2969ms)
       ✓ deposit assets and fast-forward (10059ms)
       ✓ invest ALCX/ETH vault (9733ms)
       ✓ invest ETH/ALCX vault (1122ms)
       ✓ harvest (242ms)
       ✓ harvest (357ms)
       ✓ harvest (428ms)
       ✓ harvest (413ms)
       ✓ harvest (398ms)
       ✓ harvest (379ms)
       ✓ redeem ALCX/ETH vault (2398ms)
       ✓ redeem ETH/ALCX vault (573ms)

  TrancheToken
     ✓ spin up tranche tokens on Vault creation (1381ms)
     ✓ deposit base assets during Deposit phase (109ms)
     ✓ claim tokens only after investment (141ms)
     ✓ approve and transferFrom

  Uni
     ✓ test create mock (5161ms)
     ✓ test uniPull (484ms)

  AllPairVault
    delayed Vault
       ✓ setup vault fixture (419ms)
       ✓ create vault (1496ms)
       ✓ can only deposit after start time (1281ms)
       ✓ can get multiple vaults back from getVault (4207ms)
       ✓ deposits after start time (63ms)
       ✓ deposits exceed user cap
       ✓ deposits exceed tranche cap (103ms)
       ✓ can't deposit after investment
    withdraw midterm LP deposit
       ✓ setup vault fixture (2361ms)
       ✓ create vault (1647ms)
       ✓ deposit senior asset (115ms)
       ✓ deposit junior asset (122ms)
       ✓ deposits as expected
       ✓ cannot invest too early
       ✓ invest assets (87ms)
       ✓ can't deposit or withdraw midterm unless tranche tokens are enabled
       ✓ deposit LP tokens mid-duration with signer 7 (330ms)
       ✓ withdraws as expected after depositing LP without claiming (188ms)
       ✓ withdraw received amounts (149ms)
    sell senior for leveraged junior returns
       ✓ setup vault fixture (3994ms)
       ✓ create vault (1380ms)
       ✓ deposit senior asset (136ms)
       ✓ deposit junior asset (126ms)
       ✓ deposits as expected
       ✓ cannot invest too early
       ✓ gets Vault by tranche token addresses
       ✓ gets all Vaults
       ✓ invest assets (97ms)
       ✓ claim tranche tokens and excess (150ms)
       ✓ withdraw LP mid-duration from original deposits (217ms)
       ✓ deposit LP tokens mid-duration with signer 6 (379ms)
       ✓ deposit LP tokens mid-duration with signer 7 (361ms)
       ✓ withdraw LP mid-duration with signer 6 (73ms)
       ✓ withdraw LP mid-duration with signer 7 (70ms)
       ✓ redeem LP after fee accrual (220ms)
      ✓ withdraw received amounts (235ms)
    sell all junior to partially cover senior
      ✓ setup vault fixture (3943ms)
      ✓ create vault (1732ms)
      ✓ deposit senior asset (129ms)
      ✓ deposit junior asset (123ms)
      ✓ deposits as expected
      ✓ cannot invest too early
      ✓ invest assets (87ms)
      ✓ claim tranche tokens and excess (137ms)
      ✓ withdraw LP mid-duration from original deposits (198ms)
      ✓ deposit LP tokens mid-duration with signer 6 (347ms)
      ✓ deposit LP tokens mid-duration with signer 7 (345ms)
      ✓ withdraw LP mid-duration with signer 6 (69ms)
      ✓ withdraw LP mid-duration with signer 7 (68ms)
      ✓ redeem LP after fee accrual (189ms)
      ✓ withdraw received amounts (228ms)
    sell some junior to cover senior
      ✓ setup vault fixture (3465ms)
      ✓ create vault (1376ms)
      ✓ deposit senior asset (122ms)
      ✓ deposit junior asset (121ms)
      ✓ deposits as expected
      ✓ cannot invest too early
      ✓ invest assets (89ms)
      ✓ claim tranche tokens and excess (135ms)
      ✓ withdraw LP mid-duration from original deposits (208ms)
      ✓ deposit LP tokens mid-duration with signer 6 (335ms)
      ✓ deposit LP tokens mid-duration with signer 7 (349ms)
      ✓ withdraw LP mid-duration with signer 6 (65ms)
      ✓ withdraw LP mid-duration with signer 7 (66ms)
      ✓ redeem LP after fee accrual (204ms)
      ✓ withdraw received amounts (228ms)


  259 passing (6m)
```

Code Coverage

Quantstamp usually recommends developers to increase the branch coverage to `90%`and above before a project goes live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase. For code coverage, the current targeted files by the audit achieve poor scores that must be improved before live deployment.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	85.71	63.78	85.05	86.32	
AllPairVault.sol	97.45	72.55	97.87	97.5	… 1,1064,1112
OndoRegistryClient.sol	100	100	100	100	
OndoRegistryClientInitializable.sol	85.71	62.5	71.43	86.67	47,51
Registry.sol	64.29	50	66.67	65.52	… 147,148,160
RolloverVault.sol	74.9	55.88	81.48	76.35	… 823,828,880
SampleFeeCollector.sol	100	50	100	100	
TrancheToken.sol	71.43	25	63.64	68.75	… 101,110,119
contracts/dao/	0	0	0	0	
GovernorBravoDelegate.sol	0	0	0	0	… 583,584,587
GovernorBravoDelegator.sol	0	0	0	0	… 66,67,81,83
GovernorBravoInterfaces.sol	100	100	100	100	
SafeMath.sol	0	0	0	0	… 184,203,204
Timelock.sol	0	0	0	0	… 186,188,193
contracts/interfaces/	100	100	100	100	
IBasicVault.sol	100	100	100	100	
IFeeCollector.sol	100	100	100	100	
IPairVault.sol	100	100	100	100	
IRegistry.sol	100	100	100	100	
IRollover.sol	100	100	100	100	
IStrategy.sol	100	100	100	100	
ITrancheToken.sol	100	100	100	100	
IUserTriggeredReward.sol	100	100	100	100	
IWETH.sol	100	100	100	100	
contracts/libraries/	100	100	100	100	
OndoLibrary.sol	100	100	100	100	
contracts/strategies/	80.72	54.55	86.08	81.14	
AlchemixLPStrategy.sol	73.03	54.35	75	73.38	… 441,473,474
BasePairLPStrategy.sol	**77.78**	**50**	**100**	**78.95**	**85,86,87,88**
SushiStakingV2Strategy.sol	66.05	42.11	69.57	66.67	… 784,785,791

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
SushiStrategyLP.sol	100	66.67	100	100	
UniswapStrategy.sol	100	78.57	100	100	
contracts/strategies/helpers/	63.64	25	50	63.64	
AlchemixUserReward.sol	63.64	25	50	63.64	30,31,32,42
contracts/test/	40.38	100	36.84	40.38	
ERC20Mock.sol	40	100	75	40	18,20,21
ForceSendEth.sol	100	100	100	100	
IRewarder.sol	100	100	100	100	
Imports.sol	100	100	0	100	
MockRewarder.sol	0	100	0	0	… 37,38,39,40
TestRewardHelper.sol	0	100	0	0	11,15,19,20
UniPull.sol	47.06	100	50	47.06	… 44,45,46,47
contracts/tokens/	74.86	54.05	83.33	75.68	
Ondo.sol	61	47.83	78.26	62.75	… 309,312,396
StakingPools.sol	91.57	64.29	92.31	91.57	… 144,259,260
contracts/vendor/abdk/	100	100	0	0	
ABDKMathQuad.sol	100	100	0	0	… 2,1664,1677
contracts/vendor/alchemix/	100	100	100	100	
IStakingPools.sol	100	100	100	100	
contracts/vendor/sushiswap/	100	100	100	100	
IMasterChef.sol	100	100	100	100	
IMasterChefV2.sol	100	100	100	100	
ISushiBar.sol	100	100	100	100	
contracts/vendor/uniswap/	58.82	25	62.5	58.82	
SushiSwapLibrary.sol	58.82	25	62.5	58.82	… 129,130,132
UniswapV2Library.sol	58.82	25	62.5	58.82	… 129,130,132
All files	**70.12**	**43.84**	**62.31**	**69.69**	

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

```
6a7f22d2234be1614b349c6eb1a5793f25209ebb12cffff9954685ee810d7c0b ./strategies/SushiStakingV2Strategy.sol

e98aa22663acea84b9930927eb4d5d043afdaba7ee0db2bfb93e4249f37a9040 ./strategies/helpers/AlchemixUserReward.sol
```

Tests

```
908edcc0cb080e925404af48107fa5607cf377a86691c1d69930d845c2ed3064 ./test/registry.spec.ts

567a9c7e1b3987bc44df301dfe8e47c4e9c0da6417c9b9fbe41084fc202a65fb ./test/alchemix.spec.ts

731ec323616913c12b4d4a053d966f1a9c7b3694346cf1cffbc4a914a6a2b214 ./test/staking.spec.ts

7f32b85cd16273b127afdffe45c6f7cbcfa74961af421a37aeaa79a9a68b8909 ./test/sushi2.spec.ts

641794a46f5826bd3f75e17ba1e88d705bb35c168cb59d94fd912b2bb04b7f84 ./test/ondo.spec.ts

55884a6ce789c9e7eb49d014ad5b61cb82c9342587cb8785a5e0a1ad179f17bd ./test/sushi-lp.spec.ts

898f83e353c4d0408e138b8058bab776d9ea90ef2a1fc6bc4456ed6573d2497a ./test/claim.spec.ts

4d4cced62f58bb967fdb051981cf30042e2feb4cbb5f036dd6d2bfdf0e5f3aaa ./test/token.spec.ts

632ea85d33b6cf9f011e6e1ef462cb66c2c4965ae44ff082f5ade451ae1f1c0e ./test/uni.spec.ts

ffededa76363b9e27cd3a5e1feb028fd676b2f9c0b9d668c5a3fd505691cc5df ./test/reverts.spec.ts

5d0def6f1d8d660ea6f3a968da264c422d4e9e5a1e84032f95cdb4129f21d757 ./test/rollover.spec.ts

12d8149fdc16ef1d733fb68edee18e01b1cdfe742e514208256effb1434319b3 ./test/vault.spec.ts

967a06267b1ed917e6976c4d0511034cc98218b77f5e6f047e896763c90c6dca ./test/fee.spec.ts

f53f15453e843ef9d70317f9e0b312615c3457bedb3d558fe949d82d8768f039 ./test/create2.spec.ts

d4166571a3d52d03dec09a09bc16e9bac304f5e820da1800530a1b7e86f1f084 ./test/rescue.spec.ts

3555c740183d0ca7153eb1ac8f27e1ceb4110e7062a49a5a1b2bc57052abc961 ./test/utils/bignumberhax.ts

295e2a0d7763e85150d6eaa8bcf1779b6e5b42e760b0c1f083ce02b105d85657 ./test/utils/DebugSigner.ts

ab65a2857c1fb33dc00d7181aac429709391d1c8e0bbc484756d14889011a696 ./test/utils/vault.ts

66e8bc2c94fba576664f9b3835e7622f32c416aa20b619e1e6d6b78118992477 ./test/utils/uni.ts

5db8a0062c5cc308c9114d7bfd4de63a940a3c0cbd0ea278e218d6f56f3aa90b ./test/utils/addresses.ts

95e0919c58dd3dd10cd81597a7a034888902e174046af6e66c269077ddad8810 ./test/utils/signing.ts

d629e629864f5062bfa441b427ad696853b1b00e644eb632fc9cc93d4d8bdecf ./test/utils/getters.ts
```

Changelog

- 2021-09-03 - Initial report

About Quantstamp

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

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

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

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