yAxis Process Quality Review

Score: 37%

Overview

This is a Process Quality Review of yAxis completed on May 13th 2021. It was performed using the Process Review process (version 0.7) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is 37%, a fail. The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is 70%.

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

Disclaimer

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

Chain

This section indicates the blockchain used by this protocol.

✓ **Chain: Ethereum**

Guidance:
Ethereum
Binance

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is here. This review will answer the questions;

1) Are the executing code addresses readily available? (%)
2) Is the code actively being used? (%)
3) Is there a public software repository? (Y/N)
4) Is there a development history visible? (%)
5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

✓ Answer: 100%

They are available at website https://github.com/yaxis-project/metavault as indicated in the Appendix.

Guidance:

100%	Clearly labelled and on website, docs or repo, quick to find
70%	Clearly labelled and on website, docs or repo but takes a bit of looking
40%	Addresses in mainnet.json, in discord or sub graph, etc
20%	Address found but labelling not clear or easy to find
0%	Executing addresses could not be found

2) Is the code actively being used? (%)

✓

Answer: 100%

Activity is 50 transactions a day on contract *RewardsMetavault.sol*, as indicated in the Appendix.

Percentage Score Guidance

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

3) Is there a public software repository? (Y/N)

Answer: Yes

GitHub: https://github.com/yaxis-project/metavault

Is there a public software repository with the code at a minimum, but normally test and scripts also (Y/N). Even if the repo was created just to hold the files and has just 1 transaction, it gets a Yes. For teams with private repos, this answer is No.

4) Is there a development history visible? (%)

Answer: 100%

yAxis' MetaVault has 223 commits and 2 branches in their software repo which is a healthy software development history.

This checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:

100%	Any one of 100+ commits, 10+branches
70%	Any one of 70+ commits, 7+branches
50%	Any one of 50+ commits, 5+branches
30%	Any one of 30+ commits, 3+branches
0%	Less than 2 branches or less than 10 commits

5) Is the team public (not anonymous)? (Y/N)

Answer: No

yAxis team is anonymous.

For a yes in this question the real names of some team members must be public on the website or other documentation. If the team is anonymous and then this question is a No.

Documentation

This section looks at the software documentation. The document explaining these questions is here.

Required questions are;

6) Is there a whitepaper? (Y/N)
7) Are the basic software functions documented? (Y/N)
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)
10) Is it possible to trace from software documentation to the implementation in
code (%)

6) Is there a whitepaper? (Y/N)

✓ Answer: Yes

Location: https://yaxis.ghost.io/intro/

7) Are the basic software functions documented? (Y/N)

⚠ Answer: No

There is no apparent software function documentation.

How to improve this score

Write the document based on the deployed code. For guidance, refer to the SecurEth System Description Document.

8) Does the software function documentation fully (100%) cover the deployed contracts? (%)

⚠ Answer: 0%

There is no evident software function documentation.

Guidance:

100%	All contracts and functions documented
80%	Only the major functions documented
79-1%	Estimate of the level of software documentation
0%	No software documentation

How to improve this score

This score can improve by adding content to the requirements document such that it comprehensively covers the requirements. For guidance, refer to the SecurEth System Description Document . Using tools that aid traceability detection will help.

9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)

Answer: 27%

Code examples are in the Appendix. As per the SLOC, there is 27% commenting to code (CtC).

The Comments to Code (CtC) ratio is the primary metric for this score.

Guidance:

100%	CtC > 100	Useful comments consistently on all code
90-70%	CtC > 70	Useful comment on most code
60-20%	CtC > 20	Some useful commenting
0%	CtC < 20	No useful commenting

How to improve this score

This score can improve by adding comments to the deployed code such that it comprehensively covers the code. For guidance, refer to the SecurEth Software Requirements.

10) Is it possible to trace from software documentation to the implementation in code (%)

Answer: 0%

There is no software function documentation.

Guidance:

100%	Clear explicit traceability between code and documentation at a requirement level for all code
60%	Clear association between code and documents via non explicit traceability
40%	Documentation lists all the functions and describes their functions
0%	No connection between documentation and code

How to improve this score

This score can improve by adding traceability from requirements to code such that it is clear where each requirement is coded. For reference, check the SecurEth guidelines on traceability.

Testing

This section looks at the software testing available. It is explained in this document. This section answers the following questions;

11) Full test suite (Covers all the deployed code) (%)
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)
13) Scripts and instructions to run the tests (Y/N)
14) Report of the results (%)
15) Formal Verification test done (%)
16) Stress Testing environment (%)

11) Is there a Full test suite? (%)

Answer: 40%

TtC ratio is 64%.

This score is guided by the Test to Code ratio (TtC). Generally a good test to code ratio is over 100%. However the reviewers best judgement is the final deciding factor.

Guidance:

100%	TtC > 120%	Both unit and system test visible
80%	TtC > 80%	Both unit and system test visible
40%	TtC < 80%	Some tests visible
0%		No tests obvious

How to improve this score

This score can improve by adding tests to fully cover the code. Document what is covered by traceability or test results in the software repository.

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

Answer: 88%

Code coverage done in their Quantstamp audit.

Guidance:

100%	Documented full coverage
99-51%	Value of test coverage from documented results
50%	No indication of code coverage but clearly there is a reasonably complete set of tests
30%	Some tests evident but not complete
0%	No test for coverage seen

How to improve this score

This score can improve by adding tests achieving full code coverage. A clear report and scripts in the software repository will guarantee a high score.

13) Scripts and instructions to run the tests (Y/N)

⚠ Answer: No

There are no visible instructions to run the tests on their GitHub.

How to improve this score

Add the scripts to the repository and ensure they work. Ask an outsider to create the environment and run the tests. Improve the scripts and docs based on their feedback.

14) Report of the results (%)

⚠ Answer: 0%

There are no visible test results reports.

Guidance:

100%	Detailed test report as described below
70%	GitHub Code coverage report visible
0%	No test report evident

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

15) Formal Verification test done (%)

⚠ Answer: 0%

No formal verification of yAxis appears in web searches.

16) Stress Testing environment (%)

> ✓ Answer: 100%

Stress testing addresses available at https://github.com/yaxis-project/metavault.

Security

This section looks at the 3rd party software audits done. It is explained in this document. This section answers the following questions;

17) Did 3rd Party audits take place? (%)
18) Is the bounty value acceptably high?

17) Did 3rd Party audits take place? (%)

> ! Answer: 50%

yAxis has had a Quantstamp audit and Haechi audit on Feb. 19th 2021 for their MetaVault V2 which was launched on Feb. 22nd 2021. Clearly the Quantstamp audit was done after deployment, found real concerning weakness which were not addressed by re-deployment. For this reason the 70% score is reduced to 50%.

Guidance:	
100%	Multiple Audits performed before deployment and results public and implemented or not required
90%	Single audit performed before deployment and results public and implemented or not required
70%	Audit(s) performed after deployment and no changes required. Audit report is public
20%	No audit performed
0%	Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, question

18) Is the bounty value acceptably high (%)

> ! Answer: 40%

Bug Bounty Location: https://immunefi.com/bounty/yaxis/

Guidance:

100%	Bounty is 10% TVL or at least $1M AND active program (see below)
90%	Bounty is 5% TVL or at least 500k AND active program
80%	Bounty is 5% TVL or at least 500k
70%	Bounty is 100k or over AND active program
50%	Bounty is 100k or over
40%	Bounty is 50k or over
20%	Bug bounty program bounty is less than 50k
0%	No bug bounty program offered

Active program means a third party actively driving hackers to the site. Inactive program would be static mention on the docs.

Access Controls

This section covers the documentation of special access controls for a DeFi protocol. The admin access controls are the contracts that allow updating contracts or coefficients in the protocol. Since these contracts can allow the protocol admins to "change the rules", complete disclosure of capabilities is vital for user's transparency. It is explained in this document. The questions this section asks are as follow;

19) Can a user clearly and quickly find the status of the admin controls?
20) Is the information clear and complete?
2`) Is the information in non-technical terms that pertain to the investments?
22) Is there Pause Control documentation including records of tests?

19) Can a user clearly and quickly find the status of the admin controls (%)

⚠ Answer: 0%

Location: Could not be found in protocol docs, but the Haechi audit (page 8) said the devs have effective full control over users funds.

Guidance:

100%	Clearly labelled and on website, docs or repo, quick to find
70%	Clearly labelled and on website, docs or repo but takes a bit of looking
40%	Access control docs in multiple places and not well labelled
20%	Access control docs in multiple places and not labelled
0%	Admin Control information could not be found

20) Is the information clear and complete (%)

⚠ Answer: 0%

Guidance:
All the contracts are immutable -- 100% OR

All contracts are clearly labelled as upgradeable (or not) -- 30% AND
The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
The capabilities for change in the contracts are described -- 30%

How to improve this score

Create a document that covers the items described above. An example is enclosed.

21) Is the information in non-technical terms that pertain to the investments (%)

> ⚠ Answer: 0%

Guidance:

100%	All the contracts are immutable
90%	Description relates to investments safety and updates in clear, complete non-software l language
30%	Description all in software specific language
0%	No admin control information could not be found

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

22) Is there Pause Control documentation including records of tests (%)

> ⚠ Answer: 0%

Guidance:

100%	All the contracts are immutable or no pause control needed and this is explained OR
100%	Pause control(s) are clearly documented and there is records of at least one test within 3 months
80%	Pause control(s) explained clearly but no evidence of regular tests
40%	Pause controls mentioned with no detail on capability or tests
0%	Pause control not documented or explained

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

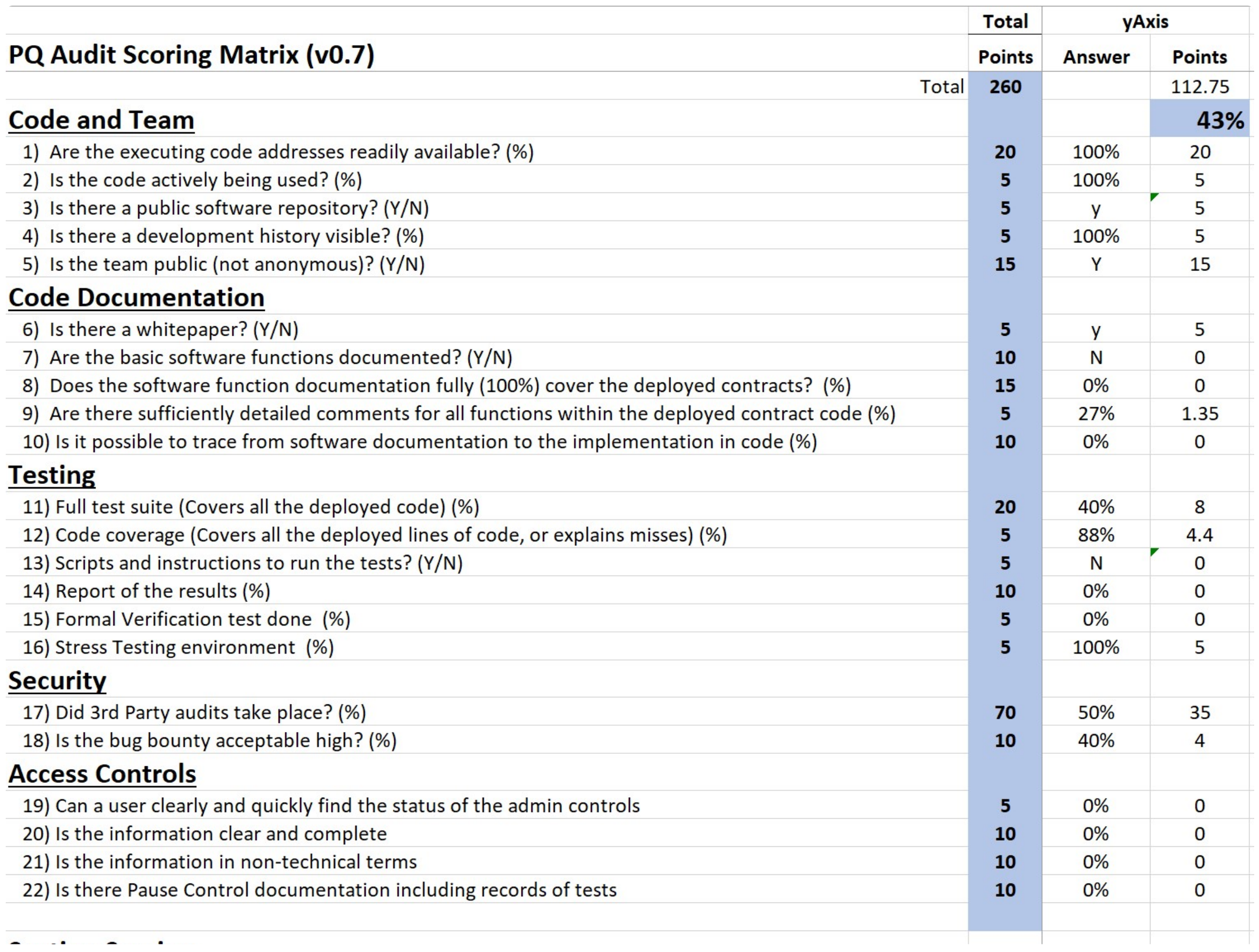
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started SecuEth.org with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got EthFoundation funding to assist in their development.

Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix



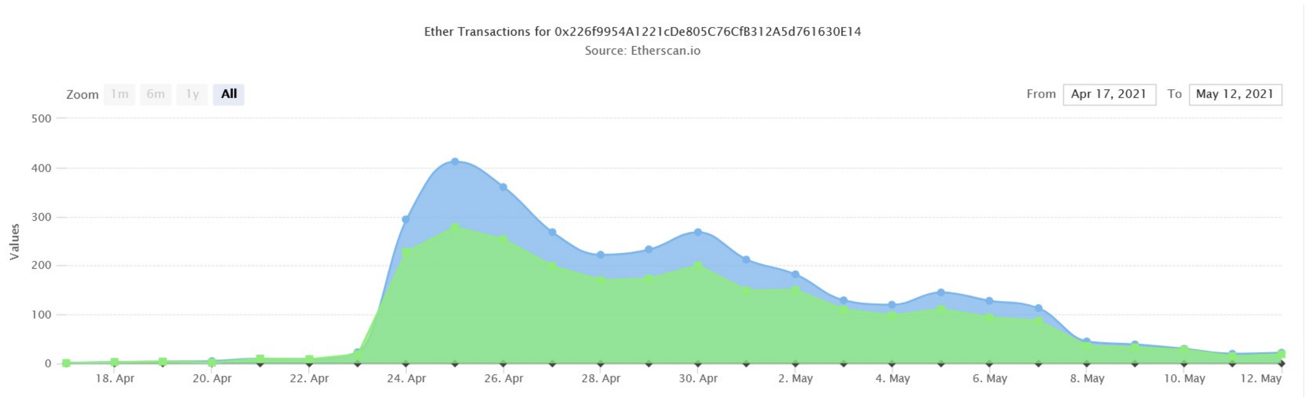
	Total	yAxis	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		112.75
Code and Team			**43%**
1) Are the executing code addresses readily available? (%)	**20**	100%	20
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	y	5
4) Is there a development history visible? (%)	**5**	100%	5
5) Is the team public (not anonymous)? (Y/N)	**15**	Y	15
Code Documentation			
6) Is there a whitepaper? (Y/N)	**5**	y	5
7) Are the basic software functions documented? (Y/N)	**10**	N	0
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	**15**	0%	0
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	27%	1.35
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	40%	8
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	88%	4.4
13) Scripts and instructions to run the tests? (Y/N)	**5**	N	0
14) Report of the results (%)	**10**	0%	0
15) Formal Verification test done (%)	**5**	0%	0
16) Stress Testing environment (%)	**5**	100%	5
Security			
17) Did 3rd Party audits take place? (%)	**70**	50%	35
18) Is the bug bounty acceptable high? (%)	**10**	40%	4
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	0%	0
20) Is the information clear and complete	**10**	0%	0
21) Is the information in non-technical terms	**10**	0%	0
22) Is there Pause Control documentation including records of tests	**10**	0%	0

Section Scoring		
Code and Team	50	100%
Documentation	45	14%
Testing	50	35%
Security	80	49%
Access Controls	35	0%

Executing Code Appendix

```
RewardsMetavault 0x226f9954A1221cDe805C76CfB312A5d761630E14
RewardsYaxis 0x3b09B9ADFe11f92225b4C55De89fa81456595CD9
RewardsYaxisEth 0xEDaFe410e2f07ab9D7F1B04316D29C2F49dCb104
StableSwap3PoolConverter 0x2eab685d85AA52E4d8b6699Ba5aAC3b0c3992C3B
StableSwap3PoolNonConverter 0x9354b082E5CEdb41422Fb9D4669Ab8b7f8511AeE
StrategyControllerV2 0x0d857688d6A223A2F4e58CDd44119ABb7DC5A790
StrategyCurve3Crv 0x5cd9D7977F9e431399E8186339D9ecBf88eD43F2
StrategyYearnV2-DAI 0xb6C352587F4A92D3c7946bf42fE6D4D3aCd1F312
Swap 0xCdF398537adbF8617a8401B14DCEe7F67CF8c64b
YaxisToken 0x0adA190c81b814548ddC2F6AdC4a689ce7C1FE73
YaxisVotePower 0x01FEF0d5d6Fd6B5701aE913CaFb11dDaeE982C9A
yAxisMetaVaultHarvester 0x5BBc6Ff70680d1DfEFd4685CbdeD5363A4db9b66
yAxisMetaVaultManager 0x443ed48F975E02eA67CA0d2be0B4d4806d1E31F2
```

Code Used Appendix



Example Code Appendix

```
// SPDX-License-Identifier: MIT

pragma solidity 0.6.12;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/math/SafeMath.sol";
import "@openzeppelin/contracts/utils/Address.sol";
import "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";

import "../IStableSwap3Pool.sol";
import "../ISwap.sol";
import "../IVaultManager.sol";
import "../IStrategy.sol";
import "../IController.sol";

/**
 * @title BaseStrategy
 * @notice The BaseStrategy is an abstract contract which all
 * yAxis strategies should inherit functionality from. It gives
 * specific security properties which make it hard to write an
 * insecure strategy.
 * @notice All state-changing functions implemented in the strategy
 * should be internal, since any public or externally-facing functions
 * are already handled in the BaseStrategy.
 * @notice The following functions must be implemented by a strategy:
 * - function _deposit() internal virtual;
 * - function _harvest() internal virtual;
 * - function _withdraw(uint256 _amount) internal virtual;
 * - function _withdrawAll() internal virtual;
 * - function balanceOfPool() public view override virtual returns (uint256);
 */
abstract contract BaseStrategy is IStrategy {
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;

    uint256 public constant ONE_HUNDRED_PERCENT = 10000;

    address public immutable override want;
    address public immutable weth;
    address public immutable controller;
    IVaultManager public immutable vaultManager;

    string public override name;
    ISwap public router;

    event ApproveForSpender(address token, address spender, uint256 amount);
    event SetController(address controller);
    event SetRouter(address router);
    event Skim();
    event Withdraw(address vault, uint256 amount);

    /**
     * @param _controller The address of the controller
     * @param _vaultManager The address of the vaultManager
     * @param _want The desired token of the strategy
     * @param _weth The address of WETH
     * @param _router The address of the router for swapping tokens
     */
    constructor(
        string memory _name,
        address _controller,
        address _vaultManager,
        address _want,
        address _weth,
        address _router
    ) public {
        require(_controller != address(0), "!_controller");
        require(_vaultManager != address(0), "!_vaultManager");
        require(_want != address(0), "!_want");
        require(_weth != address(0), "!_weth");
        require(_router != address(0), "!_router");
        name = _name;
        want = _want;
        controller = _controller;
        vaultManager = IVaultManager(_vaultManager);
        weth = _weth;
        router = ISwap(_router);
        IERC20(_weth).safeApprove(address(_router), type(uint256).max);
    }

    /**
     * GOVERNANCE-ONLY FUNCTIONS
     */

    /**
     * @notice Approves a token address to be spent by an address
     * @param _token The address of the token
     * @param _spender The address of the spender
     * @param _amount The amount to spend
     */
    function approveForSpender(IERC20 _token, address _spender, uint256 _amount) external
        require(msg.sender == vaultManager.governance(), "!governance");
        _token.safeApprove(_spender, _amount);
        emit ApproveForSpender(address(_token), _spender, _amount);
    }

    /**
     * @notice Sets the address of the ISwap-compatible router
     * @param _router The address of the router
     */
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	81	6415	833	1217	4365	439

Comments to Code / = 27%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	25	3139	307	17	2815	5

Tests to Code 2815/4365 = 64%