

# 0.7

## Beefy.Finance Process Quality Review

Score: 53%

### Overview

This is a Process Quality Review completed on June 1st 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 53%, 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

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such

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.

This completed report is copyright (c) DeFiSafety 2021. Permission is given to copy in whole, retaining this copyright label.

## Chain

This section indicates the blockchain used by this protocol.

 **Chain: Binance Smart Chain**

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: 70%

They are available at website <https://github.com/beefyfinance/beefy-protocol> 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                                   |

How to improve this score

Make the Ethereum addresses of the smart contract utilized by your application available on either your website or your GitHub (in the README for instance). Ensure the addresses is up to date. This is a very important question wrt to the final score.

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

 Answer: 100%

Activity is 400 transactions a day on contract *RewardPool.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/beefyfinance>

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: 20%

With 19 commits and 1 branch in their smart contracts repository, this is an unhealthy software repository. All the updated contracts are currently being developed in a private repository.

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

How to improve this score

Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools. A public development history indicates clearly to the public the level of continued investment and activity by the developers on the application. This gives a level of security and faith in the application.

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

 Answer: No

Beefy.Finance teams consists of anonymous devs.

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://docs.beefy.finance/beefyfinance/>

How to improve this score

Ensure the white paper is available for download from your website or at least the software repository. Ideally update the whitepaper to meet the capabilities of your present application.

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

 Answer: No

No basic smart contract function documentation was found.

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%

No software documentation was found.

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: 100%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 122% 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 evident.

#### 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: 0%

No testing suite in their smart contract repository.

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: 0%

No test for coverage in their documentation or audits.

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

No scripts or instructions to run smart contract tests.

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%

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 evidence of a Beefy.Finance formal verification was found.

#### **16) Stress Testing environment (%)**

 Answer: 0%

No testnet documentation was found.

---

## 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: 100%

Certik did a Beefy.Finance preliminary comments audit on May 11th 2021.

Certik did a security assessment on Beefy.Finance contracts on March 5th 2021.

DefiYield did a Beefy.Finance audit on December 10th 2021.

Beefy.Finance was launched on September 16th 2020.

Note: Changes in 2nd audit were implemented before deployment.

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: 0%

Bug bounty program was announced for early May, but never launched.

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?
- 21) 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: 40%

Found at <https://docs.beefy.finance/beefyfinance/faq/products/strategies>

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: 100%

All contracts are immutable as described at <https://docs.beefy.finance/beefyfinance/faq/products/strategies>

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: 100%

All the contracts are described as immutable at

<https://docs.beefy.finance/beefyfinance/faq/products/strategies>

Guidance:

- 100% All the contracts are immutable
- 90% Description relates to investments safety and updates in clear, complete non-software 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%

All Contracts are Immutable.

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	Beefy Finance	
	Points	Answer	Points
<b>PQ Audit Scoring Matrix (v0.7)</b>	Total	260	137
<b>Code and Team</b>			53%
1) Are the executing code addresses readily available? (%)	20	70%	14
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	20%	1
5) Is the team public (not anonymous)? (Y/N)	15	N	0
<b>Code Documentation</b>			
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	100%	5
10) Is it possible to trace from software documentation to the implementation in code (%)	10	0%	0
<b>Testing</b>			
11) Full test suite (Covers all the deployed code) (%)	20	0%	0
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	0%	0
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	0%	0
<b>Security</b>			
17) Did 3rd Party audits take place? (%)	70	100%	70
18) Is the bug bounty acceptable high? (%)	10	0%	0
<b>Access Controls</b>			
19) Can a user clearly and quickly find the status of the admin controls	5	40%	2
20) Is the information clear and complete	10	100%	10
21) Is the information in non-technical terms	10	100%	10
22) Is there Pause Control documentation including records of tests	10	100%	10
<b>Section Scoring</b>			
Code and Team	50	50%	
Documentation	45	22%	
Testing	50	0%	
Security	80	88%	
Access Controls	35	91%	

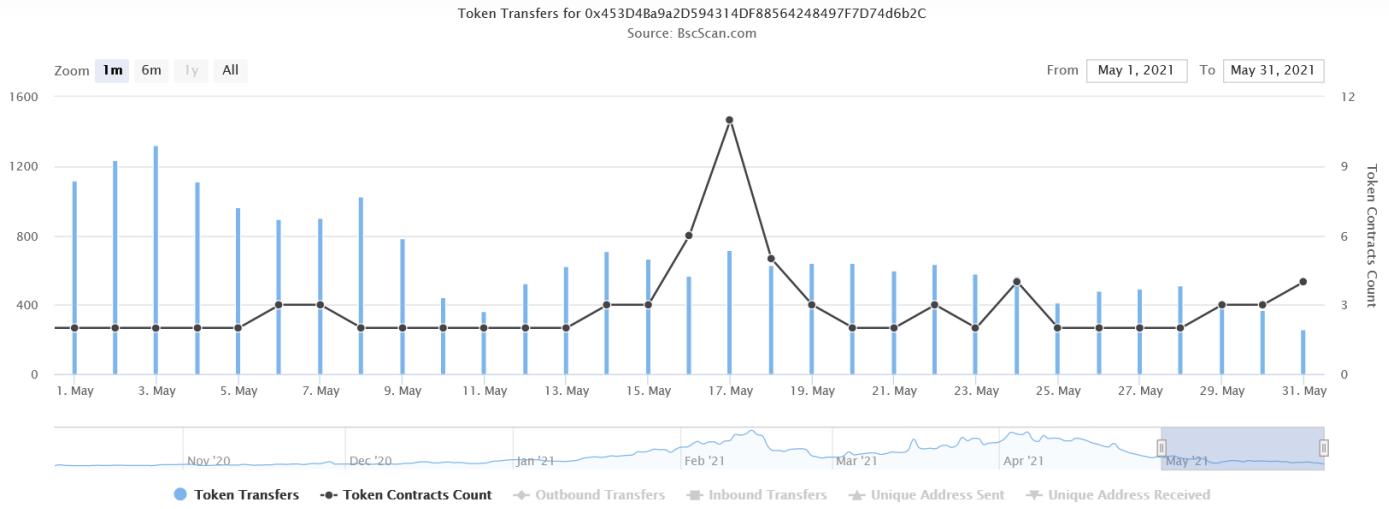
## Executing Code Appendix

```

1  export const beefyfinance = {
2    multisig: "0x8f0fFc8C7FC3157697Bdbf94B328F7141d6B41de",
3    keeper: "0xd529b1894491a0a26B18939274ae8ede93E81dbA",
4    rewardPool: "0x453D4Ba9a2D594314DF88564248497F7D74d6b2C",
5    treasury: "0x4A32De8c248533C28904b24B4cFCFE18E9F2ad01",
6    beefyFeeRecipient: "0xEB41298BA4Ea3865c33bDE8f60eC414421050d53",

```

## Code Used Appendix



## Example Code Appendix

```

1 pragma solidity ^0.5.0;
2
3 import "@openzeppelin/contracts/math/Math.sol";
4 import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
5
6 import "./IRewardDistributionRecipient.sol";
7 import "./LPTokenWrapper.sol";
8
9 contract RewardPool is LPTokenWrapper, IRewardDistributionRecipient {
10     IERC20 public yfi = IERC20(0x000000000000000000000000000000000000000000000000000000000000000);
11     uint256 public constant DURATION = 60 days;
12
13     uint256 public periodFinish = 0;
14     uint256 public rewardRate = 0;
15     uint256 public lastUpdateTime;
16     uint256 public rewardPerTokenStored;
17     mapping(address => uint256) public userRewardPerTokenPaid;
18     mapping(address => uint256) public rewards;
19
20     event RewardAdded(uint256 reward);
21     event Staked(address indexed user, uint256 amount);
22     event Withdrawn(address indexed user, uint256 amount);
23     event RewardPaid(address indexed user, uint256 reward);
24
25     modifier updateReward(address account) {
26         rewardPerTokenStored = rewardPerToken();
27         lastUpdateTime = lastTimeRewardApplicable();
28         if (account != address(0)) {
29             rewards[account] = earned(account);
30             userRewardPerTokenPaid[account] = rewardPerTokenStored;
31         }
32     ;
33 }
34
35     function lastTimeRewardApplicable() public view returns (uint256) {
36
37 }
```

```

36         return Math.min(block.timestamp, periodFinish);
37     }
38
39     function rewardPerToken() public view returns (uint256) {
40         if (totalSupply() == 0) {
41             return rewardPerTokenStored;
42         }
43         return
44             rewardPerTokenStored.add(
45                 lastTimeRewardApplicable()
46                     .sub(lastUpdateTime)
47                     .mul(rewardRate)
48                     .mul(1e18)
49                     .div(totalSupply())
50             );
51     }

```

## SLOC Appendix

### Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	17	7329	1060	3439	2830	279

Comments to Code 3439/2830 = 122%

### Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	0	0	0	0	0	0

Tests to Code 0/0 = 0%