Quantstamp Security Assessment Certificate

January 25th 2022 — Quantstamp Verified

Ondo Finance 3

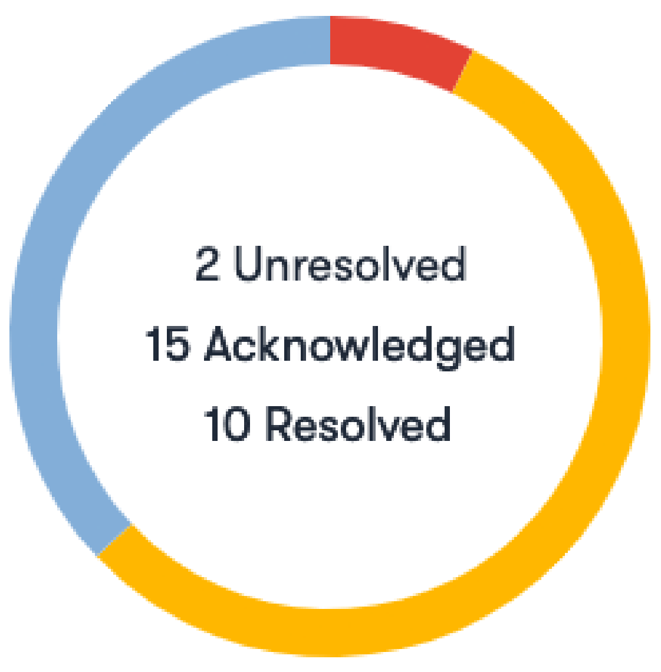
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Yield Aggregator
Auditors	Mohsen Ahmadvand, Senior Research Engineer Poming Lee, Research Engineer
Timeline	2021-12-01 through 2021-12-22
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Undetermined
Source Code	

Repository	Commit
protocol-dev	b574763
None	f2c7b97 (re-audit)

Total Issues	27	(10 Resolved)
High Risk Issues	4	(2 Resolved)
Medium Risk Issues	1	(1 Resolved)
Low Risk Issues	7	(3 Resolved)
Informational Risk Issues	11	(4 Resolved)
Undetermined Risk Issues	4	(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

This audit was done on a subset of contracts in the Ondo repository (an exhaustive list of contracts can be found in the appendix). In our incremental audit we identified four high, one medium, six low, eleven informational, and four undetermined-severity issues. Two of the high-severity issues are logical problems. Such issues should be detected through extensive testing. The other two high-severity issues relate to the possibility of malicious behavior by the privileged roles in the protocol. The platform heavily relies on several external protocols (Sushiswap, Uniswap, Pancake, etc.) across several chains. Consequently, Ondo will inherit all the vulnerabilities in its external dependencies. A large portion of the code duplicate of strategies. This makes code maintenance and increases the chances of pitfalls.

ID	Description	Severity	Status
QSP-1	`SushiStakingV2Strategy.depositIntoChef` can possibly inflate `vault.shares`	High	Fixed
QSP-2	`DopexStrategy.removeLp` deducts 2x LP per call	High	Fixed
QSP-3	`multiexcall` enables the guardian role to execute arbitrary transactions	High	Acknowledged
QSP-4	`_updateInvestor` always sets `investorLastUpdates` to `0`	High	Acknowledged
QSP-5	Use of unsafe transfers	Medium	Fixed
QSP-6	`EdenStrategy.harvest` can potentially suffer from high slippages	Low	Fixed
QSP-7	`uniRouter02.removeLiquidity` and `uniRouter02.addLiquidity` calls are suceptible to frontrunning attacks	Low	Acknowledged
QSP-8	Ownership can be renounced	Low	Acknowledged
QSP-9	`amountTransferredFromUser` in the `Deposited` event is faulty	Low	Fixed
QSP-10	Unintended unstake in the Eden strategy	Low	Fixed
QSP-11	`tokenMinting` of tranche tokens can cause inconsistencies	Low	Acknowledged
QSP-12	`RolloveVault._getUpdatedInvestor` doesn't check inputs	Informational	Unresolved
QSP-13	Discrepancy between comments and implementation of `newRollover`	Informational	Fixed
QSP-14	Why do rounds start from 1 and not zero?	Informational	Acknowledged
QSP-15	Heavy storage usage (expensive gas)	Informational	Acknowledged
QSP-16	`userCap` of a vault to a user investing through `Rollover` is not always valid	Informational	Acknowledged
QSP-17	Use of initialised variables	Informational	Fixed
QSP-18	Ignored return values	Informational	Acknowledged
QSP-19	Missing zero input checks	Informational	Fixed
QSP-20	Dependency on several external contracts	Informational	Acknowledged
QSP-21	Allowance Double-Spend Exploit	Informational	Acknowledged
QSP-22	The enablement of `tokenMinting` of tranche tokens is not checked	Informational	Fixed
QSP-23	Privileged Roles and Ownership	? Undetermined	Acknowledged
QSP-24	Invalid and detrimental paths can be supplied to `_seniorToJuniorPath` and `_juniorToSeniorPath`	? Undetermined	Acknowledged
QSP-25	`getNextVault` skips one round	? Undetermined	Acknowledged
QSP-26	Potentially dangreous use of strict equality in `createVault`	? Undetermined	Acknowledged
QSP-27	`excall` function lets the guardian to execute arbitrary transfers	Low	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 v0.8.1
- Mythril v0.2.7

Steps taken to run the tools:

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

Findings

QSP-1 `SushiStakingV2Strategy.depositIntoChef` can possibly inflate `vault.shares`

Severity: High Risk

Status: Fixed

File(s) affected: `contracts/strategies/SushiStakingV2Strategy.sol:483`, `contracts/strategies/SushiStrategyLP.sol:335`

Description: The implementation of `depositIntoChef` in `SushiStakingV2Strategy` treats vault shares slightly differently than the corresponding implementation in `SushiStrategyLP`.
SushiStakingV2Strategy:

```
if (poolData.totalLp == 0 || poolData.totalShares == 0) {
    poolData.totalShares = _amount * sharesToLpRatio;
    poolData.totalLp = _amount;
    vault.shares = _amount * sharesToLpRatio;
  } else {
    uint256 shares = (_amount * poolData.totalShares) / poolData.totalLp;
    poolData.totalShares += shares;
    vault.shares += shares; //HERE the shares are added to the current value
```

```
      poolData.totalLp += _amount;
    }
   if (poolData.totalLp == 0 || poolData.totalShares == 0) {
      poolData.totalShares = _amount;
      poolData.totalLp = _amount;
      vault.shares = _amount;
    } else {
      uint256 shares = (_amount * poolData.totalShares) / poolData.totalLp;
      poolData.totalShares += shares;
      vault.shares = shares; // HERE the shares are simply assigned
      poolData.totalLp += _amount;
    }
```

It is unclear why the two functions differ in vault share assignments.

Recommendation: Ensure that the identified discrepancy is not a bug. Consider writing test cases for the captured logical discrepancy between the two contracts.

QSP-2 `DopexStrategy.removeLp` deducts 2x LP per call

Severity: *High Risk*

Status: Fixed

File(s) affected: `contracts/strategies/DopexStrategy.sol:145`

Description: The `removeLp` function calls `withdrawFromStaking` that already includes a `totalLp` amount deduction: contracts/strategies/DopexStrategy.sol:115

```
  function withdrawFromStaking(uint256 _amount) internal {
    require(_amount > 0, "totalLP must be greater than 0");
    stakingContract.withdraw(_amount);
    totalLP -= _amount;
  }
```

Nevertheless, the `removeLp` function includes another `totalLp` amount deduction resulting in a 2x value loss.

Recommendation: Remove the amount deduction from the `removeLp` function. Consider adding test cases capturing this very bug for all your strategy contracts.

QSP-3 `multiexcall` enables the guardian role to execute arbitrary transactions

Severity: *High Risk*

Status: Acknowledged

File(s) affected: `contracts/strategies/UniswapStrategy.sol`, `contracts/strategies/SushiStakingV2Strategy.sol`, `contracts/strategies/AUniswapStrategy.sol`

Description: The `multiexcall` function provides batch execution to the guardian role. The user with such a role can execute any harmful transactions, e.g., to deplete the funds.

Recommendation: Ensure that users are informed about the guardians power in your protocol. Consider using multi-sig with a large enough set of users to mitigate the risk.

Update: Response from Ondo:

> `Multiexcall` can be called by Guardian which is in turn a multisig and community is aware of this functionality

QSP-4 `_updateInvestor` always sets `investorLastUpdates` to `0`

Severity: *High Risk*

Status: Acknowledged

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

Description: `_updateInvestor:L447` always sets `investorLastUpdates` to `0` instead of updating it to the index of the current round. This can potentially break the rounds accounting logic.

Recommendation: Ensure that the stated code does not lead to problems. We assume the correct behavior should be assigning `investorLastUpdates` to `rollover.thisRound + 1`.

Update: Response from Ondo: > the internal variable investorLastUpdates is used to find the round number in which investor's funds are associated with. This variable is updated whenever there is change of state to investor's funds i.e., claim, withdraw or deposits. When a new deposit is made, the round is updated to the round to which investor's funds are associated. When claim, withdraw or a new deposit happens, if investorLastUpdates is 0 - then the shares and excess for the investor are computed from clean state. Else, existing shares,excess from the current round are extracted and the investorLastUpdates is set to 0 because all the shares and excess are returned to the investor.

QSP-5 Use of unsafe transfers

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts/strategies/helpers/AlchemixUserReward.sol`

Description: The contract does not check whether the transfer was succesful or not. More importantly, the usage of transfer is discouraged as it may run out of gas.

Recommendation: Consider using openzeppelin's `safeTransfer` that reverts in case of failure.

QSP-6 `EdenStrategy.harvest` can potentially suffer from high slippages

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/strategies/EdenStrategy.sol`

Description: The `harvest` function does not employ minimum expected value checks. This can enable attacker (bots) to potentially utilize flash loans imposing high slippages on the protocol.

Recommendation: Consider enforcing minimum expected value checks.

QSP-7 `uniRouter02.removeLiquidity` and `uniRouter02.addLiquidity` calls are suceptible to frontrunning attacks

Severity: Low Risk

Status: Acknowledged

File(s) affected: `contracts/strategies/UniswapStrategy.sol`, `contracts/strategies/AUniswapStrategy.sol`

Description: We identified several calls to the `addLiquiditiy` and `removeLiquidity` functions that set the minimum expected tokens to 0. This way of interacting with AMM pools is susceptible to sandwich attacks.
Here are code pointers to insecure AMM invocations:

1. `contracts/strategies/AUniswapStrategy.sol:358~366`
2. `contracts/strategies/AUniswapStrategy.sol:456~463`
3. `contracts/strategies/AUniswapStrategy.sol:504~511`
4. `contracts/strategies/UniswapStrategy.sol:321~329`

Recommendation: Consider computing the minimum expected values prior to the calls and checking them in your contracts.

Update: Response from Ondo:

> We currently have checks outside the functions which make sure a certain slippage doesn't exceed but we acknowledge that we can try to do better and use non 0 values.

QSP-8 Ownership can be renounced

Severity: Low Risk

Status: Acknowledged

File(s) affected: `Ondo.sol`

Description: `Ownable.sol` has a `renounceOwnership()` function. Although it can only be called by the owner, contracts with (accidentally or maliciously) renounced ownerships can cause a denial of service.
Particularly, Ondo token's transferable state can no longer be set after ownership is renounced.

Recommendation: Consider using a two-step ownership transfer scheme and disabling the `renounceOwnership` function.

Update: Response from Ondo:

> Ondo.sol is taken from Compound and uses the same trust assumptions that Compound token does. The owner will be a multis with 3 of 5 at least and hence its okay to keep this for now.

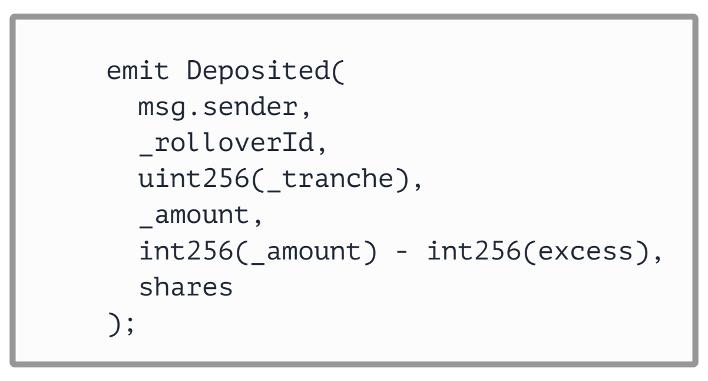
QSP-9 `amountTransferredFromUser` in the `Deposited` event is faulty

Severity: Low Risk

Status: Fixed

File(s) affected: `contracts/RolloverVault.sol:412~413`

Description: The amount deposited depends on the exceeding amount. If the exceeding amount is larger than the requested deposit amount, then nothing is transferred from the user. Instead, the diff between the exceeding amount and the intended deposit amount is transferred back to the user. In the emitted event the `amountTransferredFromUser` captures an invalid value.
This results in incorrect information being displayed to the users in the frontend.



```
emit Deposited(
  msg.sender,
  _rolloverId,
  uint256(_tranche),
  _amount,
  int256(_amount) - int256(excess),
  shares
);
```

Recommendation: Fix the `amountTransferredFromUser` value in the emit.

QSP-10 Unintended unstake in the Eden strategy

Severity: Low Risk

Status: Fixed

File(s) affected: `contracts/strategies/EdenStrategy.sol`

Description: Whenever `edenAmount` is smaller than `10000`, the function `harvest` will not restake (i.e., calling `depositIntoStaking`) after unstaking on `L158`, leaving all the `totalLP` remaining unstaked and kept in the strategy contract.

Recommendation: Make sure to restake at the end of function `harvest` in all conditions.

QSP-11 `tokenMinting` of tranche tokens can cause inconsistencies

Severity: Low Risk

Status: Acknowledged

File(s) affected: `contracts/Registry.sol`

Description: The functions in `contracts/Registry.sol`: `L78`-`L84` allow `OLib.GOVERNANCE_ROLE` to enable/disable the `tokenMinting` of tranche tokens literally "at any time". This could lead to internal accounting problems, such as part of the users getting the tranche tokens where some of the users do not.

Recommendation: This feature should be limited, removed, or at least having all the changes to the platform carefully estimated by the admin team every time before enabling/disabling this feature.

Update: Response from Ondo:

> `GOVERNANCE_ROLE` is only given to a governance multis which is 3 of 5 and will never be used to turn on token_minting and turn it off, we are aware of the inconsistencies that can occur if tokenMinting is turned on and off. It's meant to be used as a one time switch to turn on tokenMinting.

QSP-12 `RolloveVault._getUpdatedInvestor` doesn't check inputs

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/RolloverVault.sol:220~221`

Description: `RolloveVault.getUpdatedInvestor` doesn't check if `_rolloverId`, `_tranche` inputs actually exist. It is indirectly consumed by several state-changing functions, viz. `_updateInvestor`, `_updateInvestorDistribute`, `deposit`, `withdraw`, and `claim`.

Recommendation: Ensure that invalid inputs do not have an impact on the listed functions by adding test cases. Consider adding checks if inputs are valid.

Update: The newly added require statement checks if creator is not address zero:

```
require(rollover.creator != address(0), "Invalid rolloverId");
```

The check should verify if the provided `_rolloverId`, `_tranche` values exist.

QSP-13 Discrepancy between comments and implementation of `newRollover`

Severity: Informational

Status: Fixed

File(s) affected: `contracts/RolloverVault.sol`

Description: The `newRollover` function reverts upon a zero value for the `_vaultId` argument. However, the comment indicates that in such cases params should be used as the basis to create a new rollover: ``` * @dev If _vaultId is 0, use _params for everything. Otherwise, inherit most params from _vaultId instance

- @param _vaultId Optional Vault to inherit parameters from ... require(_vaultId != 0, "Invalid vaultId"); ```

Recommendation: Ensure that the revert reflects the intended behavior.

QSP-14 Why do rounds start from 1 and not zero?

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/RolloverVault.sol:231`

Description: The rollover.rounds considers index 1 as the starting element. It is not clear why index 0 is skipped.

```
function newRollover(uint256 _vaultId, OLib.RolloverParams memory _params)
    Rollover storage rollover = rollovers[rolloverId];
    require(rollover.rounds[1].vaultId == 0, "Already exists");
    ...

    rollover.rounds[1].vaultId = _vaultId;
```

Recommendation: Ensure that the logic is correct.

Update: Response from Ondo:

> Round 0 is considered a special round which means open for deposits.

QSP-15 Heavy storage usage (expensive gas)

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/RolloverVault.sol`

Description: The implementation is extremely storage-intensive. This can lead to expensive transactions or possibly denial of service if a transaction becomes larger than one block.

```
rollovers ->   mapping(uint256 => Rollover) internal rollovers;

struct Rollover {
    address creator;
    address strategist;
    mapping(uint256 => Round) rounds;
    mapping(OLib.Tranche => IERC20) assets;
    mapping(OLib.Tranche => ITrancheToken) rolloverTokens;
    mapping(OLib.Tranche => mapping(address => uint256)) investorLastUpdates;
    uint256 thisRound;
    bool dead;
}

struct Round {
    uint256 vaultId;
    mapping(OLib.Tranche => TrancheRound) tranches;
}

struct TrancheRound {
    uint256 deposited;
    uint256 invested; // Total, if any, actually invested
    uint256 redeemed; // After Vault is done, total tokens redeemed for LP
    uint256 shares;
    uint256 newDeposited;
    uint256 newInvested;
    mapping(address => OLib.Investor) investors;
}

struct Investor {
    uint256[] userSums;
    uint256[] prefixSums;
```

```
    bool claimed;
    bool withdrawn;
}

Round storage round = rollover.rounds[rollover.thisRound + 1]; ->
round.tranches[_tranche];
OLib.Investor storage investor = trancheround.investors[msg.sender];
investor.userSums[...]
vaultView.assets[uint256(_tranche)]
```

For instance, storing every deposited amount (`userSums`) might not be necessary to achieve the intended goals.

Recommendation: Perform a gas usage analysis. Consider refactoring the code to use less storage when possible.

Update: Response from Ondo:

> This is the design of contracts and cannot be changed at this stage, we will take this into consideration for the next iterations

QSP-16 `userCap` of a vault to a user investing through `Rollover` is not always valid

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/RolloverVault.sol`

Description: Currently for the Rollover contract, whenever a user calls `deposit`, the contract will check if `userSum <= userCap`, where `userCap` is for the vault for the `rollover.thisRound + 1` round. However, if the user already invested before and never calls that `deposit` function again, this check will never be performed again. And this restriction will no longer be valid when the `userCap` of vaults in the following rounds is smaller than the `userSum` of the user.

Recommendation: There is no solution to fix this problem without redesigning the mechanism. We recommend the admin team to evaluate if this level of violation to the restriction is acceptable.

Update: Response from Ondo:

> The check is to stop users from depositing into the vault beyond the usercap. However, if the usercap is exceeded in the subsequent rounds we shouldn't limit rounds with this restriction. That is how the contract is designed

QSP-17 Use of initialised variables

Severity: Informational

Status: Fixed

Description: Several usage of uninitialised variables were detected throughout the source code:

```
EdenStrategy.harvest(uint256).wethAmount (contracts/strategies/EdenStrategy.sol#161) is a local variable never initialized
BondStrategy.harvest(uint256).usdcAmount (contracts/strategies/BondStrategy.sol#157) is a local variable never initialized
AlchemixLPStrategy._compound().ethAmount (contracts/strategies/AlchemixLPStrategy.sol#328) is a local variable never initialized
RolloverVault.deposit(uint256,OLib.Tranche,uint256).excess (contracts/RolloverVault.sol#358) is a local variable never initialized
AlchemixLPStrategy._compound().amountToSwap (contracts/strategies/AlchemixLPStrategy.sol#372) is a local variable never initialized
RolloverVault._getUpdatedInvestor(address,uint256,OLib.Tranche).shares (contracts/RolloverVault.sol#199) is a local variable never initialized
SushiStakingV2Strategy._compound(IERC20,SushiStakingV2Strategy.PoolData).amountToSwap (contracts/strategies/SushiStakingV2Strategy.sol#408) is
a local variable never initialized
DopexStrategy.harvest(uint256).wethAmount (contracts/strategies/DopexStrategy.sol#160) is a local variable never initialized
```

Recommendation: Ensure to initialise every variable before use to avoid unexpected results.

QSP-18 Ignored return values

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/RolloverVault.sol:667~674`, `contracts/RolloverVault.sol:799~800`

Description: `RolloverVault._migrate` and `RolloverVault._invest` ignore return values of `AllPairsVault.redeem` and `AllPairsVault.invest`, respectively.

Recommendation: Consider not returning anything if the values are not needed.

Update: Response from Ondo:

> we need these return values to get minimums to prevent slippage while executing non-rollover vault invest/redeem

QSP-19 Missing zero input checks

Severity: Informational

Status: Fixed

Description: The listed contracts do not check the input parameters:

1. `contracts\SampleFeeCollector.sol`: `L23`: address `vault` in `constructor`.
2. `contracts\TrancheToken.sol`: `L37`: `_vault`.
3. `contracts\Registry.sol`: `L35-L37`: `_governance` and `_weth`.

Recommendation: Consider checking the input parameters.

QSP-20 Dependency on several external contracts

Severity: Informational

Status: Acknowledged

Description: The protocol relies on functionalities of external contracts such as: UniswapV2, Sushiswap, Quickswap, Pancakeswap, etc. Therefore, security of the project depends on these contracts. While we are unaware of any immediate issues, it is important to note that DeFi protocols can be susceptible to flash loan attacks, market manipulation, computational errors, etc.

Recommendation: We strongly recommend to monitor all the external contracts and follow up on their vulnerabilities. Consider integrating logics to redeem all funds quickly from strategies in case of zero-day hacks.

Update: Response from Ondo:

> We will monitor all the contracts that we are interacting with and have a test suite to test it with forking

QSP-21 Allowance Double-Spend Exploit

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/TrancheToken.sol`

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

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

Update: Response from Ondo:

> TrancheToken is mostly taken from open zeppelin and has the functions that are recommended, we will recommend the users to use increaseAllowance instead of approve.

QSP-22 The enablement of `tokenMinting` of tranche tokens is not checked

Severity: Informational

Status: Fixed

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

Description: The functions that trigger mint do not check whether minting is enabled or not: The enablement of `tokenMinting` of tranche tokens is not checked in:

1. `contracts\AllPairVault.sol`: `L541-L544`
2. `contracts\AllPairVault.sol`: `L545-L548`
3. `contracts\AllPairVault.sol`: `L896-L903`
4. `contracts\AllPairVault.sol`: `L1142`

Recommendation: Consider checking whether the mint is enbaled before calling `mint`.

QSP-23 Privileged Roles and Ownership

Severity: Undetermined

Status: Acknowledged

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

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. There are some actions that could have important consequences for end-users. The `OLib.GUARDIAN_ROLE` in the Ondo platform is be able to execute arbitrary code on `contracts\AllPairVault.sol`.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update: Response from Ondo:

> Multiexcall can be called by Guardian which is in turn a multisig.

QSP-24 Invalid and detrimental paths can be supplied to `_seniorToJuniorPath` and `_juniorToSeniorPath`

Severity: Undetermined

Status: Acknowledged

File(s) affected: `contracts/strategies/UniswapStrategy.sol`

Description: `setPathJuniorToSenior` and `setPathSeniorToJunior` functions accepts any arbitrary conversion paths. Invalid paths (paths that do not end with junior or senior tranche tokens accordingly) can lead to undetermined issues in the protocol. Moreover, paths can have different economical characteristics. As constructed, the strategist role can set those paths. A strategist can potentially favour paths to their benefit and in determent to other users (e.g., to include their own pools by forcing unnecessary swaps to certain tokens).

Recommendation: Ensure the the path array's last element matches the senior and junior tranche tokens in the `setPathJuniorToSenior` and `setPathSeniorToJunior` functions, respectively. Inform the users (through the public-facing documentations) about the strategist's power in setting paths in the protocol. Consider enabling the protocol users to vote on the swap paths or use third-party contracts to compute paths with the highest returns.

Update: Response from Ondo:

> It's done on purpose to allow for arbitrary paths if needed, this strategy is mostly used with our DAO partners who can request different tokens or different trades and we did not want to limit setting the paths

QSP-25 `getNextVault` skips one round

Severity: Undetermined

Status: Acknowledged

File(s) affected: `contracts/RolloverVault.sol:181`

Description: `getNextVault` looks up `rollover.thisRound + 2` in `rollover.rounds`:

```
uint256 vaultId = rollover.rounds[rollover.thisRound + 2].vaultId;
```

In the previous audit the look up index was `rollover.thisRound + 1`.

Recommendation: Ensure that the implementation is correct. Consider adding test cases to verity the correctness.

Update: Response from Ondo:

> The implementation is correct. Added more tests to validate this.

QSP-26 Potentially dangreous use of strict equality in `createVault`

Severity: Undetermined

Status: Acknowledged

File(s) affected: `contracts/AllPairVault.sol:337`

Description: Description: The createVault function sets the `startAt` value to the passed input (`_params.startTime`). The function requires the startTime to be greater than/equal to the current block timestamp at line #267: `require(_params.startTime >= block.timestamp, "Invalid start time");` Thereupon, the function checks if the startAt is strictly equal to the current block's timestamp to decide whether to set the vault_.state to deposit (#337-#339):

```
  if (vault_.startAt == block.timestamp) {
    vault_.state = OLib.State.Deposit;
  }
```

It unclear what will happen if the deposit state condition is not met (i.e., `vault_.startAt != block.timestamp`). Given that the state of deposit can not be set otherwise, the contract may experience denial of service. It is not safe to rely on the block.timestamp to be what you expect it to be upon triggering a createVault call.

Recommendation: Ensure that the described circumstances does not lead to a denial of service. Consider turning the deposit state into a dynamically computed state, instead of storing it:

```
function canDepositIntoVault view internal (){
    return vault_.startAt <= block.timestamp;
}
```

Update: Response from Ondo:

> This isn't a bug, its an optimisation, the idea is that if the start time happens to be exactly the same as block time, then vault state changes to deposit. It is ensured that this does not cause a denial of service.

QSP-27 `excall` function lets the guardian to execute arbitrary transfers

Severity: Low Risk

Status: Unresolved

File(s) affected: `contracts/RolloverVault.sol`

Description: The newly added function enables the guardian to execute arbitrary transfers:

```
function excall(address target, bytes calldata data)
     external
     isAuthorized(OLib.GUARDIAN_ROLE)
     returns (bytes memory returnData)
   {
     bool success;
     (success, returnData) = target.call(data);
     require(success, "CF");
   }
```

It is not clear why the guardian needs to have such a privilege in the system. This can lead to fund losses if the guardian is compromised.

Recommendation: Include the stated function in the public-facing documentation and justify the need for such a function. Consider using multisig for the guardian role.

Automated Analyses

Slither

We analysed the findings and incorporated the relevant issues to the report.

Adherence to Specification

1. `contracts\AllPairVault.sol`: `L272`: should be `1e6` instead, otherwise should modify the warning message to `1000000%`.

Code Documentation

1. Consider describing the mechanism of `contracts\RolloverVault.sol` and `ROLLOVER_ROLE` in the documentations and roles.
2. `contracts\RolloverVault.sol`: `L37`: mis-spell in word `Rollove`.
3. `contracts\RolloverVault.sol`: `L42`: mis-spell in word `exces`.

Adherence to Best Practices

1. Strategies seem to have a lot in common. The strategy logic entails depositing assets, tracking LP tokens, withdrawing and re-investing (compounding), and redeeming. This common logic is to a great extend duplicated across the contracts. Maintaining and securing all the duplicated code will have a huge overhead. Consider refactoring the code such that all the common codes reside in one single contract. By restoring to hooks you can have the flexibility of calling platform-specific codes to write platform-agnostic strategies.

2. `contracts\RolloverVault.sol`: the use of class instance `TrancheToken` for rollover tokens is confusing. Consider creating another class that inherits `TrancheToken` and named it as `RolloverToken` to improve the clarity of the code.

3. `contracts\RolloverVault.sol`: `L731` the condition ">" is not possible. Consider reviewing the possibility of this condition again and make sure if the handling logic in `L732` is acceptable.

4. TODO in `contracts\strategies\DopexStrategy.sol`: `L91`.

5. TODO in `contracts\strategies\DopexStrategy.sol`: `L148`.

6. `contracts\AllPairVault.sol`: consider adding a zero check to the `_params.strategist` of function `createVault` in order to make sure that an uninitialised vault could never get pass the check (i.e., `address(vault_.strategist) == address(0)`) in `L302`.

7. `contracts\AllPairVault.sol`: `L337` please make sure that the strict equality check `==` in this line is intended, and cannot be replaced by `>=`.

8. `contracts\Registry.sol`: function `tokensDeclaredDead` uses a list `deadTokens` to record input dead tokens. This list does not guarantee the uniqueness of the dead tokens. This could lead to errors in the function `recycleDeadTokens` when trying to recycle the dead tokens. Please make sure that this condition does not happen.

9. `mainToken` is not initalized in the abstract class but used in some functions `contracts/strategies/ASushiswapStrategy.sol` `mainToken` is not initialized in the constructor of `ASushiswapStrategy`. However, it is used by some abstract function implementations. Both consumers of `ASushiswapStrategy`, viz. `bsc/contracts/strategies/PancakeStrategyLP.sol`, and `polygon/contracts/strategies/QuickswapStrategyLP.sol`, do initialize `mainToken` in their constructors. Nevertheless, it is a better practice to initialize all the used variables as part of the abstract class construtor. `solidity constructor( address _registry, address _router, address _rewardsFactory, address _factory, address _mainToken, address _stakingRewardToken ) ASushiswapStrategy(_mainToken /*initialize ASushiswapStrategy._mainToken*/, _registry, _router, _factory)`

10. apply the check-effects-interactions pattern to prevent re-entrancy attacks:
    1. AUniswapStrategy._invest(uint256,uint256,uint256,uint256,uint256,uint256,uint256) (contracts/strategies/AUniswapStrategy.sol#225-259)
    2. RolloverVault._migrate(uint256,IRollover.SlippageSettings) (contracts/RolloverVault.sol#659-723)
    3. AllPairVault._withdraw(uint256,OLib.Tranche,address) (contracts/AllPairVault.sol#824-852)
    4. SushiStakingV2Strategy.midTermDepositLp(IERC20,uint256) (contracts/strategies/SushiStakingV2Strategy.sol#190-219)
    5. SushiStrategyLP.midTermDepositLp(IERC20,uint256) (contracts/strategies/SushiStrategyLP.sol#199-209)
    6. RolloverVault.newRollover(uint256,OLib.RolloverParams) (contracts/RolloverVault.sol#231-302)
    7. AlchemixLPStrategy.removeLp(uint256,uint256,address) (contracts/strategies/AlchemixLPStrategy.sol#247-271)
    8. BondStrategy.removeLp(uint256,uint256,address) (contracts/strategies/BondStrategy.sol#134-143)
    9. DopexStrategy.removeLp(uint256,uint256,address) (contracts/strategies/DopexStrategy.sol#136-146)
    10. EdenStrategy.removeLp(uint256,uint256,address) (contracts/strategies/EdenStrategy.sol#138-147)
    11. SushiStakingV2Strategy.removeLp(uint256,uint256,address) (contracts/strategies/SushiStakingV2Strategy.sol#159-185)
    12. SushiStrategyLP.removeLp(uint256,uint256,address) (contracts/strategies/SushiStrategyLP.sol#158-182)

11. Consider declaring the following functions as external:
    1. rescueTokens(address[],uint256[]) (contracts/OndoRegistryClientInitializable.sol#73-80)
    2. authorized(bytes32,address) (contracts/Registry.sol#58-65)
    3. paused() (contracts/Registry.sol#99-101)

Test Results

Test Suite Results

13 tests are failing.

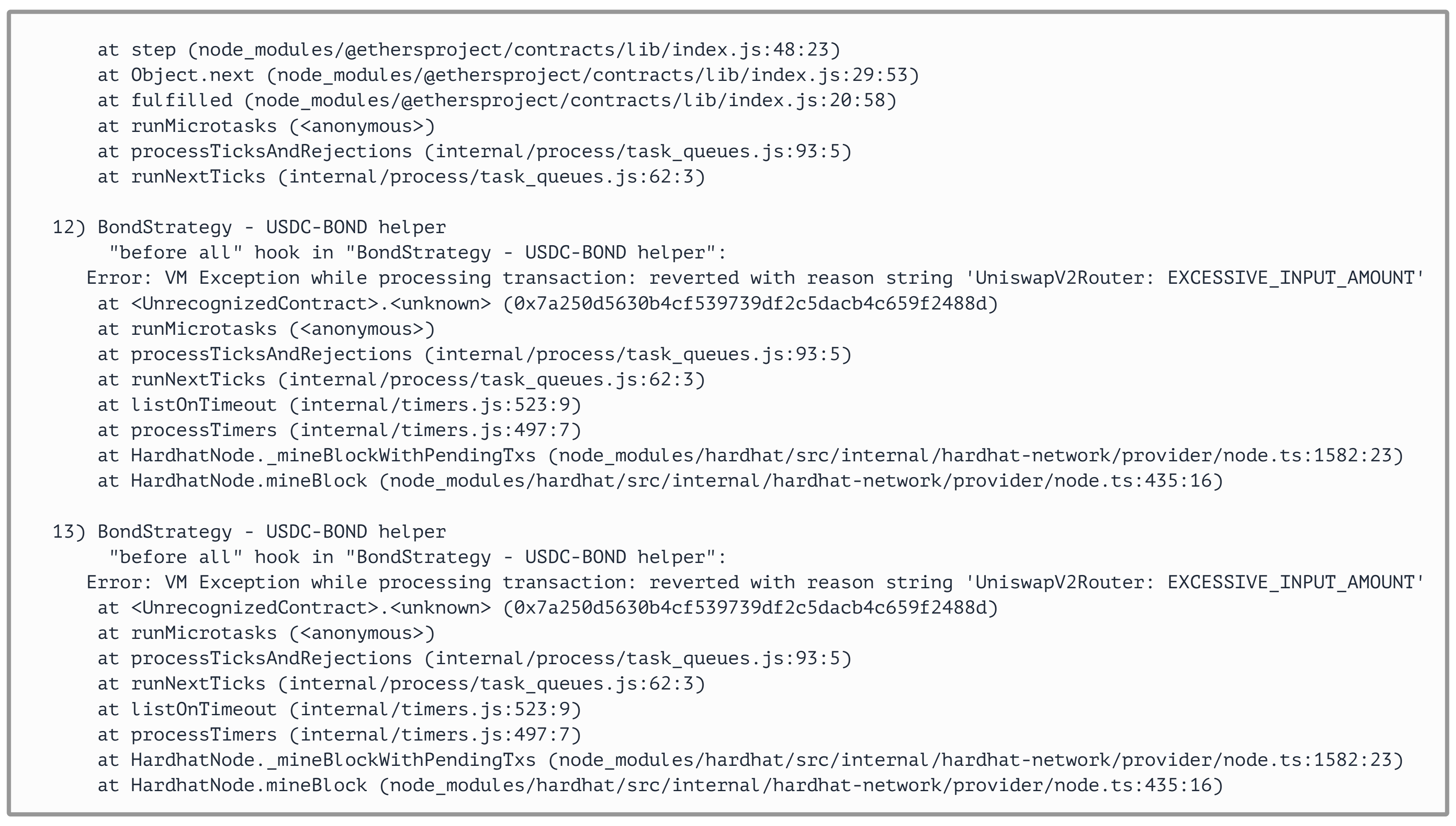
```
Network Info
============
> HardhatEVM: v2.6.2
> network:    hardhat

No need to generate any newer typings.
(node:8413) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)


  masterchefv2 dual incentive pool
    mcv2 vault lifecycle
```

```
      ✓ get assets (13584ms)
      ✓ create vaults (3996ms)
      ✓ deposit assets and fast-forward (10355ms)
      ✓ invest ALCX/ETH vault (35228ms)
      ✓ invest ETH/ALCX vault (6754ms)
     1) get LP from the sushiswap pool and deposit
     2) withdraw LP from ALCX/ETH vault
     3) deposit LP in ALCX/ETH vault
      ✓ harvest (353ms)
      ✓ harvest (342ms)
      ✓ harvest (359ms)
      ✓ harvest (336ms)
      ✓ harvest (335ms)
      ✓ harvest (335ms)
      ✓ redeem ALCX/ETH vault (3252ms)
      ✓ redeem ETH/ALCX vault (745ms)

  EdenStrategy - WETH-EDEN helper
    4) "before all" hook in "EdenStrategy - WETH-EDEN helper"

  BondStrategy - USDC-BOND helper
    USDC-BOND - deployment validation
      ✓ should not allow zero address for the staking address deploying uniswap strategy (609ms)
      ✓ should not allow zero address for the yieldfarm address deploying uniswap strategy
      ✓ should not allow zero address for the usdc address deploying uniswap strategy
      ✓ should not allow zero address for the bond address deploying uniswap strategy
      ✓ should not allow zero address for the usdcBondUniLp address deploying uniswap strategy
      USDC-BOND - vault lifecycle
        ✓ should allow Creator to create vault (1192ms)
        ✓ should NOT allow any one other than creator to create vault
        ✓ should allow investors to deposit assets in the junior tranche when with in the user and tranche cap (3701ms)
        ✓ should allow investors to deposit assets in the senior tranche when with in the user and tranche cap (2745ms)
        ✓ should not allow strategist to invest before the investAt time
        ✓ should not allow investors to deposit assets when exceeds usercap for the tranche
        ✓ should not allow investors to deposit assets in the live vault
        ✓ should allow investors to claim uninvested asset plus tranche tokens for the corresponding asset
        ✓ should not return tranche tokens on claim if enableTokens flag is disabled
        ✓ should allow investors to deposit ETH into Active vault only if the tranche is WETH
        ✓ should allow investors to claim ETH though the vault asset is WETH
        ✓ should allow performance fee to be set and changed by strategist on inactive vaults only
        ✓ should not allow performance fee to be more than a predetermined amount
        ✓ should be able to retrieve vault for specific vault index or a range of vault indexes
        ✓ should be able to retrieve vault for specific trancheToken address
        ✓ should be able to retrieve expected senior amount for the vault
        ✓ should be able to retrieve correct usercaps set in the vault
        ✓ should be able to retrieve all the correct details of vault investor like position, claimable and withdrawable balances, excess
        USDC-BOND vault behaviour - invest
          ✓ should allow strategist to invest - On invest one asset should go to 0 (12794ms)
          USDC-BOND vault behaviour - harvest and redeem
            ✓ should increase the amount of lp tokens in staking contract when harvest in strategy is called (44760ms)
            ✓ should not redeem if the slippage is too high (3367ms)
            ✓ should increase both sr and jr values on redeem (400ms)
            ✓ should calculate the correct amount of shares for given number of lp tokens
            ✓ should calculate the correct amount of lptokens for given number of shares
            ✓ should not compound if sushi rewards and second rewards are less than 10000
            ✓ should convert all the sushi to the end asset in path[0] in pool and all reward tokens to the end asset path[1] in the pool and invest into LP
            ✓ should be able to get more LP with better reward paths. eg., instead of [[sushi,DAI],[LDO,ETH,DAI,WSTETH]] we should be able to use [[sushi,DAI],[LDO,ETH,DAI]]
            ✓ should be able to compound with senior uninvested leftover
            ✓ should be able to compound with junior uninvested leftover
            ✓ should be able to compound with no uninvested leftover
            ✓ should be able to compound when tokenA is leftover after investing both rewards into LP
            ✓ should be able to compound when tokenB is leftover  after investing both rewards into LP
            USDC-BOND mid term deposits
              ✓ should allow investors to claim senior tranche tokens and excess (109ms)
             5) should allow investors to claim junior tranche tokens and excess
              ✓ should allow investors to trade tranche tokens (119ms)
              ✓ should allow investors to withdraw LP tokens (525ms)
              ✓ should allow investors to deposit LP in the Live vault and get back tranche tokens (554ms)
              ✓ should allow harvest and redeem if all LP is withdrawn
              ✓ should not allow investors to deposit LP when vault is not Live
              ✓ should allow investors to deposit ETH into Active vault only if the tranch is WETH
              ✓ should allow investors to withdraw ETH even though the balance in strategy is WETH
              ✓ should allow investors to withdraw LP in the correct ratio by depositing tranche tokens. Only correct ratio of tranche tokens burnt and remaining left in the contract
              ✓ should allow governace to set performanceFee collector for vault
              ✓ should increase the amount of lp tokens in staking contract when harvest in strategy is called (472ms)
              ✓ should allow redeem and burn tranche tokens (1097ms)
              USDC-BOND emergency withdraw from Uniswap
                ✓ should allow governace to withdraw LP from uniswap to strategy (42ms)
                ✓ should not allow any one other than governace to call multiexcall
                ✓ should return error if multicall fails
                ✓ should allow guardian role to rescue the tokens from strategy (47ms)
                ✓ should not allow anyone other than guardian role to rescue the LP tokens withdrawn from mcv2

  Claim
    ✓ Create vault (1373ms)
    ✓ Get some DAI
    ✓ Deposit on both sides (11687ms)
    ✓ Move to invest state (2627ms)
    ✓ Try to claim DAI (69ms)
    ✓ Try to claim ETH (70ms)
    ✓ Redeem funds (1897ms)
    ✓ Withdraw all funds (146ms)

  Create2
    ✓ create Vault (3587ms)

  Performance fees
    delayed Vault
      ✓ setup vault fixture (2810ms)
      ✓ create vault (1354ms)
      ✓ can only deposit after start time (1074ms)
      ✓ setup fee collector and performance fee
      ✓ deposits after start time (112ms)
      ✓ invest and redeem (292ms)

  Ondo
    metadata
      ✓ has a name
      ✓ has a symbol
    balanceOf
      ✓ grants to initial account
    delegateBySig
      ✓ reverts if the signatory is invalid (565ms)
      ✓ 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 (114ms)
    Ondo transfer disabled
      ✓ transfer reverts when transfers are disabled
      ✓ account with owner permission 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 (123ms)
      ✓ rescue tokens (72ms)
      ✓ stops the pause and restores functionality

  test reverts
    ✓ reverts (1164ms)

  RolloverVault
    basic round
      ✓ create pool (4907ms)
      ✓ revert: create rollover with invalid vault id
      ✓ revert: create rollver with invalid start time (1027ms)
      ✓ sucess: create rollover (3977ms)
      ✓ success: deposit senior asset (263ms)
      ✓ success: deposit junior asset (264ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (85ms)
      ✓ revert: add vault with invalid tranche assets (1108ms)
      ✓ revert: add vault with invalid start time (1036ms)
      ✓ success: adds another Vault to rollover tip (1112ms)
      ✓ success: migrate (286ms)
    round with single deposit
      ✓ create pool (3123ms)
      ✓ revert: create rollover with invalid vault id
      ✓ revert: create rollver with invalid start time (975ms)
      ✓ sucess: create rollover (2996ms)
      ✓ success: single deposit (95ms)
      ✓ success: deposit senior asset (263ms)
      ✓ success: deposit junior asset (264ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (77ms)
      ✓ revert: add vault with invalid tranche assets (1014ms)
      ✓ revert: add vault with invalid start time (1097ms)
      ✓ success: adds another Vault to rollover tip (1131ms)
      ✓ success: migrate (289ms)
    round with claim
```

```
      ✓ create pool (3067ms)
      ✓ revert: create rollover with invalid vault id
      ✓ revert: create rollver with invalid start time (965ms)
      ✓ sucess: create rollover (4968ms)
      ✓ success: single deposit (92ms)
      ✓ success: single deposit (89ms)
      ✓ success: deposit senior asset (265ms)
      ✓ success: deposit junior asset (268ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (82ms)
      ✓ revert: add vault with invalid tranche assets (1014ms)
      ✓ revert: add vault with invalid start time (2004ms)
      ✓ success: adds another Vault to rollover tip (1099ms)
      ✓ success: migrate (282ms)
      ✓ success: claim user tokens and excess (57ms)
      ✓ success: claim user tokens and excess (55ms)
    round with withdraw
      ✓ create pool (3166ms)
      ✓ revert: create rollover with invalid vault id
      ✓ revert: create rollver with invalid start time (967ms)
      ✓ sucess: create rollover (1976ms)
      ✓ success: single deposit (91ms)
      ✓ success: single deposit (89ms)
      ✓ success: deposit senior asset (273ms)
      ✓ success: deposit junior asset (266ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (77ms)
      ✓ revert: add vault with invalid tranche assets (2070ms)
      ✓ revert: add vault with invalid start time (1120ms)
      ✓ success: adds another Vault to rollover tip (1201ms)
      ✓ success: migrate (278ms)
      ✓ revert: withdraw zero amount
      ✓ revert: withdraw more than shares (71ms)
      ✓ success: withdraw (124ms)
      ✓ revert: withdraw zero amount
      ✓ revert: withdraw more than shares (39ms)
      ✓ success: withdraw (119ms)
    round with depositLp
      ✓ create pool (3524ms)
      ✓ revert: create rollover with invalid vault id (99ms)
      ✓ revert: create rollver with invalid start time (1089ms)
      ✓ sucess: create rollover (2042ms)
      ✓ success: single deposit (88ms)
      ✓ success: single deposit (91ms)
      ✓ success: deposit senior asset (270ms)
      ✓ success: deposit junior asset (267ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (77ms)
      ✓ revert: add vault with invalid tranche assets (990ms)
      ✓ revert: add vault with invalid start time (1023ms)
      ✓ success: adds another Vault to rollover tip (1084ms)
      ✓ success: migrate (274ms)
      ✓ revert: withdraw zero amount
      ✓ revert: withdraw more than shares
      ✓ success: withdraw (119ms)
      ✓ revert: withdraw zero amount
      ✓ revert: withdraw more than shares
      ✓ success: withdraw (114ms)
      ✓ success: deposit LP tokens mid-duration with signer 7 (674ms)
    round with withdrawLp
      ✓ create pool (3072ms)
      ✓ revert: create rollover with invalid vault id
      ✓ revert: create rollver with invalid start time (2013ms)
      ✓ sucess: create rollover (2028ms)
      ✓ success: single deposit (83ms)
      ✓ success: single deposit (92ms)
      ✓ success: deposit senior asset (267ms)
      ✓ success: deposit junior asset (267ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (74ms)
      ✓ revert: add vault with invalid tranche assets (997ms)
      ✓ revert: add vault with invalid start time (1038ms)
      ✓ success: adds another Vault to rollover tip (1098ms)
      ✓ success: migrate (276ms)
      ✓ revert: withdraw zero amount
      ✓ revert: withdraw more than shares (40ms)
      ✓ success: withdraw (113ms)
      ✓ revert: withdraw zero amount
      ✓ revert: withdraw more than shares
      ✓ success: withdraw (119ms)
      ✓ success: deposit LP tokens mid-duration with signer 7 (694ms)
      ✓ success: deposit LP tokens mid-duration with signer 8 (693ms)
      ✓ success: withdraw LP mid-duration with signer 7 (221ms)
      ✓ success: withdraw LP mid-duration with signer 8 (225ms)
    deposit after claim
      ✓ create pool (5586ms)
      ✓ revert: create rollover with invalid vault id
      ✓ revert: create rollver with invalid start time (980ms)
      ✓ sucess: create rollover (1985ms)
      ✓ success: single deposit (92ms)
      ✓ success: single deposit (88ms)
      ✓ success: deposit senior asset (271ms)
      ✓ success: deposit junior asset (268ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (73ms)
      ✓ revert: add vault with invalid tranche assets (1000ms)
      ✓ revert: add vault with invalid start time (1007ms)
      ✓ success: adds another Vault to rollover tip (1066ms)
      ✓ success: migrate (277ms)
      ✓ success: claim user tokens and excess (50ms)
      ✓ success: claim user tokens and excess (53ms)
      ✓ success: single deposit (88ms)
      ✓ success: single deposit (91ms)
      ✓ success: deposit senior asset (313ms)
      ✓ success: deposit junior asset (297ms)
      ✓ revert: deposit with invalid rollover id
      ✓ revert: deposit exceeds senior user cap (79ms)
      ✓ revert: add vault with invalid tranche assets (994ms)
      ✓ revert: add vault with invalid start time (1000ms)
      ✓ success: adds another Vault to rollover tip (1089ms)
      ✓ success: migrate (638ms)
      ✓ success: claim user tokens and excess (271ms)
      ✓ success: claim user tokens and excess (55ms)

  StakingPools
    ✓ should set correct state variables (442ms)
    ✓ should allow emergency withdraw (328ms)
    ✓ should give out ondos only after farming time (2036ms)
    ✓ should not distribute ONDOs if no one deposit (1136ms)
    ✓ should distribute ondos properly for each staker (2475ms)
    ✓ should give proper ONDOs allocation to each pool (1578ms)
    ✓ should stop giving bonus ONDOs after the bonus period ends (2079ms)
    ✓ should track minimumOndoRequiredBalance properly (1725ms)

  SushiStrategyLP
    ✓ pool add and update reverts (13238ms)
    batchCreate
      ✓ create Vault: sell senior for leveraged junior returns (1013ms)
      ✓ create Vault: sell all junior to partially cover senior (993ms)
      ✓ create Vault: sell some junior to cover senior (1050ms)
    batchDeposit
      ✓ deposit senior asset: sell senior for leveraged junior returns (143ms)
      ✓ deposit junior asset: sell senior for leveraged junior returns (146ms)
      ✓ deposit senior asset: sell all junior to partially cover senior (145ms)
      ✓ deposit junior asset: sell all junior to partially cover senior (137ms)
      ✓ deposit senior asset: sell some junior to cover senior (228ms)
      ✓ deposit junior asset: sell some junior to cover senior (505ms)
    increase time
      ✓ increase time
    batchInvest
      ✓ invest assets: sell senior for leveraged junior returns (7377ms)
      ✓ invest assets: sell all junior to partially cover senior (769ms)
      ✓ invest assets: sell some junior to cover senior (353ms)
    increase time
      ✓ increase time
    batchMidDeposit
      ✓ deposit LP tokens mid-duration with signer 4: sell senior for leveraged junior returns (1256ms)
      ✓ deposit LP tokens mid-duration with signer 4: sell all junior to partially cover senior (734ms)
      ✓ deposit LP tokens mid-duration with signer 4: sell some junior to cover senior (742ms)
    batchMidDeposit
      ✓ deposit LP tokens mid-duration with signer 5: sell senior for leveraged junior returns (741ms)
      ✓ deposit LP tokens mid-duration with signer 5: sell all junior to partially cover senior (744ms)
      ✓ deposit LP tokens mid-duration with signer 5: sell some junior to cover senior (738ms)
    increase time
      ✓ increase time
    batchHarvest
      ✓ harvest rewards: sell senior for leveraged junior returns (265ms)
      ✓ harvest rewards: sell all junior to partially cover senior (258ms)
      ✓ harvest rewards: sell some junior to cover senior (267ms)
    batchMidWithdraw
      ✓ withdraw LP mid-duration with signer 4: sell senior for leveraged junior returns (180ms)
      ✓ withdraw LP mid-duration with signer 4: sell all junior to partially cover senior (175ms)
      ✓ withdraw LP mid-duration with signer 4: sell some junior to cover senior (170ms)
    batchMidDeposit
      ✓ deposit LP tokens mid-duration with signer 6: sell senior for leveraged junior returns (744ms)
      ✓ deposit LP tokens mid-duration with signer 6: sell all junior to partially cover senior (732ms)
```

```
     ✓ deposit LP tokens mid-duration with signer 6: sell some junior to cover senior (737ms)
  batchHarvest
     ✓ harvest rewards: sell senior for leveraged junior returns (257ms)
     ✓ harvest rewards: sell all junior to partially cover senior (265ms)
     ✓ harvest rewards: sell some junior to cover senior (434ms)
  increase time
     ✓ increase time
  batchClaim
     ✓ claim tranche tokens: sell senior for leveraged junior returns (405ms)
     ✓ claim tranche tokens: sell all junior to partially cover senior (172ms)
     ✓ claim tranche tokens: sell some junior to cover senior (183ms)
  redeem and withdrawal: sell senior for leveraged junior returns
     ✓ redeem LP after fee accrual (508ms)
     ✓ withdraw received amounts (554ms)
  redeem and withdrawal: sell all junior to partially cover senior
     ✓ redeem LP after fee accrual (500ms)
     ✓ withdraw received amounts (536ms)
  redeem and withdrawal: sell some junior to cover
     ✓ redeem LP after fee accrual (487ms)
     ✓ withdraw received amounts (572ms)
  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 (1082ms)
       ✓ create Vault: sell all junior to partially cover senior (985ms)
       ✓ create Vault: sell some junior to cover senior (1047ms)
    batchDeposit
       ✓ deposit senior asset: sell senior for leveraged junior returns (2496ms)
       ✓ deposit junior asset: sell senior for leveraged junior returns (144ms)
       ✓ deposit senior asset: sell all junior to partially cover senior (107ms)
       ✓ deposit junior asset: sell all junior to partially cover senior (137ms)
       ✓ deposit senior asset: sell some junior to cover senior (106ms)
       ✓ deposit junior asset: sell some junior to cover senior (149ms)
    increase time
       ✓ increase time
    batchInvest
       ✓ invest assets: sell senior for leveraged junior returns (1032ms)
       ✓ invest assets: sell all junior to partially cover senior (324ms)
       ✓ invest assets: sell some junior to cover senior (326ms)
    increase time
       ✓ increase time
    batchMidDeposit
       ✓ deposit LP tokens mid-duration with signer 4: sell senior for leveraged junior returns (732ms)
       ✓ deposit LP tokens mid-duration with signer 4: sell all junior to partially cover senior (724ms)
       ✓ deposit LP tokens mid-duration with signer 4: sell some junior to cover senior (734ms)
    batchMidDeposit
       ✓ deposit LP tokens mid-duration with signer 5: sell senior for leveraged junior returns (723ms)
       ✓ deposit LP tokens mid-duration with signer 5: sell all junior to partially cover senior (734ms)
       ✓ deposit LP tokens mid-duration with signer 5: sell some junior to cover senior (732ms)
    increase time
       ✓ increase time
    batchHarvest
       ✓ harvest rewards: sell senior for leveraged junior returns (228ms)
       ✓ harvest rewards: sell all junior to partially cover senior (483ms)
       ✓ harvest rewards: sell some junior to cover senior (406ms)
    batchMidWithdraw
       ✓ withdraw LP mid-duration with signer 4: sell senior for leveraged junior returns (178ms)
       ✓ withdraw LP mid-duration with signer 4: sell all junior to partially cover senior (171ms)
       ✓ withdraw LP mid-duration with signer 4: sell some junior to cover senior (170ms)
    batchMidDeposit
       ✓ deposit LP tokens mid-duration with signer 6: sell senior for leveraged junior returns (720ms)
       ✓ deposit LP tokens mid-duration with signer 6: sell all junior to partially cover senior (727ms)
       ✓ deposit LP tokens mid-duration with signer 6: sell some junior to cover senior (725ms)
    batchHarvest
       ✓ harvest rewards: sell senior for leveraged junior returns (236ms)
       ✓ harvest rewards: sell all junior to partially cover senior (226ms)
       ✓ harvest rewards: sell some junior to cover senior (229ms)
    increase time
       ✓ increase time
    batchClaim
       ✓ claim tranche tokens: sell senior for leveraged junior returns (175ms)
       ✓ claim tranche tokens: sell all junior to partially cover senior (166ms)
       ✓ claim tranche tokens: sell some junior to cover senior (189ms)
    redeem and withdraw
       ✓ redeem LP after fee accrual (4314ms)

SushiStakingV2Strategy - Alchemix helper
  alchemix vault lifecycle
     ✓ get assets (312ms)
     ✓ add pool (41ms)
     ✓ create vaults (2513ms)
     ✓ deposit assets and fast-forward (6346ms)
     ✓ invest ALCX/ETH vault (8542ms)
     ✓ invest ETH/ALCX vault (886ms)
     ✓ harvest (371ms)
     ✓ harvest (372ms)
     ✓ harvest (384ms)
     ✓ harvest (367ms)
     ✓ harvest (375ms)
     ✓ harvest (381ms)
     ✓ redeem ALCX/ETH vault (1600ms)
     ✓ redeem ETH/ALCX vault (214ms)

SushiStakingV2Strategy - WETH-ALCX helper
  6) "before all" hook in "SushiStakingV2Strategy - WETH-ALCX helper"

SushiStakingV2Strategy - WETH-BIT multiple vaults
  WETH-BIT - invest first vault
    7) "before all" hook: Setup for "should allow registry strategist to create pool in the strategy"

SushiStakingV2Strategy - WETH-BIT rescue from all pair
  WETH-BIT - deposit
    8) "before all" hook: Setup for "should allow registry strategist to create pool in the strategy"

SushiStakingV2Strategy - WETH-BIT helper
  9) "before all" hook in "SushiStakingV2Strategy - WETH-BIT helper"

SushiStakingV2Strategy - ETH-YGG-JR-LOSE-ALL helper
  10) "before all" hook in "SushiStakingV2Strategy - ETH-YGG-JR-LOSE-ALL helper"

SushiStakingV2Strategy - ETH-YGG helper
  11) "before all" hook in "SushiStakingV2Strategy - ETH-YGG helper"

TrancheToken
   ✓ spin up tranche tokens on Vault creation (1001ms)
   ✓ deposit base assets during Deposit phase (145ms)
   ✓ claim tokens only after investment (236ms)
   ✓ approve and transferFrom (38ms)

Uni
   ✓ test create mock (4547ms)
   ✓ test uniPull (650ms)

BondStrategy - USDC-BOND helper
  12) "before all" hook in "BondStrategy - USDC-BOND helper"

BondStrategy - USDC-BOND helper
  13) "before all" hook in "BondStrategy - USDC-BOND helper"

AllPairVault
  delayed Vault
     ✓ setup vault fixture (541ms)
     ✓ create vault (1030ms)
     ✓ can only deposit after start time (1054ms)
     ✓ can get multiple vaults back from getVault (2899ms)
     ✓ deposits after start time (100ms)
     ✓ deposits exceed user cap
     ✓ deposits exceed tranche cap (159ms)
     ✓ can't deposit after investment
  withdraw midterm LP deposit
     ✓ setup vault fixture (2253ms)
     ✓ create vault (2244ms)
     ✓ deposit senior asset (173ms)
     ✓ deposit junior asset (170ms)
     ✓ deposits as expected
     ✓ cannot invest too early
     ✓ invest assets (160ms)
     ✓ can't deposit or withdraw midterm unless tranche tokens are enabled
     ✓ deposit LP tokens mid-duration with signer 7 (460ms)
     ✓ withdraws as expected after depositing LP without claiming (322ms)
     ✓ withdraw received amounts (268ms)
  sell senior for leveraged junior returns
     ✓ setup vault fixture (535ms)
     ✓ create vault (2080ms)
     ✓ deposit senior asset (171ms)
     ✓ deposit junior asset (171ms)
     ✓ deposits as expected
     ✓ cannot invest too early
     ✓ gets Vault by tranche token addresses
     ✓ gets all Vaults
     ✓ invest assets (152ms)
     ✓ claim tranche tokens and excess (256ms)
     ✓ withdraw LP mid-duration from original deposits (359ms)
     ✓ deposit LP tokens mid-duration with signer 6 (442ms)
     ✓ deposit LP tokens mid-duration with signer 7 (469ms)
```

```
      ✓ withdraw LP mid-duration with signer 6 (111ms)
      ✓ withdraw LP mid-duration with signer 7 (111ms)
      ✓ redeem LP after fee accrual (312ms)
      ✓ withdraw received amounts (423ms)
    sell all junior to partially cover senior
      ✓ setup vault fixture (2588ms)
      ✓ create vault (1106ms)
      ✓ deposit senior asset (173ms)
      ✓ deposit junior asset (165ms)
      ✓ deposits as expected
      ✓ cannot invest too early
      ✓ invest assets (152ms)
      ✓ claim tranche tokens and excess (262ms)
      ✓ withdraw LP mid-duration from original deposits (359ms)
      ✓ deposit LP tokens mid-duration with signer 6 (437ms)
      ✓ deposit LP tokens mid-duration with signer 7 (470ms)
      ✓ withdraw LP mid-duration with signer 6 (108ms)
      ✓ withdraw LP mid-duration with signer 7 (108ms)
      ✓ redeem LP after fee accrual (282ms)
      ✓ withdraw received amounts (422ms)
    sell some junior to cover senior
      ✓ setup vault fixture (3033ms)
      ✓ create vault (1201ms)
      ✓ deposit senior asset (173ms)
      ✓ deposit junior asset (175ms)
      ✓ deposits as expected
      ✓ cannot invest too early
      ✓ invest assets (154ms)
      ✓ claim tranche tokens and excess (258ms)
      ✓ withdraw LP mid-duration from original deposits (356ms)
      ✓ deposit LP tokens mid-duration with signer 6 (450ms)
      ✓ deposit LP tokens mid-duration with signer 7 (469ms)
      ✓ withdraw LP mid-duration with signer 6 (113ms)
      ✓ withdraw LP mid-duration with signer 7 (109ms)
      ✓ redeem LP after fee accrual (275ms)
      ✓ withdraw received amounts (427ms)


  411 passing (8m)
  13 failing

  1) masterchefv2 dual incentive pool
       mcv2 vault lifecycle
         get LP from the sushiswap pool and deposit:
     AssertionError: Expected "2" to be equal 1
      at Context.<anonymous> (test/alchemix.spec.ts:273:38)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)

  2) masterchefv2 dual incentive pool
       mcv2 vault lifecycle
         withdraw LP from ALCX/ETH vault:
     AssertionError: Expected "2" to be equal 1
      at Context.<anonymous> (test/alchemix.spec.ts:306:39)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)

  3) masterchefv2 dual incentive pool
       mcv2 vault lifecycle
         deposit LP in ALCX/ETH vault:
     AssertionError: Expected "2" to be equal 1
      at Context.<anonymous> (test/alchemix.spec.ts:317:38)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)

  4) EdenStrategy - WETH-EDEN helper
       "before all" hook in "EdenStrategy - WETH-EDEN helper":
     Error: call revert exception (method="decimals()", errorArgs=null, errorName=null, errorSignature=null, reason=null, code=CALL_EXCEPTION, version=abi/5.4.1)
      at Logger.makeError (node_modules/@ethersproject/logger/src.ts/index.ts:225:28)
      at Logger.throwError (node_modules/@ethersproject/logger/src.ts/index.ts:237:20)
      at Interface.decodeFunctionResult (node_modules/@ethersproject/abi/src.ts/interface.ts:425:23)
      at Contract.<anonymous> (node_modules/@ethersproject/contracts/src.ts/index.ts:332:44)
      at step (node_modules/@ethersproject/contracts/lib/index.js:48:23)
      at Object.next (node_modules/@ethersproject/contracts/lib/index.js:29:53)
      at fulfilled (node_modules/@ethersproject/contracts/lib/index.js:20:58)

  5) BondStrategy - USDC-BOND helper
       USDC-BOND - deployment validation
         USDC-BOND - vault lifecycle
           USDC-BOND vault behaviour - invest
             USDC-BOND vault behaviour - harvest and redeem
               USDC-BOND mid term deposits
                 should allow investors to claim junior tranche tokens and excess:
     AssertionError: expected undefined to equal 0
      at Context.<anonymous> (test/behaviour/auniswap/vaultMidtermdeposits.ts:170:48)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)

  6) SushiStakingV2Strategy - WETH-ALCX helper
       "before all" hook in "SushiStakingV2Strategy - WETH-ALCX helper":
     Error: VM Exception while processing transaction: reverted with reason string 'UniswapV2Router: EXCESSIVE_INPUT_AMOUNT'
      at <UnrecognizedContract>.<unknown> (0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)

  7) SushiStakingV2Strategy - WETH-BIT multiple vaults
       WETH-BIT - invest first vault
         "before all" hook: Setup for "should allow registry strategist to create pool in the strategy":
     Error: Transaction reverted without a reason string
      at <UnrecognizedContract>.<unknown> (0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)
      at EthModule._sendTransactionAndReturnHash (node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1494:18)
      at HardhatNetworkProvider.request (node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:108:18)
      at EthersProviderWrapper.send (node_modules/@nomiclabs/hardhat-ethers/src/internal/ethers-provider-wrapper.ts:13:20)

  8) SushiStakingV2Strategy - WETH-BIT rescue from all pair
       WETH-BIT - deposit
         "before all" hook: Setup for "should allow registry strategist to create pool in the strategy":
     Error: Transaction reverted without a reason string
      at <UnrecognizedContract>.<unknown> (0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)

  9) SushiStakingV2Strategy - WETH-BIT helper
       "before all" hook in "SushiStakingV2Strategy - WETH-BIT helper":
     Error: Transaction reverted without a reason string
      at <UnrecognizedContract>.<unknown> (0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)

  10) SushiStakingV2Strategy - ETH-YGG-JR-LOSE-ALL helper
       "before all" hook in "SushiStakingV2Strategy - ETH-YGG-JR-LOSE-ALL helper":
     Error: call revert exception (method="decimals()", errorArgs=null, errorName=null, errorSignature=null, reason=null, code=CALL_EXCEPTION, version=abi/5.4.1)
      at Logger.makeError (node_modules/@ethersproject/logger/src.ts/index.ts:225:28)
      at Logger.throwError (node_modules/@ethersproject/logger/src.ts/index.ts:237:20)
      at Interface.decodeFunctionResult (node_modules/@ethersproject/abi/src.ts/interface.ts:425:23)
      at Contract.<anonymous> (node_modules/@ethersproject/contracts/src.ts/index.ts:332:44)
      at step (node_modules/@ethersproject/contracts/lib/index.js:48:23)
      at Object.next (node_modules/@ethersproject/contracts/lib/index.js:29:53)
      at fulfilled (node_modules/@ethersproject/contracts/lib/index.js:20:58)

  11) SushiStakingV2Strategy - ETH-YGG helper
       "before all" hook in "SushiStakingV2Strategy - ETH-YGG helper":
     Error: call revert exception (method="decimals()", errorArgs=null, errorName=null, errorSignature=null, reason=null, code=CALL_EXCEPTION, version=abi/5.4.1)
      at Logger.makeError (node_modules/@ethersproject/logger/src.ts/index.ts:225:28)
      at Logger.throwError (node_modules/@ethersproject/logger/src.ts/index.ts:237:20)
      at Interface.decodeFunctionResult (node_modules/@ethersproject/abi/src.ts/interface.ts:425:23)
      at Contract.<anonymous> (node_modules/@ethersproject/contracts/src.ts/index.ts:332:44)
```



```
      at step (node_modules/@ethersproject/contracts/lib/index.js:48:23)
      at Object.next (node_modules/@ethersproject/contracts/lib/index.js:29:53)
      at fulfilled (node_modules/@ethersproject/contracts/lib/index.js:20:58)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)

  12) BondStrategy - USDC-BOND helper
       "before all" hook in "BondStrategy - USDC-BOND helper":
     Error: VM Exception while processing transaction: reverted with reason string 'UniswapV2Router: EXCESSIVE_INPUT_AMOUNT'
      at <UnrecognizedContract>.<unknown> (0x7a250d5630b4cf539739df2c5dacb4c659f2488d)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)

  13) BondStrategy - USDC-BOND helper
       "before all" hook in "BondStrategy - USDC-BOND helper":
     Error: VM Exception while processing transaction: reverted with reason string 'UniswapV2Router: EXCESSIVE_INPUT_AMOUNT'
      at <UnrecognizedContract>.<unknown> (0x7a250d5630b4cf539739df2c5dacb4c659f2488d)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:93:5)
      at runNextTicks (internal/process/task_queues.js:62:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)
```

Code Coverage

Some tests are failing and thus it is not possible to reliably evaluate the code coverage at this stage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	94.36	71.94	90.2	94.29	
AllPairVault.sol	96.69	72.12	97.78	96.75	… 9,1170,1171
OndoRegistryClient.sol	100	100	100	100	
OndoRegistryClientInitializable.sol	92.86	62.5	85.71	93.33	51
Registry.sol	64.29	50	66.67	65.52	… 147,148,160
RolloverVault.sol	96.68	80.3	100	96.7	… 681,686,735
SampleFeeCollector.sol	100	50	100	100	
TrancheToken.sol	71.43	25	63.64	68.75	… 100,109,118
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	
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/	67.37	43.95	67.12	67.79	
ASushiswapStrategy.sol	0	0	0	0	… 445,446,447
AUniswapStrategy.sol	65.56	39.29	63.64	66.67	… 475,476,484
AlchemixLPStrategy.sol	95.39	56.52	100	95.45	… 473,505,506
BasePairLPStrategy.sol	77.78	50	100	78.95	85,86,87,88
BondStrategy.sol	100	77.78	100	100	
DopexStrategy.sol	**0**	**0**	**0**	**0**	**… 186,190,192**
EdenStrategy.sol	21.28	25	12.5	21.28	… 179,184,186

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
SushiStakingV2Strategy.sol	65.57	41.89	69.57	66.19	… 816,817,818
SushiStrategyLP.sol	100	66.67	100	100	
UniswapStrategy.sol	91.67	68.75	93.33	91.8	… 373,374,375
contracts/strategies/helpers/	70	25	66.67	70	
AlchemixUserReward.sol	70	25	66.67	70	30,31,32
contracts/test/	40.38	100	36.84	40.38	
ERC20Mock.sol	40	100	75	40	18,20,21
ForceSendEth.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	… 43,44,45,46
contracts/test/barnbridge/	100	100	100	100	
MockStaking.sol	100	100	100	100	
contracts/tokens/	74.16	54.17	82.35	74.86	
Ondo.sol	58.95	47.73	76.19	60.42	… 292,295,379
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/barnbridge/	100	100	100	100	
Staking.sol	100	100	100	100	
YieldFarmLP.sol	100	100	100	100	
contracts/vendor/dopex/	100	100	100	100	
IStakingRewards.sol	100	100	100	100	
contracts/vendor/eden/	100	100	100	100	
IRewardsManager.sol	100	100	100	100	
contracts/vendor/sushiswap/	100	100	100	100	
IMasterChef.sol	100	100	100	100	
IMasterChefV2.sol	100	100	100	100	
IRewarder.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	**67.71**	**43.21**	**60.35**	**67.26**	

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

```
b7fb3be8abcaf9350d8ff0f6cbfd2e54bceb306d7287c0eb61f388a33ea207d9 ./StakingPools.sol
d0acfa5eadf123ed7066569a6c9db9d76d97053458d0728af517e27319fd86bd ./Ondo.sol
262c395ddf87a2a478930445380bc533ac0857d6f4724c891767e17c0fef1b2f ./OndoLibrary.sol
01343f283af0ea51d215feefa79c5a59b5e4432f29a2e6907fba1adf2b404a4e ./TrancheToken.sol
```

```
6c263bde545659f48051814be385874dca90acc11a4bcdbfa18f2beba20e5ac0 ./SampleFeeCollector.sol
365c205a33be504ee56729d962fadffbdf7c5c82f9d7dfbc94ac68353441e1a7 ./RolloverVault.sol
491d582b501cae87b72f58c2f5b98ee7435d1308e1afcd4b386e306815578f82 ./Registry.sol
a4ce7c250d5ca36d16e03f16f659424a0434dfa1c6c7241e407fd1e1ccc702b8 ./AllPairVault.sol
176a90318116f4051fe0d543d20aa192502a8dbd783d7fe1cb865f94ee110909 ./OndoRegistryClientInitializable.sol
15e64cc312a3d7ebaa2cbd4ad30bf15f380616bbe80e2a92e32d99828f142c5c ./OndoRegistryClient.sol
fe9fad57c5ddee134040f86e9c5434e0c337a138fff20ead12b1806fece3dc75 ./AlchemixUserReward.sol
9def31e71e84f9b14ba45290bb3dd9d8293c29f6d893bae1c5ddede80b7d7081 ./UniswapStrategy.sol
e2921400d0d44176395bfafa878892af391badad3a12a927104c76e95bbffb8b ./SushiStrategyLP.sol
fc7153b62f538e370f8bc4eb0a5524cb6e76c05f1fb870da8e7d02708f2378b7 ./SushiStakingV2Strategy.sol
4ffa161e757a3a0374936d78d47b7ce4b202c9db43c7fdb726a6ad852dc20867 ./EdenStrategy.sol
f940caa491e39d43818e653064d89f2199bc194265d1fe2bf66032627af3cc0c ./DopexStrategy.sol
b28ed5920f4e4a05dab38f4385a607a7a2f33d5da10699fd9eedd23da7a42659 ./BondStrategy.sol
5f27f6caa1f4b8a5ddd9f4dde0810feed79467782128be117755c27b4c2f59c9 ./AlchemixLPStrategy.sol
6193076e3f6fb2cb878b68310710618f2123250ed77b5fe9d4cd86f4aa066ac6 ./BasePairLPStrategy.sol
5e4bfc31dbbf3a9e9cbef91a76a3aec6efa4a6ee89cce28cf1e49e63ce0011b4 ./AUniswapStrategy.sol
765410b533d5a99c57aea49ffbefb8eb698b16df66095513a06b46be0d966843 ./ASushiswapStrategy.sol
8e487d567f6814e03d5c7a7d5cd94afb9573ee10bf8b8cb38b4e1e4ceb0205c2 ./QuickSwapLibrary.sol
46489c93eff8908a663110b3d16da8e8e1681d69d0d786b1df2e8003ccce25a8 ./QuickswapStrategyLP.sol
4d7d9fe5b5918e64550ff7b991dca72f86dadc491ae5b76dbf210ee8c092f86b ./PancakeSwapLibrary.sol
6655f3a6f1563db0fd3c12e364eae9f7ffd22a7bbb9ea403875ff1f2dd8b6453 ./PancakeStrategyLP.sol
311019158d3c5fc17c24462df46332f7e93c82f391142b34b6b08758857a0113 ./PancakeStrategy.sol
1f9a1232376a48efaa33173bde866ba332d4034a556712e6243c41ab4bcd6888 ./contracts/RolloverVault.sol
15e64cc312a3d7ebaa2cbd4ad30bf15f380616bbe80e2a92e32d99828f142c5c ./contracts/OndoRegistryClient.sol
9c46b41bb23ad9def04afffaa1766fa143b4d4b5c0f011bad6356bfb58de88e5 ./contracts/Registry.sol
aa6972e837edea97fd06e1a81eef4e535c52cb54e0182b041b6fb32b9be63a9b ./contracts/AllPairVault.sol
e8026853efba0286ab7efadd01b4b4ff3947bf3e7f88cbc094ec91e5916d7088 ./contracts/TrancheToken.sol
b7fb3be8abcaf9350d8ff0f6cbfd2e54bceb306d7287c0eb61f388a33ea207d9 ./contracts/tokens/StakingPools.sol
d0acfa5eadf123ed7066569a6c9db9d76d97053458d0728af517e27319fd86bd ./contracts/tokens/Ondo.sol
262c395ddf87a2a478930445380bc533ac0857d6f4724c891767e17c0fef1b2f ./contracts/libraries/OndoLibrary.sol
5b5c45cd2d6ab0286484c21e05fb2f92f0d661cece1d7ee43cb9f9839f7331fa ./contracts/strategies/UniswapStrategy.sol
765410b533d5a99c57aea49ffbefb8eb698b16df66095513a06b46be0d966843 ./contracts/strategies/ASushiswapStrategy.sol
05cf5ab589cb58db1cbef3571d65a786c41959a8bc93aeb71142fac371eb1d55 ./contracts/strategies/BondStrategy.sol
098606fe8f5487482b21f7f00723117d60226d56c52c42f27b10bb9f5d1627f3 ./contracts/strategies/SushiStakingV2Strategy.sol
5e4bfc31dbbf3a9e9cbef91a76a3aec6efa4a6ee89cce28cf1e49e63ce0011b4 ./contracts/strategies/AUniswapStrategy.sol
8c278ef208d263d197a1d6c5bd58396e510d1b2065ee583c2a701411e6c8a43f ./contracts/strategies/DopexStrategy.sol
6193076e3f6fb2cb878b68310710618f2123250ed77b5fe9d4cd86f4aa066ac6 ./contracts/strategies/BasePairLPStrategy.sol
50a42214be23c799465893152ec6131eece3745fb95c8a446efb0d3dee2ed5e8 ./contracts/strategies/AlchemixLPStrategy.sol
7fd125893cfc7b2b5006fc114920ba9a71195e09fda6fb42c9888b22f6e42a93 ./contracts/strategies/SushiStrategyLP.sol
89d39c13b5c702cc841c126d4ddbad9c1564bc501ae73f61c0f2d922f7c33a2a ./contracts/strategies/EdenStrategy.sol
35c9982ba76ec7f3b8deba76861d3307a5774424a2373b20664d89f8217451ba ./contracts/strategies/helpers/AlchemixUserReward.sol
```

Tests

```
f53f15453e843ef9d70317f9e0b312615c3457bedb3d558fe949d82d8768f039 ./test/create2.spec.ts
2adb4b543786993b2cd6e07b2f8e56a480b7b4da4444271f8c1842c29b1e8f36 ./test/sushi2.spec.weth-ygg.ts
e79f5713e8f75fa048c4362c9a12c67bd66ae6b8e67227450dec4a3c4bd89605 ./test/staking.spec.ts
7700a794145cfd15235ee1743f8fc5c09e4a240b7ea73c7483a52f2e65a15810 ./test/reverts.spec.ts
8c2e9ca7ea47c012968dc72e4a856f9040ef410435d23a0d7db9c4a33ad4f3b7 ./test/sushi2.spec.weth-bit-rescue-from-allpair.ts
989e36c091ee53c818598103ffde2c979770f56ca36f5c0adaaaa3efe24b6d71 ./test/sushi2.spec.weth-ygg-jnrsLose.ts
0193a90e7c513149a5731bc6e5732df679e180a72d5f594216d59f004954ba0e ./test/sushi2.spec.weth-bit.ts
19681bfbafe32c7161ea18af32ba8cd8b948e8ab5df161d9e52bd6044a0cd42c ./test/uni.spec.usdt-bond-multivaults.ts
b0e250667999bf0bed33db084e6a001fd1e6e6b0df3eb38d4ad79537e76e689d ./test/uni.spec.usdt-bond-jrs-gain.ts
9ef67da498823f00c0a8a45b8042a6a5ab42455ea8ba3f5be462ffcbe5838124 ./test/fee.spec.ts
6e4a5170ee1dfb39ee6d0d0f081555034c300ef0209ad5552c9ef405ca687796 ./test/ondo.spec.ts
632ea85d33b6cf9f011e6e1ef462cb66c2c4965ae44ff082f5ade451ae1f1c0e ./test/uni.spec.ts
17afdb8ff44c2183959ead7d20be172a8e6c0a156a27b4445ad8855a4c36894b ./test/token.spec.ts
a1e740005618534bf1758c6cfe82ae0ff9f1845c9e83c510b880cf28d22fedf7 ./test/rollover.spec.ts
c4c1079f34f3c223e75f07296ec3f604aa612c0930a8c7adeda26c4b6589f095 ./test/uni.spec.usdt-bond.ts
d3c37dba547ba56afbac2045693ec627bfc84ace8b9aee69ffd7d284d3735936 ./test/sushi2.spec.weth-bit-multiple-vaults.ts
0194112e64af2b47636a5717a770d82de4bbca6ba6e23f263796b75258bf3c53 ./test/claim.spec.ts
ec87cdb2058a3cc3071a4546b55dea0b6b66372dd0414b01e4e606b484e035b1 ./test/vault.spec.ts
```

```
654a1bd43af19f69a41858629489e90a037ff9534dddc427d6c68eb9034b9df4 ./test/auni.spec.usdt-bond.ts
d8d0238f8f2a486e94b822c331e381d943736895bda6dae6ff88be1d9c6b47a8 ./test/asushi.spec.weth-eden.ts
d8502b60cea7501fdc780ae30f8a26da8cddf03cc00bbc1153c5d0c334cc375e ./test/registry.spec.ts
91e20259239d37e6cab96e800ac07781d612d95d7ec85964c09e6536e64a329b ./test/sushi2.spec.ts
78fb7c7d8a65242a1867edcf91796aae53530290c845712df8c03885c9c077e9 ./test/alchemix.spec.ts
9c93d37cabeab06ccdb89f72bab7e2c73ef28c261516966ae0c5c1bce8f13cd8 ./test/sushi-lp.spec.ts
438de978ff6d4cf5452eb6a68917d7a31a47b0ecfc238a0a8d56312a625f76c5 ./test/rescue.spec.ts
96ef1266ddbe8f0f67e1942d7e8c4946d3be230c6939467cb1a4b805f5c1eaa7 ./test/rollover.spec-forked.ts
96acb6f7f02998f39f382f5c42d67ecbc3ee467a8561561c1b946c5d59985711 ./test/sushi2.spec.weth-alcx.ts
a120f177affd17ce416947cc181ccd35457854e945122185fc811e7347a3c3a1 ./test/behaviour/vaultHarvestAndRedeemMultiVaultsSimple.ts
f2cead3facc2a0cdb0961a4e63cc177b12003cef76cd9f88db436b0b51d5bd01 ./test/behaviour/emergencyRescueToSigner.ts
adcea2615c31632bb9890e6b1ed19f8343a1c307aff1158e9d6ab47d7e2fcd1a ./test/behaviour/vaultMidtermdeposits.ts
28795da284d4b6bf636dee6d1e1b5897cf59f4818c21e37a580ebe29608015cc ./test/behaviour/vaultInvestmentLifecycle.ts
d38f5b63d71924e6962dfbd332813f4f093e8d0334809e27d5e9767afbcafe38 ./test/behaviour/vaultInvestmentSimple.ts
7c6b5a1962fedad466b312b4203b9588068806d7a88e4b233082a1edf6c96116 ./test/behaviour/vaultRollovers.ts
09cb79ef65922fa89a000e7dd641aaf5b91fd1df61599ffce3963926cbf3fbb9 ./test/behaviour/vaultLifecycle.ts
275d4088de32d9bcdd182ec3cee7c7ebaa0efdf5cdeb0e79d22a95befd69e4c9 ./test/behaviour/vaultInvestViaCallFunction.ts
f10abaa7b7a24e4f7c7f6ff76d93d4d03849a3e92a7d4201f4c8f58e0de7adca ./test/behaviour/emergencyRescueToStrategy.ts
9e459bf35587c33097c27e824f702d25c4b2d78af1f8ac2bfedfaaee8611496e ./test/behaviour/vaultHarvestAndRedeem.ts
945c6b3ee037a30b17c49fc3f9146934b9789916d3580a84a2735b897c91633b ./test/behaviour/auniswap/invest.ts
ce5b53b2d5b0e2910a9903c21e6fe9b30eb0464423d4ed6f5083089ff1da037e ./test/behaviour/auniswap/vaultMidtermdeposits.ts
b7ca16a52a8996ec35280135914c3488f0f9c6a78842f44581a40d0c3b5bf84d ./test/behaviour/auniswap/uniStrategySetup.ts
852d50384f83cdf9f93336c33933a8670d784421db7ed0e8ae0b16cdbd6b8d54 ./test/behaviour/auniswap/vaultLifecycle.ts
9bddd6fb998581bbe7214fdf61b72be384f6bfe566b915bd71ea13d4e1d73341 ./test/behaviour/auniswap/vaultInvestment.ts
b16d1ef54e0054ae8832e0661fb4c72d8723f2ac76b4b543377a2ce606a46bb9 ./test/behaviour/auniswap/emergencyRescueToStrategy.ts
5851859ffb174551f1144dcf50c5b74025242570018e01aabbde726bd585562c ./test/behaviour/auniswap/vaultHarvestAndRedeem.ts
945c6b3ee037a30b17c49fc3f9146934b9789916d3580a84a2735b897c91633b ./test/behaviour/asushiswap/invest.ts
c2860058dbb342749a7b8ee403a964fe6871837f4fd6fad67d59fa487b197230 ./test/behaviour/asushiswap/vaultMidtermdeposits.ts
91e9d6df5314e71d3aad744ccef76bcc3e9e7911a7ef24109f2c4f408e7bb54f ./test/behaviour/asushiswap/uniStrategySetup.ts
852d50384f83cdf9f93336c33933a8670d784421db7ed0e8ae0b16cdbd6b8d54 ./test/behaviour/asushiswap/vaultLifecycle.ts
9bddd6fb998581bbe7214fdf61b72be384f6bfe566b915bd71ea13d4e1d73341 ./test/behaviour/asushiswap/vaultInvestment.ts
55f8ad322bc1ad9b6146eac4a4fbbda73aa8e2ded5e73ef4a818ab8042634ace ./test/behaviour/asushiswap/emergencyRescueToStrategy.ts
5851859ffb174551f1144dcf50c5b74025242570018e01aabbde726bd585562c ./test/behaviour/asushiswap/vaultHarvestAndRedeem.ts
a7d7c451e51bd007c6a9956e2c01c16e4192a95af70684502f275f014604163b ./test/behaviour/uniswap/invest.ts
e507350f2d10504d6ddad65fb0b85cb4b06f78499b8ef49cd66edc27594982d3 ./test/behaviour/uniswap/vaultMidtermdeposits.ts
3d16c88e2dc51dd619ef42baab17bc323d35bdea6860036cc379ccd7f8142f82 ./test/behaviour/uniswap/vaultPathChangesRedeem.ts
4f37b82a75bc131d05c33ab387a883f0ffd004c6264f2c0e04820e9d3961f4e2 ./test/behaviour/uniswap/multivaultLifecycle.ts
3ece72a24adc556fdba395d2dbca705bfaa47585c74afc82c07d1928a4e13eaa ./test/behaviour/uniswap/uniStrategySetup.ts
2ac78115e9513a637c2efecfa06248a3d31b4f084ad68b26a5470ccd3390b1c2 ./test/behaviour/uniswap/vaultRedeem.ts
65432da3bd3b94a26ee8396205322706aa84c23c455e3fe8d1350923430d9fe2 ./test/behaviour/uniswap/vaultLifecycle.ts
85397c8642108c2fac2204994d4c3c3f2e6cba40d8247f7fa896852f949e42c4 ./test/behaviour/uniswap/vaultInvestment.ts
1a657fcdbafda3af377e3e82a13a554b8899f7f49153f8c109c6b9cc9fb0a084 ./test/behaviour/uniswap/emergencyRescueToStrategy.ts
295e2a0d7763e85150d6eaa8bcf1779b6e5b42e760b0c1f083ce02b105d85657 ./test/utils/DebugSigner.ts
5db8a0062c5cc308c9114d7bfd4de63a940a3c0cbd0ea278e218d6f56f3aa90b ./test/utils/addresses.ts
fa43c22f095ba4b2eb52dea637f35cf59bd86ef5599492e889f041e2c3237f56 ./test/utils/gen-utils.ts
9069b3f8a3a137776ea906d8b81b0a7a0cf9338231b7a4947d98d7648736bb54 ./test/utils/bignumberhax.ts
6f603be8959b57dd21f708ec298728c27ac43521e19461bedf9a12bb9ae25c0e ./test/utils/vault.ts
b14c977b9e56be27600fe7f0ec37a64bcc0f0f47e4251645a7844b4bc0b74da9 ./test/utils/rollover-forkednw.ts
66e8bc2c94fba576664f9b3835e7622f32c416aa20b619e1e6d6b78118992477 ./test/utils/uni.ts
c3d718f6909e8ed75a73b2c4c9416487197d84108a617fda9315afe60f3df5af ./test/utils/logger.ts
8cc6c09f8181c159c64257318bbe50cc3a3e0574fdfbeb07a65d12a82659bd28 ./test/utils/rollover.ts
d629e629864f5062bfa441b427ad696853b1b00e644eb632fc9cc93d4d8bdecf ./test/utils/getters.ts
95e0919c58dd3dd10cd81597a7a034888902e174046af6e66c269077ddad8810 ./test/utils/signing.ts
```

Changelog

- 2021-12-22 - Initial report
- 2022-01-25 - Re-audit

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.