Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

February 25th 2021 — Quantstamp Verified

MetaVault V2

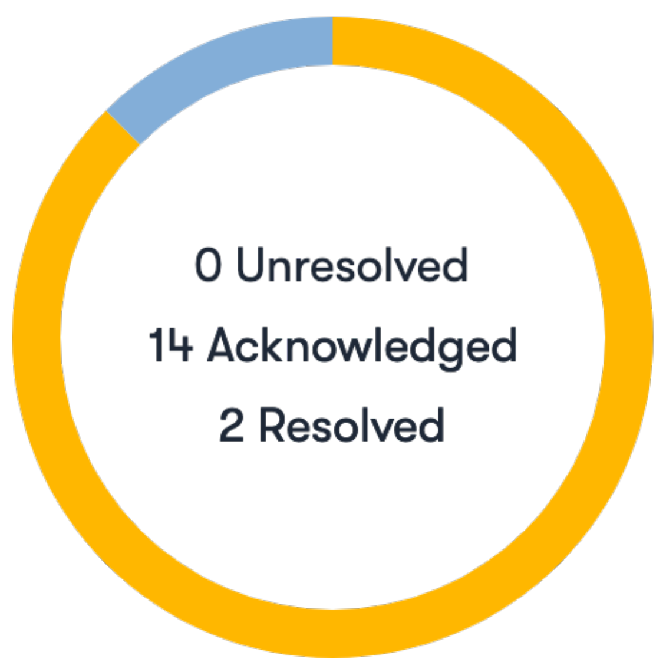
This security review was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	DeFi Aggregator
Reviewers	Sebastian Banescu, Senior Research Engineer Jose Ignacio Orlicki, Senior Engineer Martin Derka, Senior Research Engineer
Timeline	2021-01-25 through 2021-02-20
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	yAxis Blog
Documentation Quality	Low
Test Quality	Medium
Source Code	

Repository	Commit
metavault (audit)	3538b8a
metavault (1st reaudit)	d742367
metavault (2nd reaudit)	e6f1d1c

Total Issues	**16**	(2 Resolved)
High Risk Issues	1	(0 Resolved)
Medium Risk Issues	2	(0 Resolved)
Low Risk Issues	7	(2 Resolved)
Informational Risk Issues	5	(0 Resolved)
Undetermined Risk Issues	1	(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

After first audit: Quantstamp has performed a security review of the yAxis Metavault V2. During this review we have not uncovered any high severity vulnerabilities. We have detected 15 vulnerabilities of Medium and lower severity levels, as well as 6 best practice issues, missing tests and code comments. It is recommended to address these issues before deploying the system in production.

After reaudit: The report has been updated based on the fixes performed in commit `d742367`. Even though the summary below indicates that most items have been Acknowledged, we note that this is mainly due to the issues found in the `yAxisMetavault` contract. Most of the issues found in other contracts have been resolved as indicated in the sub-items of each of the detailed finding descriptions on the following pages of this report. For those findings which were partially fixed, we have indicated the status per sub-item in the enumeration. We have also indicated "Updates from the dev team" for each finding under the "Recommendation". Best practice issues, missing tests and code comment issues have not been resolved.

Note: Only the files listed in the Appendix of this audit report were in scope for the audit and the reaudit. The following files were out of scope and were not audited:

1. `contracts/metavault/strategies/StrategyFlamIncome.sol`
2. `contracts/metavault/strategies/StrategyGenericVault.sol`
3. `contracts/metavault/strategies/StrategyIdle.sol`
4. `contracts/metavault/strategies/StrategyYearnV2.sol`
5. `contracts/metavault/strategies/StrategydYdXSoloMargin.sol`

ID	Description	Severity	Status
QSP-1	Curve 3pool Imbalance Attack	High	Acknowledged
QSP-2	Integer Overflow / Underflow	Medium	Acknowledged
QSP-3	Strategy caps are not always enforced	Medium	Acknowledged
QSP-4	High slippage possible	Low	Acknowledged
QSP-5	Epochs can overlap affecting `getMultiplier`	Low	Acknowledged
QSP-6	Wrong address could be used instead of stable-coin	Low	Fixed
QSP-7	Adding and removing strategies is error prone	Low	Fixed
QSP-8	Gas Usage / `for` Loop Concerns	Low	Acknowledged
QSP-9	Missing input validation	Low	Acknowledged
QSP-10	Sensitive functions do not emit any events	Low	Acknowledged
QSP-11	Block Timestamp Manipulation	Informational	Acknowledged
QSP-12	Implicit assumptions	Informational	Acknowledged
QSP-13	Privileged Roles and Ownership	Informational	Acknowledged
QSP-14	Unchecked Return Value	Informational	Acknowledged
QSP-15	Unlocked Pragma	Informational	Acknowledged
QSP-16	Defense in depth to avoid reentrancy	Undetermined	Acknowledged

Quantstamp Review 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 reviewing 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 security review.

Setup

Tool Setup:

- Slither v0.7.0

Steps taken to run the tools:

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

Findings

QSP-1 Curve 3pool Imbalance Attack

Severity: *High Risk*

Status: Acknowledged

File(s) affected: `StableSwap3PoolConverter.sol, yAxisMetaVault.sol`

Description: Since the yAxis Metavalut allows deposits of DAI, USDC, USDT into the Curve 3pool in any proportion, it is vulnerabile to the large variation of the Curve 3pool imbalance attack, which have exploited the yearn.finance yDAI vault on February 4th, 2021. This is possible in the current implementation where the Chainlink oracle is being used because the `StableSwap3PoolConverter.convert` function will be oblivious to any imbalance between the 3 tokens. Moreover, the price returned by the Chainlink oracle is not the real-time price and it may be stale. The auditors believe that the deposit path is vulnerable to market manipulation under very specific market conditions and for a limited period of time (e.g., the ratios of funds inside 3pool naturally change, and the oracle does not hold the proper price yet). An analogous situation applies for the withdrawal path.

Recommendation: The auditors suggest that yAxis disables deposits and withdrawals of a single asset by reverting inside the convert function based on the `_input` and `_output` values:

1. on deposit, conversion to 3CRV needs to revert, and
2. on withdrawal, conversion from 3CRV needs to revert.

If this is not achievable using the current interface between the converter and the `yAxisMetaVault`, the team can implement a converter that can hold additional mandatory context (indication of a deposit/withdrawal), and prepending the `yAxisMetaVault` contract with an auxiliary contract that is the only contract able to set and clear the context for the converter. This would force users to deposit and withdraw only via the auxiliary contract, would ensure that the converter has the context it needs for deciding if the conversion should be performed, and it could revert under an unsafe transaction.

Update from dev team: We have implemented an oracle contract to be used by the converter which is secured by Chainlink's price feeds. While this can't completely mitigate the possibility of attack due to even small amounts of slippage, this issue will be resolved in our next iteration of the vault which is under active development.

QSP-2 Integer Overflow / Underflow

Severity: Medium Risk

Status: Acknowledged

File(s) affected: `StrategyControllerV2.sol`, `yAxisMetaVault.sol`

Related Issue(s): SWC-101

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute. Integer overflow and underflow may cause many unexpected kinds of behavior and was the core reason for the `batchOverflow` attack. Here's an example with `uint8` variables, meaning unsigned integers with a range of `0..255`. `function under_over_flow() public { uint8 num_players = 0; num_players = num_players - 1; // 0 - 1 now equals 255! if (num_players == 255) { emit LogUnderflow(); // underflow occurred } uint8 jackpot = 255; jackpot = jackpot + 1; // 255 + 1 now equals 0! if (jackpot == 0) { emit LogOverflow(); // overflow occurred } }`
The following instances of this issue were detected in the code base:

1. **[Fixed]** In function `StrategyControllerV2.getBestStrategyEarn()` there is a typo or logic error that leads to underflow of the loop iterator on L493: `for (uint i = k; i >= 0; i--) {`
2. **[Acknowledged]** There is a subtraction `10000 - _withdrawFee` inside the `yAxisMetaVault.calc_token_amount_withdraw` function. This could lead to integer underflow if the `_shares` parameter provided by the caller of this `external` function is too high.

Recommendation: Use `SafeMath` instead of primitive arithmetic operations.

Update from dev team: [regarding item 2] "this will be taken into account in consuming UIs, but is not worth redeploying and making all users withdraw and deposit."

QSP-3 Strategy caps are not always enforced

Severity: Medium Risk

Status: Acknowledged

File(s) affected: `StrategyControllerV2.sol`

Description: The `StrategyControllerV2.getBestStrategyEarn` function does not revert if there is no strategy found which would satisfy the cap requirement. This function returns the last strategy for the given token in that case. This could be problematic if the cap for a given strategy is expected to be enforced.

Recommendation: Force a revert after the `for`-loop inside the `StrategyControllerV2.getBestStrategyEarn` function.

Update from dev team: this is intentional and also documented already. Having no strategies which satisfy the cap requirement would be a failure of the strategist, and our processes will ensure that it doesn't happen. However, users should still be able to deposit.

QSP-4 High slippage possible

Severity: Low Risk

Status: Acknowledged

File(s) affected: `StableSwap3PoolConverter.sol`, `BaseStrategy.sol`

Description: [Fixed] The `StableSwap3PoolConverter.convert` function uses the magic number `1` to specify the minimum amount of tokens expected back when:

1. Adding liquidity to the `stableSwap3Pool` on L87: `stableSwap3Pool.add_liquidity(amounts, 1);`
2. Removing liquidity from the `stableSwap3Pool` on L98: `stableSwap3Pool.remove_liquidity_one_coin(_inputAmount, i, 1);`

This could lead to a high slippage when performing the trade if the pool is not properly balanced at that point in time. This might be problematic for functions such as `yAxisMetavault.withdraw` and `yAxisMetabult.earnExtra`, which do not have a minimum expected amount as an input parameter like the `yAxisMetavault.deposit` function does. This is also problematic when withdrawing funds from some of the strategies, because they employ the converter contract as well.

[Acknowledged] The same issue occurs inside the `BaseStrategy._swapTokens` internal function, which is used when paying fees and harvesting. It swaps on L257-264 with the minimum expected amount of `1`:

```
router.swapExactTokensForTokens(
            _amount,
            1,
            path,
            address(this),
            // solhint-disable-next-line not-rely-on-time
            block.timestamp.add(1800)
        );
```

Recommendation: Replace the `1` value with a value that is close to the expected value.

Update from dev team:

1. [regarding the Fixed item] the converter now has a configurable slippage variable (updated by governance, defaulted to 1%) which will revert if the conversion has slippage beyond that amount.
2. [regarding the Acknowledged item] the real fix would be a price oracle, but slippage there isn't too much of a concern because it's purchasing for fees, but doesn't affect user deposits.

QSP-5 Epochs can overlap affecting `getMultiplier`

Severity: Low Risk

Status: Acknowledged

File(s) affected: `yAxisMetaVault.sol`

Description: The `yAxisMetaVault.setEpochEndBlock` function doesn't check if the `_epochEndBlock` for the given `_index` is higher than the epoch end block for the following index. This could lead to setting the end block for `_index`, higher than the end block for `_index + 1`, which would affect the value of the multiplier returned by the `yAxisMetaVault.getMultiplier` function. More specifically, that function and any other function using it would throw a subtraction overflow error. This is because the `yAxisMetaVault.getMultiplier` function assumes the `epochEndBlocks` array is sorted in ascending order.

Recommendation: Add a check inside `yAxisMetaVault.setEpochEndBlock` to check that `_epochEndBlock < epochEndBlocks[_index + 1]` iff `_index < 4`.

Update from dev team: we have no plans of calling this function in the future.

QSP-6 Wrong address could be used instead of stable-coin

Severity: Low Risk

Status: Fixed

File(s) affected: `StrategyPickle3Crv.sol`

Description: The `StrategyPickle3Crv.setStableForLiquidity` function does not check if the provided input parameter is the address of a stable-coin. Therefore, an authorized address could set a different address (accidentally or intentionally), which would lead to unexpected results when harvesting funds from that strategy.

Recommendation: Check if `_stableForAddLiquidity` is equal to the address of the 3 supported stable coins: DAI, USDC, USDT.

Update from dev team: should be noted that the wrong addresses could still be provided at deployment time. However, the team and community would be able to check that before the strategies is added to the controller.

QSP-7 Adding and removing strategies is error prone

Severity: Low Risk

Status: Fixed

File(s) affected: `StrategyControllerV2.sol`

Description: If a strategy can be harvested, it should be added to the `yAxisMetaVaultHarvester` contract as well as the `StrategyControllerV2`. However, this is done by calling 2 separate functions manually:

1. The `StrategyControllerV2.addStrategy` function and the
2. The `yAxisMetaVaultHarvester.addStrategy` function.

Due to human-error the operator of the authorized addresses could forget to add the strategy in one of the contracts. Or they could manually enter incorrect input parameters due to mistyping (copying).
The same issue occurs when removing strategies, because they should be removed from both contracts.

Recommendation: Change the add and remove functions inside `StrategyControllerV2` such that they automatically add and remove strategies to and from `yAxisMetaVaultHarvester` respectively.

Update from dev team: we have consolidated these functions to the `StrategyControllerV2.addStrategy` method, as well as the controller's `setStrategy` method.

QSP-8 Gas Usage / `for` Loop Concerns

Severity: Low Risk

Status: Acknowledged

File(s) affected: `StrategyControllerV2.sol`

Description: Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely.

There is no hard cap on the maximum number of strategies allowed by the `StrategyControllerV2` contract. The `maxStrategies` is set to `10` by default, but the `setMaxStrategies()` function could increase that number. If the number becomes too large then `balanceOf()`, `withdraw()`, `getBestStrategyEarn()` and `getBestStrategyWithdraw()` functions could reach the block gas limit due to the `for`-loop that iterates over all strategies. This would create a DoS for the end-user.

Recommendation: It is best to break such loops into individual functions as possible. If that is not possible, then perform a gas analysis and check what is the maximum number of strategies that can be supported by the aforementioned functions and add a constraint that will prevent `maxStrategies` being set to a higher value than that.

Update from dev team: this has been investigated internally and the most efficient way to allow for potentially hundreds of strategies is to rewrite a significant amount of code, including the `yAxisMetavault`. Until then, the strategist will be concerned about preventing too many strategies from being added at once.

QSP-9 Missing input validation

Severity: Low Risk

Status: Acknowledged

File(s) affected: `yAxisMetaVault.sol`, `StrategyControllerV2.sol`, `BaseStrategy.sol`, `StrategyCurve3Crv.sol`, `StrategyDforce.sol`, `StrategyPickle3Crv.sol`

Description: The following functions are missing input parameter validations:

1. **[Acknowledged]** `yAxisMetaVault.setMin` does not require the value of `_min` to be lower than `max`. This could have a high impact on the value returned by `yAxisMetaVault.available`.
2. **[Acknowledged]** `yAxisMetaVault.setGovernance` does not require the value of `_governance` to be different from `address(0)` and different from the current `governance` address.
3. **[Acknowledged]** `yAxisMetaVault.setController` does not require the value of `_controller` to be different from `address(0)` and different from the current `controller` address.
4. **[Acknowledged]** `yAxisMetaVault.setConverter` does not require the value of `_converter` to be different from `address(0)` and different from the current `converter` address.
5. **[Acknowledged]** `yAxisMetaVault.setVaultManager` does not require the value of `_vaultManager` to be different from `address(0)` and different from the current `vaultManager` address.
6. **[Acknowledged]** `yAxisMetaVault.setYaxPerBlock` does not require the value of `_yaxPerBlock` to be greater than `0`.

7. **[Acknowledged]** `yAxisMetaVault.setTreasuryWallet` does not require the value of `_treasuryWallet` to be different from `address(0)` and different from the current `treasuryWallet` address.
8. **[Acknowledged]** `yAxisMetaVault.getMultiplier` does not require the value of the `_from` to be lower than `_to`. Failing to do so would result in a SafeMath error.
9. **[Fixed]** `StrategyControllerV2.reorderStrategies` does not check if `_strategy1 != _strategy2`.
10. **[Partially resolved]** The `StrategyControllerV2.setMaxStrategies` function does not check if the given input parameter is greater than zero. It also does not check that this number be at least as high as the current highest number of strategies for any given token. Therefore, this number could be lower than the actual number of strategies that a certain token has and it will be undetected by the logic.
11. **[Fixed]** The `BaseStrategy.constructor` does not check if the 5 input parameters of type `address` are different from `address(0)`. Note that 2 of these are immutable and cannot be changed afterwards. The same issue applies to all the strategies that extend the `BaseStrategy`.

Recommendation: Add input parameter validation for each of the functions mentioned in the description above.

Update from dev team:

1. [regarding the Acknowledged items] the yAxisMetavault contract is already deployed and this change isn't significant enough to require users to withdraw and deposit.
2. [regarding the Partially resolved item] that _maxStrategies could be less than the number of strategies for a given token. This would intentionally prevent us from adding more strategies to that token.

QSP-10 Sensitive functions do not emit any events

Severity: Low Risk

Status: Acknowledged

File(s) affected: `All`

Description: There is a discrepancy regarding how events are used in the code base. For example, the `StrategyControllerV2` contract defines some events which are emitted in the corresponding functions. However, there are several sensitive functions in that contract and other contracts which do not emit any events, even though the functions perform changes that could have significant implications for end-users. Here are a few examples of such functions (just to name a few):

1. **[Fixed]** `StrategyControllerV2.{setVaultManager, setConverter}`
2. **[Fixed]** All external, non-view functions from `BaseStrategy`
3. **[Acknowledged]** `yAxisMetaVault.{setGovernance, setController, setGovernance}`
4. **[Fixed]** `StrategyPickle3Crv.{setStableForLiquidity, setPickleMasterChef, setPoolId}`

Recommendation: Declare and emit events in all external, non-view functions which can have an impact on end-users.

Update from dev team:

1. added events to `setVaultManager` and `setConverter`.
2. added events for `ApproveForSpender`, `SetController`, `SetRouter`, `Skim`, and `Withdraw`. The other non-view functions have events in the `StrategyControllerV2` contract.
3. the `yAxisMetavault` contract is already deployed and this change isn't significant enough to require users to withdraw and deposit.
4. added events for `setStableForLiquidity`, `setPickeMasterChef`, and `setPoolId`.

QSP-11 Block Timestamp Manipulation

Severity: Informational

Status: Acknowledged

File(s) affected: `yAxisMetaVaultHarvester.sol, StrategyStabilize.sol`

Related Issue(s): SWC-116

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes by up to 900 seconds. If a smart contract relies on a timestamp, it must take this into account.

The following instances of this issue have been observed in the code base:

1. The `yAxisMetaVaultHarvester.canHarvest` function returns `true` or `false` based on the values of `block.timestamp`.
2. The `StrategyStabilize.calculateZPATokenWithdrawFee` functions computes the withdrawal fee based on the `block.timestamp`.

Recommendation: Add integration tests that demonstrate that a 900 second difference in the `block.timestamp` will not have a significant impact on any end-user. Otherwise, clarify to end-users that `block.timestamp` can be manipulated by malicious miners by `900` seconds and what impact that may have.

Update from dev team:

1. this is not a critical function to be able to call harvest with absolute precision.
2. this is an external dependency where the withdrawal fee is based on `block.timestamp`.

QSP-12 Implicit assumptions

Severity: Informational

Status: Acknowledged

File(s) affected: `All contracts`

Description: The following implicit assumptions were observed while auditing the code:

1. The `BLOCKS_PER_WEEK` constant assumes that the average block time is and will remain 13 seconds for all 5 epochs (6 months) after launch.
2. The `epochEndBlocks` values assume that one month has exactly 4 weeks.
3. There is an ordering of roles, namely `governance > strategist > harvester` inside `StrategyControllerV2.sol`. This means that `governance` can do anything that `strategist` can do, who can do anything that `harvester` can do.
4. The `StrategyControllerV2.getBestStrategyWithdraw` function assumes there are sufficient funds in all strategies of a token to cover any requested withdraw amount. If not, then vault withdraw function call reverts. However, the user doesn't get a descriptive error message.
5. The `BaseStrategy.balanceOf` function assumes that `balanceOfWant()` and `balanceOfPool()` return the balance using the same token address. This does not seem to be strictly enforced anywhere and must be checked in the code for each newly developed strategy.

Recommendation: To avoid user annoyance or any type of reputation damage, we recommend making these assumptions explicit to end-users via publicly facing documentation (e.g. FAQ) and/or GUI tool-tips or pop-ups.

Update from dev team:

1. distribution should end within the next few months. Block times aren't critical for this distribution strategy.
2. same reason as above.
3. this is intentional. Governance should be able to do anything that any privileged role can do. The Strategist can maintain the the protocol in a limited function. And the Harvester can only harvest. Getting better revert reason messages doesn't seem to be worth the extra bytecode in these instances.
4. it should also be noted that the MetaVault can have funds that are taken as priority to withdrawing out of strategies.
5. to convert balances to the same token address here would make all user interactions with the vault significantly more expensive (since this function is used to determine the amount of vault shares MVLT to provide to the user).

QSP-13 Privileged Roles and Ownership

Severity: *Informational*

Status: Acknowledged

File(s) affected: `yAxisMetaVault.sol`, `BaseStrategy.sol`, `StrategyControllerV2.sol`

Description: Smart contracts will often have `owner` variables to designate the address with special privileges to make modifications to the smart contract. In this project there there are other important roles, which will be described next. The `governance` address has many privileges in the `yAxisMetaVault`, `yAxisMetaVaultManager`, `yAxisMetaVaultHavester` contracts, namely:

1. It can claim the entire `insurance` amount at any point in time, which will transfer the amount in 3CRV to the treasury wallet.
2. It can set any state variable including the: `treasuryWallet`, `epochRewardMultipliers`, `epochEndBlocks`, `yaxPerBlock`, `totalDepositCap`, `vaultManager`, `converter`, `controller`, `governance`, `min`, `insuranceFee`, `insurancePool`, `insurancePoolFee`, `stakingPool`, `stakingPoolShareFee`, `strategist`, `trasury`, `trasuryFee`, `withdrawalProtectionFee`, the `YAX` token address, the status of any controller, strategy and vault, `harvester`.
3. It can transfer any amount of any token from the MetaVault, any strategy or controller to itself. This includes the 3CRV, MVLT and YAX tokens.
4. It can approve any address to spend any amount of any token from any strategy.
5. It can set the `router` of any strategy.

The `controller` address has the following privileges in the `yAxisMetaVault`, `yAxisMetaVaultManager` contract, namely:

1. It can call `claimInsurance()` at any point in time, which will cause the `insurance` amount in 3CRV to be simply set to zero, without making any transfer. The benefit here is that this would increase the share price.
2. It can call `harvest()` at any point in time and transfer any amount of any token (except the 3CRV) from the MetaVault, this includes the MVLT and YAX tokens.

The 2 aforementioned roles have the following privileges in the `yAxisMetaVaultHarvester` contract, namely:

1. They can set the following state variables: `controller`, `harvester` and `vaultManager`.
2. They can add or remove strategies.
3. They can send stuck `want` tokens in any strategy to the `controller`.
4. They can transfer any amount of the `want` token to the vault.
5. They can transfer the full balance of any token (except for the `want` token) from any strategy to the controller.

Recommendation: These privileged roles and their capabilities need to be made clear to the users via publicly facing documentation (e.g., blog post, FAQ page, etc.)

Update from dev team: we'll describe the roles of privileged addresses in the project repository's wiki to start.

QSP-14 Unchecked Return Value

Severity: *Informational*

Status: Acknowledged

File(s) affected: `All`

Description: Most functions will return a `true` or `false` value upon success. Some functions, like `send()`, are more crucial to check than others. It's important to ensure that every necessary function is checked. Here are just a few examples of functions which ignore return values (the list is not exhaustive, otherwise it would be too long):

1. **[Acknowledged]** `yAxisMetaVault.deposit` ignores return value by `converter.convert(_input,address(token3CRV),_amount)`
2. **[Acknowledged]** `yAxisMetaVault.depositAll` ignores return value by `converter.convert_stables(_stablesAmounts)`
3. **[Acknowledged]** `yAxisMetaVault.stakeShares` ignores return value by `IERC20(address(this)).transferFrom(msg.sender,address(this),_shares)`
4. **[Acknowledged]** `yAxisMetaVault.unstake(uint256)` ignores return value by `IERC20(address(this)).transfer(msg.sender,_amount)`
5. **[Acknowledged]** `yAxisMetaVault.earnExtra(address)` ignores return value by `converter.convert(_token,address(token3CRV),_amount)`
6. **[Fixed]** `StrategyControllerV2.withdrawAll` ignores return value by `IStrategy(_strategy).withdrawAll()`

Recommendation: Always check return values of functions and handle them accordingly.

Update from dev team:

1. [regarding the Acknowledged items] the yAxisMetavault contract is already deployed and this change isn't significant enough to require users to withdraw and deposit.
2. [regarding the Fixed item] since the strategy sends funds directly to the vault, there doesn't seem to be any point of this return value so it has been removed.

QSP-15 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

File(s) affected: `interfaces/*.sol`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.6.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked". Several contracts inside the `interface/` subdirectory seem to have an unlocked pragma.

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

Update from dev team: any base contract that is actually what is deployed should have a locked pragma.

QSP-16 Defense in depth to avoid reentrancy

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `All`

Description: Due to the high amount of external dependencies and interactions with other DeFi platforms and tokens, there is a non-negligible risk of complex re-entrancy attacks. Such complex attacks have been able fairly recently been able to exploit projects such as dForce https://quantstamp.com/blog/how-the-dforce-hacker-used-reentrancy-to-steal-25-million

Recommendation: All functions which involve transfers of funds in contracts such as strategies, vaults, converters and controllers should include reentrancy guards.

Update from dev team: we restrict the use of smart contract depositors which would cause reentrancy to be an issue. Adding a `nonReentrant` modifier to every function would add an unnecessary gas cost to end users.

Automated Analyses

Slither

Slither reported hundreds of issues. We have filtered out all false positives and have integrated all true positives in the findings in this report.

Code Documentation

Each function should at least have a brief description of its purpose and a description of each input and output parameter. This is not the case with many functions and contracts in the code base. Counting the number of lines of code versus the number of lines with comments in the `contracts` folder shows that the ratio of comment to code is around 25% (see table below). We recommend having at least a 50% comment to code ratio to improve maintainability of the code. We note that some files are properly commented (e.g. `StrategyControllerV2.sol`), while others are poorly commented (e.g. `yAxisMetaVault.sol`).

```
contracts % cloc .
      43 text files.
      43 unique files.
       0 files ignored.

github.com/AlDanial/cloc v 1.88  T=0.03 s (1706.8 files/s, 145794.6 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Solidity                        43            496            637           2540
-------------------------------------------------------------------------------
SUM:                            43            496            637           2540
-------------------------------------------------------------------------------
```

Adherence to Best Practices

1. Inconsistent use of `uint` vs `uint256`. Replace all usages of `uint` with the right bit-width of unsigned integers, e.g. `uint256`, `uint8`, etc.
2. Inconsistent usage of hard-coded addresses. The `treasuryWallet` in `yAxisMetaVault.sol` is hardcoded, but the addresses of the DAI, USDC, USDT, YAX and 3CRV tokens are not. Instead they are provided as `constructor` parameters.
3. Inconsistent naming style, e.g.
    - the `max` constant in `yAxisMetaVault.sol` is not written in `UPPER_CASE`.
    - the `calc_token_amount_deposit` function does not use `camelCase`.
4. Magic numbers should be replaced by named constants to improve code maintainability. The name of the constants should be indicative of their semantics not their value. The following instances were detected:
    - `1e12` is used 7 times in `yAxisMetaVault.sol` in several functions.
    - `10000` is used 4 times in `yAxisMetaVault.sol` in several functions.
    - `10000` is used once in `StrategyControllerV2.sol` in `withdrawFee`.

- . `10**12` is used twice in `StrategyCurve3Crv.sol` in `getMostPremium`.
- . `1e18` is used 10 times in multiple files and functions.

5. Error messages in `require` statements should server as debugging aids for users and developers. There are several instances of error messages in the `yAxisVaultHarverster`, `StrategyControllerV2` contracts where the error message is simply the a word or function name prefixed by an exclamation mark, e.g. `!harvester`, `!canHarvest`. These error messages should be changed to descriptive sentences.
6. The `yAxisMetaVault` contract code estimate surpasses 24576 bytes (a constraint presented in EIP-170). This contract may not be deployable on mainnet. Consider empowering the optimizer, turning off error strings, or utilizing libraries. Read official discussion about contract size limits and how-to reduce the size of your contracts (https://ethereum.org/en/developers/tutorials/downsizing-contracts-to-fight-the-contract-size-limit/).

Test Results

Test Suite Results

We confirm that all existing tests are passing.

```
Network Info
============
> HardhatEVM: v2.0.7
> network:    hardhat



  BaseStrategy
     ✓ should not allow unpermissioned callers (117ms)
     ✓ should approve tokens for spending
     ✓ should skim stuck tokens out of the strategy (48ms)
     ✓ should send the insurancePoolFee to the insurancePool (423ms)
     ✓ should set the controller
     ✓ should set the router

  MockPickleJar
     ✓ pjar deposit (117ms)
     ✓ pjar withdraw (52ms)
     ✓ pjar withdrawAll (49ms)
     ✓ get PJAR (via pjar deposit)
     ✓ pchef deposit (49ms)
     ✓ pchef deposit(0) - claim (52ms)
     ✓ pchef withdraw (38ms)
     ✓ pchef emergencyWithdraw (39ms)

  StableSwap3PoolConverter
     ✓ should not allow unpermissioned callers (57ms)
     ✓ should approve for spender (50ms)
     ✓ should set the vault manager (148ms)
     ✓ should set the StableSwap3Pool (165ms)

  StrategyControllerV2
     ✓ should deploy with expected state
     ✓ should not allow unpermissioned callers (206ms)
     ✓ should add a strategy (81ms)
     ✓ should deposit into first strategy (357ms)
     ✓ should obey maximum strategies amount (44ms)
     ✓ should add an additional strategy (77ms)
     ✓ should deposit into second strategy (252ms)
     ✓ should reorder strategies (48ms)
     ✓ should withdraw excess funds when reducing a strategy cap (121ms)
     ✓ should deposit into first strategy when cap of second is reached (236ms)
     ✓ should withdraw small amounts (254ms)
     ✓ should deposit large amounts into a single strategy (281ms)
     ✓ should withdraw large amounts from multiple strategies (275ms)
     ✓ should remove strategies (121ms)
     ✓ should deposit/earn to the remaining strategy (213ms)
     ✓ should allow all strategies to be removed (87ms)
     ✓ should allow deposits without strategies (156ms)
     ✓ should earn to a newly added strategy (136ms)
     ✓ should harvest strategy through controller (269ms)

  StrategyCurve3Crv
     ✓ should deploy with initial state set (39ms)
     ✓ should deposit DAI (236ms)
     ✓ should harvest (253ms)
     ✓ should withdraw to DAI (217ms)
     ✓ should withdrawAll to 3CRV (157ms)
     ✓ should deposit USDT (198ms)
     ✓ should withdrawAll by controller (49ms)

  StrategyDforce
     ✓ should deploy with initial state set
     ✓ should deposit DAI (362ms)
     ✓ should harvest (268ms)
     ✓ should withdraw to DAI (328ms)
     ✓ should withdrawAll to 3CRV (265ms)
     ✓ should deposit USDT (374ms)
     ✓ should withdrawAll by controller (125ms)

  StrategyFlamIncome
     ✓ should deploy with initial state set
     ✓ should deposit USDT (380ms)
     ✓ should withdraw to DAI (344ms)
     ✓ should withdrawAll to 3CRV (121ms)
     ✓ should deposit USDT (349ms)
     ✓ should withdrawAll by controller (131ms)

  StrategyGenericVault
     ✓ should deploy with initial state set (40ms)
     ✓ should deposit USDT (385ms)
     ✓ should withdraw to DAI (356ms)
     ✓ should withdrawAll to 3CRV (159ms)
     ✓ should deposit USDT (344ms)
     ✓ should withdrawAll by controller (121ms)

  StrategyIdle
     ✓ should deploy with initial state set (43ms)
     ✓ should deposit DAI (319ms)
     ✓ should harvest (255ms)
     ✓ should withdraw to DAI (356ms)
     ✓ should withdrawAll to 3CRV (297ms)
     ✓ should deposit USDT (348ms)
     ✓ should withdrawAll by controller (176ms)

  StrategyPickle3Crv
     ✓ should deploy with initial state set
     ✓ should deposit DAI (267ms)
     ✓ should harvest (274ms)
     ✓ should withdraw to DAI (249ms)
     ✓ should withdrawAll to 3CRV (193ms)
     ✓ should deposit USDT (282ms)
     ✓ should withdrawAll by controller (68ms)

  StrategyStabilize
     ✓ should deploy with initial state set
     ✓ should deposit DAI (370ms)
     ✓ should harvest (242ms)
     ✓ should withdraw to DAI (348ms)
     ✓ should withdrawAll to 3CRV (297ms)
     ✓ should deposit USDT (391ms)
     ✓ should withdrawAll by controller (159ms)

  StrategyYearnV2
     ✓ should deploy with initial state set
     ✓ should deposit DAI (324ms)
     ✓ should withdraw to DAI (298ms)
     ✓ should withdrawAll to 3CRV (250ms)
     ✓ should deposit USDT (339ms)
     ✓ should withdrawAll by controller (116ms)

  StrategydYdXSoloMargin
     ✓ should deploy with initial state set (40ms)
     ✓ should deposit DAI (566ms)
     ✓ should withdraw to DAI (348ms)
     ✓ should withdrawAll to 3CRV (265ms)
```

```
    ✓ should deposit USDT (354ms)
    ✓ should withdrawAll by controller (147ms)

  stuck_funds.test
    ✓ deposit (219ms)
    ✓ stuck WETH in strategy (74ms)
    ✓ stuck WETH in controller (67ms)
    ✓ stuck t3crv.address (core) in strategy (103ms)

  yAxisMetaVault
    ✓ should deposit (200ms)
    ✓ should depositAll (185ms)
    ✓ should stakeShares (167ms)
    ✓ should pendingYax
    ✓ should unstake(0) for getting reward (57ms)
    ✓ should unstake (47ms)
    ✓ should withdraw T3CRV (55ms)
    ✓ should withdraw DAI (105ms)
    ✓ should withdraw USDT (116ms)
    ✓ should withdraw need unstake (158ms)
    ✓ should withdrawAll to USDC (272ms)

  yAxisMetaVaultHarvester
    ✓ should not allow unpermissioned callers (91ms)
    ✓ should set the controller
    ✓ should set the vault manager
    ✓ should set harvesters
    ✓ should add strategies
    ✓ should harvest added strategies (197ms)
    ✓ should add additional strategies (58ms)
    ✓ should rotate harvesting strategies (232ms)
    ✓ should not allow harvestNextStrategy until timeout has passed (109ms)
    ✓ should remove strategies

  yAxisMetaVaultManager
    ✓ should deploy with expected state (55ms)
    ✓ should not allow unpermissioned callers (109ms)
    ✓ should set the insurance fee
    ✓ should set the insurance pool
    ✓ should set the insurance pool fee
    ✓ should set the staking pool
    ✓ should set the staking pool fee
    ✓ should set the treasury
    ✓ should set the treasury balance
    ✓ should set the treasury fee
    ✓ should set the withdrawal protection fee
    ✓ should set the yax.address token
    ✓ should set the controller status
    ✓ should set the vault status
    ✓ should set the harvester
    ✓ should set the strategist
    ✓ should set the governance


  138 passing (32s)
```

Code Coverage

Branch coverage is very low. We recommend that all coverage values be close to 100% to ensure that all the functionality of the smart contracts is properly tested. This way, any changes made to the code which introduce a bug have higher changes of being automatically detected by the test suite.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
metavault/	80.94	55.8	78.82	81.63	
IController.sol	100	100	100	100	
IConverter.sol	100	100	100	100	
IHarvester.sol	100	100	100	100	
IMetaVault.sol	100	100	100	100	
IStableSwap3Pool.sol	100	100	100	100	
IStrategy.sol	100	100	100	100	
IStrategyControllerConverter.sol	100	100	100	100	
ISwap.sol	100	100	100	100	
IVaultManager.sol	100	100	100	100	
StableSwap3PoolConverter.sol	85.07	70.83	73.33	85.07	… 168,171,179
yAxisMetaVault.sol	72.69	40.97	65	73.42	… 487,488,489
yAxisMetaVaultHarvester.sol	100	92.86	100	100	
yAxisMetaVaultManager.sol	97.83	85.71	100	97.83	69
metavault/controllers/	91.38	77.27	79.31	92.06	
StrategyControllerV1.sol	100	100	100	100	
StrategyControllerV2.sol	91.38	77.27	79.31	92.06	… 480,498,509
metavault/strategies/	95.34	54.08	93.26	95.34	
BaseStrategy.sol	100	70.59	100	100	
StrategyCurve3Crv.sol	90.38	42.31	100	90.38	93,94,97,98,101
StrategyDforce.sol	100	50	100	100	
StrategyFlamIncome.sol	100	100	100	100	
StrategyGenericVault.sol	96.15	57.14	87.5	96	67
StrategyIdle.sol	100	50	100	100	
StrategyPickle3Crv.sol	84.38	50	72.73	84.38	… 9,93,94,152
StrategyStabilize.sol	98.25	53.85	100	98.25	78
StrategyYearnV2.sol	96.15	58.33	85.71	96.15	59
StrategydYdXSoloMargin.sol	93.94	50	85.71	93.94	93,144
All files	**88.9**	**57.11**	**85.22**	**89.32**	

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

```
4d42b55c63f65117e7998fb3162cb83bb28e27560826d260665fed9885955ceb ./contracts/metavault/IStableSwap3Pool.sol
f5507ad8b143ae52d6d86a2953d33efdc14d127d4cf487d4ca97c932ddf1e52a ./contracts/metavault/StableSwap3PoolConverter.sol
6235b87e9b50d36614d72115fd796d4eac9c05cb23d9559cb84fc817ca9cc12c ./contracts/metavault/IStrategyControllerConverter.sol
546d8ada5e488c13d15b1b59800b548b57ba70b30c8095e2f4c6857f3b702f79 ./contracts/metavault/yAxisMetaVaultHarvester.sol
86d46e8086c18d16cee593f786602b64453ec454ce8b38c2eb7cd5f9ed2ad9e6 ./contracts/metavault/ISwap.sol
70c1be5bfe224e9d37bac2c364ac1b54bc8ad2e79a60e9a386a63a2d3f218883 ./contracts/metavault/yAxisMetaVault.sol
37485916543faa659a85b8c4a63ba9173a33ef541a2dddc1f3e8d099b9d5cfca ./contracts/metavault/yAxisMetaVaultManager.sol
b3111fd8d071bedfcf62d6f3914b1656ee610fe56844f48e6188cf81add6abe7 ./contracts/metavault/IStrategy.sol
b63ffba776449ecba884e647f03fc5134f1e31737b20509167339065d177cf5e ./contracts/metavault/IVaultManager.sol
fb1187f4f9702f1909450a3dc4c2c91f6664ed313756db108dc7c19422c73d2a ./contracts/metavault/IConverter.sol
a3bd974d307fd4dd18bcb614e502ac01ab38f16b497cc30533faea858a02ef85 ./contracts/metavault/IController.sol
```

```
c89c3812b88e41619fc833b6ffa3797befc30d8f08e0c5434ce6059e09b0132d ./contracts/metavault/IMetaVault.sol
a2879d2dde29fbe9949d4cc0f8760c5c7b5adcb774f3996fb1de51fe9d9f4114 ./contracts/metavault/controllers/StrategyControllerV2.sol
c3b96887941a25f44ec9b7d0e45742370f7a49eed338ef08c9c090bcb600ffc0 ./contracts/metavault/controllers/StrategyControllerV1.sol
6af661af0f57fb33ba3db1d27cacaf0a5620bfa42190a502f26beea3d7c293da ./contracts/metavault/mock/MockCurveGauge.sol
361a5fc678ee0a78bbc149c055ef84b85f32818e7ba2c0c71e65a6a6807fd6a4 ./contracts/metavault/mock/MockPickleMasterChef.sol
09a4fb43a7c5b2a74abf526532e7b8232d5d5dddab25b4ac02f6943cc3deac92 ./contracts/metavault/mock/MockERC20.sol
82055856d08062ff5629355654b60549d8d7a120bf0824b03682bfca725da808 ./contracts/metavault/mock/MockUniswapRouter.sol
c2d587a66111dec05155303fcd3e189719f5c51ec9f87fd0e62a444fb4d87ff0 ./contracts/metavault/mock/MockDRewards.sol
8f44b0a17bd587e5623aec20ec9caf67890fb2fcfe6400c58e2cf64ceb0e1bec ./contracts/metavault/mock/MockPickleJar.sol
da0b5dc2dd299dcef1e3c983c9400dd2f60acc12c403ea4e7d07f06ae7880a07 ./contracts/metavault/mock/MockCurveMinter.sol
59ca1debc843a7fe3b12dda690ad14a6cc96d7929012321e5fd7d8f325b4313d ./contracts/metavault/mock/MockzpaToken.sol
f11e1d2db041707871f4d7a3c3e60d48473d43a57b1cbe3bf9634caf641d8624 ./contracts/metavault/mock/MockDErc20.sol
9745593bbc788e23385f841ecfd531f7650de83c7942836a7edf5a1f213c6dc8 ./contracts/metavault/mock/MockStabilizePool.sol
b5023c65bf478b3c8edf8b9ad09a542362739bcad44921c995401cf067665c52 ./contracts/metavault/mock/MockStableSwap3Pool.sol
a5053a6281e741d7f97a1738ba9e8a15fbba2e63cc31267aff1a92b3b4e7c792 ./contracts/metavault/strategies/StrategyStabilize.sol
9b0c72dba00bc42dfa4efa3f391dfec391da29ea26660f6d693fbf5a7c09d969 ./contracts/metavault/strategies/BaseStrategy.sol
e61c83b708c1761fd76d72c7a93cd2721a6d6a503426e11a8671d686ee188d65 ./contracts/metavault/strategies/StrategyDforce.sol
41fb816ed187a7792047c6b688bb4ba9b09119dba05758dae9bc178962993b0c ./contracts/metavault/strategies/StrategyPickle3Crv.sol
a1b7329986ecf454ed3eade55f76cfd9016b7dfe36d8530df048ff6252a8116a ./contracts/metavault/strategies/StrategyCurve3Crv.sol
8a9316edc48adb5067ca27cc4f94384074d9741515cce7c358bd1d01ccf504d9 ./contracts/interfaces/Balancer.sol
e70a733c64043b9b3b60a25c011c315a0c3d510aa6adb536c8d9893d49212401 ./contracts/interfaces/PickleMasterChef.sol
7fdd69af9a8b16882e102883a5f8f957bc47d985fd98772d0edf11f6e39bb668 ./contracts/interfaces/Yfii.sol
aafcaa79cc79654e2ddab07587e3fb5205ba14ab7ec4fb440fe27fb46bb50fce ./contracts/interfaces/DForce.sol
f7d7ef334dc624b42cbfa457b1d153a044f1727aa2f516170187c012a518ff70 ./contracts/interfaces/Aave.sol
753969697e7fb4837e9b0c3ee82435ddf8ccef4b03b7502b5dcf6f0c36e0e0ca ./contracts/interfaces/PickleJar.sol
5a733214b687ab80fe7d38533bab66e0a64c1c155ce6429bcde4801c7602c5a6 ./contracts/interfaces/Uniswap.sol
2401ca8069e2919bf57e6bcb0726f14eb1bb51383f958c2550a527e9ec8bf81b ./contracts/interfaces/MStable.sol
536ddf2d31e29cec157fffe562d3b366801f868809b6973e447352d539abb8c9 ./contracts/interfaces/Stabilize.sol
ab4cf1111d9388aea22ab596574ff43450ea780167b42ec258820cc2c6f90272 ./contracts/interfaces/Converter.sol
d21e4a65e319c222fdbb0477db16e9cd1425d88c4859ad230d35014ead0c2536 ./contracts/interfaces/Curve.sol
456cde2c5f36110f07d4a5351612fb85b4c05a532e4589a8593e9da2c513cee2 ./contracts/interfaces/OneSplitAudit.sol
04e8ea9f84b3b9a0d6371278fed263ae6462a210acadda6a2a4316040fa37f7a ./contracts/interfaces/Gauge.sol
```

Tests

```
44094a9677548c7f1d4d98445a7a128a0088f8d8a08b7e75881d88edd49ddbd8 ./test/metavault/stuck_funds.test.js
2cd777bf61647ace79177a1cf80d08ef461c2766773395d574810fc60f554aa8 ./test/metavault/StrategyControllerV2.test.js
456d3e33061952b5ccf5c50f6ed9ff492d413446c9ebd725eeefd101fbd482be ./test/metavault/yAxisMetaVaultHarvester.test.js
fd00c2c1cbb8e8a9413076c128812fd155a4326028c37d1a762e9f2dce3cdf01 ./test/metavault/StrategyStabilize.test.js
e3f1f4c528d61f56fc90858e32bf2a1a04fc38bdbe837b4c20eb67278c34264f ./test/metavault/yAxisMetaVault.test.js
0ed19cec757ee30d82cf5daedd4fc6408655da90262de39cfce2b466b453645a ./test/metavault/MockPickleJar.test.js
082d5338665267cf35a477cda316242d1c0c40da1c8e505282874aa6c74289ad ./test/metavault/StrategyCurve3Crv.test.js
4d8072eb26c47716eb622e2e77dca12390f9fef3412fdb8c8a71e32fa791cb58 ./test/metavault/yAxisMetaVaultManager.test.js
3ad4123ac36ce45d98737b0c80d9f0d8c83b684ccf945619de63e59c4ccc6ab4 ./test/metavault/StrategyDforce.test.js
eff380d8623b410bebbc1ab41345a392b3549ed4f582cb50cff1262483a684e4 ./test/metavault/StrategyPickle3Crv.test.js
d87a41a0872494a4c041a6a768131bc074c7a10be9a34e52052a4328deb94c10 ./test/helpers/setup.js
e7ad61a93c4824ef0a53071077bb49c20c2f94fd97ab6b7858b0315765c2fbbc ./test/live/multi_strategy_controller_live.test.js
```

Changelog

- 2021-02-06 - Initial report based on commit `3538b8a8c1ea4ec2b68a635c48b938aa1acfbc26`
- 2021-02-09 - Updated report based on commit `d742367bbcffd5604c22a3f1347e668ef067bd09`
- 2021-02-20 - Updated report based on commit `e6f1d1c3ae1b0b126c785659889851294ee1f877`

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.