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**

**Specification**
yakis Blog

**Documentation Quality**
Low

**Test Quality**
Medium

**Source Code**

<table>
<thead>
<tr>
<th>Repository</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>metavault (audit)</td>
<td>36366eb0</td>
</tr>
<tr>
<td>metavault (1st reaudit)</td>
<td>272367</td>
</tr>
<tr>
<td>metavault (2nd reaudit)</td>
<td>ed1564</td>
</tr>
</tbody>
</table>

**Total Issues**
16 (2 Resolved)

<table>
<thead>
<tr>
<th>High Risk Issues</th>
<th>1 (0 Resolved)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium Risk Issues</td>
<td>2 (0 Resolved)</td>
</tr>
<tr>
<td>Low Risk Issues</td>
<td>7 (2 Resolved)</td>
</tr>
<tr>
<td>Informational Risk Issues</td>
<td>5 (0 Resolved)</td>
</tr>
<tr>
<td>Undetermined Risk Issues</td>
<td>1 (0 Resolved)</td>
</tr>
</tbody>
</table>

**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

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

```solidity
uint8 num_players = 0; num_players = num_players - 1; // 0 - 1 now equals 255!
```

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 L492:
   ```solidity
   for (uint i = k; i >= 0; i--) {
   ```

2. [Acknowledged] There is a subtraction 100000 - _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 foo-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, 1, 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 yAxisMetaVault.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:

```solidity
address swapFromToken = BaseToken(tokenToSwap).address;
... // similar code...
```

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
2. The `yAxisMetaVaultHarvester.addStrategy` 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 (occasionally 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: we have consolidated these functions to the `StrategyControllerV2.addStrategy` method, as well as the controller’s `setStrategy` method.

### QSP-8 Gas Usage / Loop Concerns

**Severity:** Low Risk

**Status:** Acknowledged

**File(s) affected:** `yAxisMetaVault.sol`, `StrategyControllerV2.sol`, `BaseStrategy.sol`, `StrategyCurve3Crv.sol`, `StrategyDforce.sol`, `StrategyPickle3Crv.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` function could increase that number. If the number becomes too large then the function could 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]` if `_index < 4.

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

### 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. `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. `yAxisMetaVault.setGovernance` does not require the value of `_governance` to be different from `address(0)` and different from the current governance address.

3. `yAxisMetaVault.setController` does not require the value of `_controller` to be different from `address(0)` and different from the current controller address.
Update from dev team:

- [Acknowledged] `yAxisMetaVault.setConverter` does not require the value of `_converter` to be different from `address(0)` and different from the current `converter` address.
- [Acknowledged] `yAxisMetaVault.setVaultManager` does not require the value of `_vaultManager` to be different from `address(0)` and different from the current `vaultManager` address.
- [Acknowledged] `yAxisMetaVault.setYaxPerBlock` does not require the value of `_yaxPerBlock` to be greater than 0.
- [Acknowledged] `yAxisMetaVault.setTreasuryWallet` does not require the value of `_treasuryWallet` to be different from `address(0)` and different from the current `treasuryWallet` address.
- [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.
- [Acknowledged] `yAxisMetaVault.setConverter` does not check if `_strategy1 != _strategy2`.

**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`
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`, `Skin`, 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`, `setPickleMasterChef`, and `setPoolId`.

**QSP-11 Block Timestamp Manipulation**

**Severity:** Informational

**Status:** Acknowledged

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

**Related issue(s):** `SWC-94`

**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.camHarvest` function returns `true` or `false` based on the values of `block.timestamp`.
2. The `StrategyStabilize.calculateZPATokenWithdrawFee` function 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 998 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 contract 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 are other important roles, which will be described next. The `governance` address has many privileges in the `yAxisMetaVault`, `yAxisMetaVaultManager`, `yAxisMetaVaultHarvester` 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`, epochRewardMultiples, `epochEndBlocks`, `yaxPerBlock`, totalDepositCap, `vaultManager`, `converter`, `controller`, governance, min, `insuranceFee`, `insurancePoolFee`, `stakingPoolShareFee`, strategist, `treasury`, `treasuryFee`, `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 `routier` 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 `claimInsurance()` 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
Adherence to Best Practices

Properly commented (e.g., 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 contracts in the code base. Counting the number of lines of code versus the number of lines with comments in the

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

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

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](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`).

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 6 times in yAxisMetaVault.sol in several functions.
   - \(1000\) is used once in StrategyControllerV2.sol in withdrawFee.
   - \(18**12\) 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 serve as debugging aids for users and developers. There are several instances of error messages in the yAxisVaultHarvesters, StrategyControllerV2 contracts where the error message is simply the a word or function name prefixed by an exclamation mark, e.g. `!harvester`, `!callHarvest`. These error messages should be changed to descriptive sentences.

6. The yAxisMetaVault contract code estimate surpasses 2\(\times\)1076 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://medium.com/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.
✓ 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 (40ms)
✓ 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 (48ms)
✓ 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 rotate harvesting strategies (197ms)
✓ should not allow harvestNextStrategy until timeout has passed (198ms)
✓ 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.

<table>
<thead>
<tr>
<th>File</th>
<th>% Stmts</th>
<th>% Branch</th>
<th>% Funcs</th>
<th>% Lines</th>
<th>Uncovered Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>metavault/</td>
<td>80.94</td>
<td>55.8</td>
<td>78.82</td>
<td>81.63</td>
<td></td>
</tr>
<tr>
<td>IController.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IConverter.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IHarvester.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IMetaVault.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IStableSwap3Pool.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IStrategy.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IStrategyControllerConverter.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>ISwap.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IVaultManager.sol</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>StableSwap3PoolConverter.sol</td>
<td>85.07</td>
<td>70.83</td>
<td>73.33</td>
<td>85.07</td>
<td>... 168,171,179</td>
</tr>
<tr>
<td>yAxisMetaVault.sol</td>
<td>72.69</td>
<td>48.97</td>
<td>65</td>
<td>75.42</td>
<td>... 487,488,489</td>
</tr>
<tr>
<td>yAxisMetaVaultHarvester.sol</td>
<td>100</td>
<td>92.86</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>yAxisMetaVaultManager.sol</td>
<td>97.83</td>
<td>85.71</td>
<td>100</td>
<td>97.83</td>
<td>69</td>
</tr>
<tr>
<td>metavault/contracts/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StrategyControllerV1.sol</td>
<td>91.38</td>
<td>77.27</td>
<td>79.31</td>
<td>92.06</td>
<td></td>
</tr>
<tr>
<td>StrategyControllerV2.sol</td>
<td>91.38</td>
<td>77.27</td>
<td>79.31</td>
<td>92.06</td>
<td>... 480,498,509</td>
</tr>
<tr>
<td>metavault/strategies/</td>
<td>95.34</td>
<td>54.08</td>
<td>93.26</td>
<td>95.34</td>
<td></td>
</tr>
<tr>
<td>BaseStrategy.sol</td>
<td>100</td>
<td>78.59</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>StrategyCurve3Crv.sol</td>
<td>90.38</td>
<td>42.31</td>
<td>100</td>
<td>90.38</td>
<td>93,94,97,98,101</td>
</tr>
<tr>
<td>StrategyDforce.sol</td>
<td>96.15</td>
<td>57.14</td>
<td>87.5</td>
<td>96</td>
<td>67</td>
</tr>
<tr>
<td>StrategyFlameIncome.sol</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>StrategyGenericVault.sol</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>StrategyIdle.sol</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>StrategyPickle3Crv.sol</td>
<td>84.38</td>
<td>50</td>
<td>72.73</td>
<td>84.38</td>
<td>... 9,93,94,152</td>
</tr>
<tr>
<td>StrategyStabilize.sol</td>
<td>98.25</td>
<td>53.85</td>
<td>100</td>
<td>98.25</td>
<td>78</td>
</tr>
<tr>
<td>StrategyYearnV2.sol</td>
<td>96.15</td>
<td>58.33</td>
<td>85.71</td>
<td>96.15</td>
<td>59</td>
</tr>
<tr>
<td>StrategydYdXSoloMargin.sol</td>
<td>93.94</td>
<td>50</td>
<td>85.71</td>
<td>93.94</td>
<td>93,144</td>
</tr>
<tr>
<td>All files</td>
<td>88.9</td>
<td>57.11</td>
<td>85.22</td>
<td>89.32</td>
<td></td>
</tr>
</tbody>
</table>

**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

4d42b55c6f65117e799fb3b162cb8bb2ba2e7756b2626b065fe0d885955ceb ./contracts/metavault/IStableSwap3Pool.sol
f5f8adbb145a3e36386a299353e8fb3e42a44be09b9e95d16a52a ./contracts/metavault/StableSwap3PoolConverter.sol
6235b87e9b50d36614d72115fd796640ac9b05cbb3d9559cbb430c817ca9ec12c ./contracts/metavault/IStrategyControllerConverter.sol
546d8ada5e488c13d15b1b59800b548b57ba7b058c095ae2f48571580f179 ./contracts/metavault/yAxisMetaVaultHarvester.sol
8ed46e88688c18d5e60a59f17866d264453ec454ce8b38c8e7d6df5ed9a96e6 ./contracts/metavault/ISwap.sol
70c1be9bfe2249d37bac2c564ac1b54bc8ad2e79ad6e9a366a3a2d3f2f188b3 ./contracts/metavault/yAxisMetaVaultManager.sol
3748519e543fa9a5b8b4a4c39a175a335e541a26dd1f2e8d099b9c95cfca ./contracts/metavault/IVaultManager.sol
b311ffdd071bedcfc626f3914b1656ee0f0a56844f48e6188c8f81add6abe7 ./contracts/metavault/IStrategy.sol
b63ffba776494ebba8b48e46478f0f5c5134f1e31737b2059016733905d177cf5e ./contracts/metavault/IVaultManager.sol
• 2021-02-09 - Updated report based on commit 3538b8a8c1ea4ec2b68a635c48b938a19a1c26b2c6
• 2021-02-06 - Initial report based on commit 3538b8a8c1ea4ec2b68a635c48b938a19a1c26b2c6
• 2021-02-03 - Updated report based on commit d743567bb1b4df5604c22a3f114766a8f67b0d9
• 2021-02-02 - Updated report based on commit e61d1c3ae1b8b126c785658988512944e1877
About Quantstamp

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

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

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

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

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such websites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.