0.8

0.8 Blank

Overview

This is a ___ Process Quality Review completed on the _ of __, 202_. It was performed using the Process Review process (version 0.8) and is documented here. The review was performed by ______ of DeFiSafety. Check out our Telegram.

The final score of the review is ___%, a ______. 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**
- **This is where the protocol gets its data, and these are some proactive steps we've taken to avoid problems.**

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

Chain

This section indicates the blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

✓ **Chain:**

Guidance:

Ethereum

Binance Smart Chain

Polygon

Avalanche

Terra

Celo

Arbitrum

Solana

Code and Team

This section looks at the code deployed on the relevant chain that gets reviewed and its corresponding software repository. The document explaining these questions is here. This section will answer the following questions:

1) Are the smart contracts easy to find? (%)

2) How active is the primary contract? (%)

3) Does the protocol have a public GitHub repository? (Y/N)

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

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

1) Are the smart contract addresses easy to find?(%)

✓ **Answer:**

They can be found at ___ , as indicated in the Appendix.

Guidance:

100% Clearly labelled and on website, documents or repository, quick to find

70% Clearly labelled and on website, documents or repository 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 addresses of the smart contracts utilized by your application available on either your website, Gitbook or GitHub (in the README for instance). Ensure the addresses are up to date, and that they can be verified using a block explorer. Your documentation should ideally comprise a section called “Smart Contracts”, “Smart Contract Addresses”, or “Deployed Addresses” in order to make them easy to find.

2) How active is the primary contract? (%)

✓ **Answer:**

Contract ___ is used ___ times a day, as indicated in the Appendix.

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

How to improve this score:

Make sure to list your main executing smart contract addresses in a section of your documentation. This will make it easier for us to evaluate your contract’s user interactions, as well as make it easier for you to get 100% for this question.

3) Does the protocol have a public software repository? (Y/N)

✓ **Answer:**

Software repository:

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

How to improve this score

Ensure your contracts are available for viewing on a public software repository (like GitHub). The link to it can be from your protocol's website or GitBook documentation.

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

✓ **Answer:**

This metric 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 30 commits

How to improve this score:

In order to maximize your score for this question, make sure to actively develop your public GitHub. This doesn't mean that you cannot also have a private GitHub, but your users should at least be able to see a decent amount of development history in it. A good way to rack up the commits is to also add your testing suite to your GitHub, and then continuously test for bugs which would go straight into your CI.

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

✓ **Answer:**

bah bah bah bah bah bign bing bong please don't fuck me gitbook i want to kill you

Where we found the team is documented in our team appendix.

Guidance:

100%: At least two names can be easily found in the protocol's website, documentation or medium. These are then confirmed by the personal websites of the individuals / their linkedin / twitter.

50%: At least one public name can be found to be working on the protocol.

0%: No public team members could be found.

How to improve this score:

Create a section of documentation linking to employees, founders or contributors in an easily accessible place such as a website, LinkedIn etc. To score fully, name the contributors to this protocol in your documentation and ensure that this corroborates information that can be found elsewhere on the internet (e.g. LinkedIn/ a personal website). Alternatively, software repository contributors can be public.

Documentation

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

For this section, the questions are as follows

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

7) Is the protocol software architecture documented? (Y/N)

8) Does the software documentation fully cover the deployed contracts' source code? (Y/N)

9) Is it possible to trace the documented software to its implementation in the protocol's source code? (%)

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

✓ **Answer:**

Location:

How to improve this score:

Ensure that your white paper is available for viewing from your website's front page, GitBooks documentation, or at least the README.md of your GitHub repository. Moreover, protocols should update their whitepapers frequently to meet the capabilities of their present application.

7) Is the protocol's software architecture documented? (Y/N)

✓ **Answer:**

This protocol's software architecture is documented in _____.

Guidance:

Yes: the documents identify software architecture and contract interaction through any of the following: diagrams, arrows, specific reference to software functions or a written explanation on how smart contracts interact. Protocols receive a **No** If none of these are included.

How to improve this score:

Write this document based on the deployed code and how it operates. For guidance, refer to the SecurEth System Description Document. This document can be written after deployment, though as with all documentation earlier is better.

8) Does the software documentation fully cover the deployed contracts' source code? (%)

✓ **Answer:**

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 be improved by ensuring protocol documents fully and comprehensively cover everything deployed by the protocol. A good way to do it is to list literally every function for every smart contract that you have deployed for your protocol. Not only that, but you also include a brief description of what the specific contract function does and how it does it. For guidance, refer to the SecurEth System Description Document
.

9) Is it possible to trace the documented software to its implementation in the protocol's source code? (%)

✓ **Answer:**

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 be improved by adding traceability from documents to code such that it is clear where each outlined code function in the documentation is coded in the protocol's GitHub repository. 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;

10)Has the protocol tested their deployed code? (%)

11) How covered is the protocol's code? (%)

12) Does the protocol provide scripts and instructions to run tests? (Y/N)

13) Is there a report of the protocol's test results? (%)

14) 14) Has the protocol undergone formal verification? (Y/N)

15) Were the smart contracts deployed to a testnet? (Y/N)

10) Has the protocol tested their deployed code? (%)

✓ **Answer:**

Code examples are in the Appendix. As per the SLOC, there is ___% testing to code (TtC).

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 be improved by adding tests to fully cover the code. Document what is covered by traceability or test results in the software repository. Ideally, you should have a test file per smart contract deployed on the blockchain. In addition, providing scripts to run your tests is an essential part of testing transparency, and no testing suite should come without them.

11) How covered is the protocol's code? (%)

✓ **Answer:**

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 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 be improved by performing code coverage tests that are as close to 100% coverage as possible. In the event that some lines of code or entire contracts are missed, you should clearly outline why this is the case in your coverage report. Hence, you should also be aiming to perform code coverage tests upon every single deployment. This proves that the code is rigorously tested, and therefore has a degree of reliability attributed to it.

12) Does the protocol provide scripts and instructions to run their tests? (Y/N)

✓ **Answer:**

Scripts/Instructions location:

Guidance:

Yes - Scripts and/or instructions to run tests are available in the testing suite

No - Scripts and/or instructions to run tests are not available in the testing suite

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.

13) Is there a detailed report of the protocol's test results?(%)

✓ **Answer:**

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 code coverage test report with the results. This should not only be a code coverage output, but rather a combination of your coverage output and a deeper insight on your methodology used. An exemplary test report from Balancer Finance can be found here.

14) Has the protocol undergone Formal Verification? (Y/N)

✓ **Answer:**

Guidance:

Yes - Formal Verification was performed and the report is readily available

No - Formal Verification was not performed and/or the report is not readily available.

How to improve this score

Undergo a Formal Verification for your protocol's algorithms, and use the services provided by the reputable formal verifiers in the space such as Certora and Runtime Verification. Although this can become expensive, it is an essential part of legitimizing your software's integrity.

15) Were the smart contracts deployed to a testnet? (Y/N)

✓ **Answer:**

Guidance:

Yes (100%) - Protocol has proved their tesnet usage by providing the addresses

No (0%) - Protocol has not proved their testnet usage by providing the addresses

How to improve your score

Make sure to utilize testnet networks in your developing and deployment process. It is an environment that lets you fully test the capability and functionality of your smart contracts. Subsequently, it ensures a certain degree of stability. Finally, don’t forget to link your testnet smart contract addresses in your documentation, as we need proof in order to score this question.

Security

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

16) Is the protocol sufficiently audited? (%)

17) Is the bounty value acceptably high? (%)

16) Is the protocol sufficiently audited? (%)

> ✓ **Answer:**

Guidance:

100% - Multiple Audits performed before deployment and the audit findings are public and implemented or not required

90% - Single audit performed before deployment and audit findings are public and implemented or not required

70% - Audit(s) performed after deployment and no changes required. The Audit report is public.

65% - Code is forked from an already audited protocol and a changelog is provided explaining why forked code was used and what changes were made. This changelog must justify why the changes made do not affect the audit.

50% - Audit(s) performed after deployment and changes are needed but not implemented.

30% - Audit(s) performed are low-quality and do not indicate proper due diligence.

20% - No audit performed

0% Audit Performed after deployment, existence is public, report is not public OR smart contract address' not found.

Deduct 25% if the audited code is not available for comparison.

How to improve your score:

Your score for this question can improve by having your future deployments rigorously audited before deployment. In order to achieve full marks, you must have had your smart contracts audited twice before being deployed. However, even having just one audit performed before deployment gives you a great score for this metric. Another point you can seek to improve on is implementing the recommendations brought to you by the audit. We read every audit thoroughly, and not implementing important fixes will definitely affect your score negatively. Finally, having one audit published either before or after a deployment is infinitely better than having none at all. Even if you have just one audit, you're on the right track. Just make sure that it is traceable to your own code, and that the audited smart contracts have their addresses publicly available.

17) Is the bounty value acceptably high (%)

✓ **Answer:**

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
60%	Bounty is 100k or over
50%	Bounty is 50k or over AND active program
40%	Bounty is 50k or over
20%	Bug bounty program bounty is less than 50k
0%	No bug bounty program offered / the bug bounty program is dead

An active program means that a third party (such as Immunefi) is actively driving hackers to the site. An inactive program would be static mentions on the docs.

How to improve your score

The whole idea of a Bug Bounty program is to increase the amount of eyes that are continuously checking your source code for bugs. How do you improve that? Through improving the monetary incentives of your program. In this scenario, the more incentives you put in for community members to find errors in your code, the more your code will be secure in the long run. The other side of the coin is that posting a million dollar bounty will naturally make you feel more inclined to be absolutely certain that your code does not have any flaws in it. Therefore, Bug Bounties are an incentive for both the users and the developers.

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;

18) Is the access control information easy to find? (%)

19) Are relevant contracts clearly labelled as upgradeable? (%)

20) Is the type of contract ownership clearly indicated? (%)

21) Are the contract change capabilities described? (%)

22) Is the information easy to understand? (%)

23) Is there sufficient Pause Control documentation? (%)

24) Is there sufficient timelock documentation? %

25) Is the TimeLock an adequate length?

18) Is the protocol's admin control information easy to find? (%)

Answer:

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

How to improve your score

You can improve this score by putting all of your Access Control information under one overarching section of your documentation. It needs to be as easy to find and as easy to identify as possible. Additional acceptable identifiers of Access Control documentation include: “Governance”, “DAO” (if applicable), or “Admin Access/Powers”.

19) Are relevant contracts clearly labelled as upgradeable or immutable? (%)

✓ **Answer:**

Guidance:

100%: Both the contract documentation and the smart contract code state that the code is not upgradeable or immutable.

80%: All Contracts are clearly labelled as upgradeable (or not)

50%: Code is immutable but not mentioned anywhere in the documentation

0%: Admin control information could not be found

How to improve your score

You can improve your score by clearly labelling the upgradeability capabilities of each of your deployed smart contracts. Doing this is important from a user perspective. However, in order to get full points here you need to make all of your contracts immutable. Examples can be found here.

20) Is the type of smart contract ownership clearly indicated? (%)

✓ **Answer:**

Guidance:

100% The type of ownership is clearly indicated in their documentation. (OnlyOwner / MultiSig / etc)

50% The type of ownership is indicated, but only in the code. (OnlyOwner / MultiSig / etc)

0% Admin Control information could not be found

How to improve your score

You can improve your score through clearly outlining the ownership functions (OnlyOwner, MultiSig, Defined Roles) attributed to your deployed smart contract. Ideally, this would all be clearly formatted within your Access Controls section. At the bare minimum, these functions should be established and explained in your protocol’s code or code comments. If you do not have any ownership functions within your software’s code, do consider adding them in the next deployment of your smart contracts.

21) Are the protocol's smart contract change capabilities described? (%)

✓ **Answer:**

Guidance:

100% - The documentation covers the capabilities for change for all smart contracts

50% - The documentation covers the capabilities for change in some, but not all contracts

0% - The documentation does not cover the capabilities for change in any contract

How to improve your score:

You can improve your score for this question by adding all the necessary information regarding the capabilities of your smart contracts' upgradeability in your documentation. An example of this can be found here.

22) Is the protocol's admin control information easy to understand? (%)

✓ **Answer:**

Guidance:

100% - All the contracts are immutable

90% - Description relates to investments safety in clear 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.

23) Is there sufficient Pause Control documentation? (%)

✓ **Answer:**

Guidance:

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

To improve your score, you should either implement or expand upon the information available on your Pause Control function. Additionally, make sure to perform consistent “fire drill” tests on your emergency pause so that we can be assured that it has been battle-tested. You should also create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

24) Is there sufficient Timelock documentation? (%)

✓ **Answer:**

Guidance:

100% - Documentation identifies and explains why the protocol does not need a Timelock OR Timelock documentation identifies its duration, which contracts it applies to and justifies this time period.

60% - A Timelock is identified and its duration is specified

30% - A Timelock is identified

0% - No Timelock information was documented

How to improve this score

Create a document that identifies the timelock, its length and which contracts it is applicable to. This should be included in the protocol documentation. An example of good timelock documentation is Uniswap’s. Protocols may seek to use this as a starting point for their own Timelock documentation.

25) Is the Timelock of an adequate length? (Y/N)

✓ **Answer:**

Guidance:

100% - Timelock is between 48 hours to 1 week OR justification as to why no Timelock is needed / is outside this length.

50% - Timelock is less than 48 hours or greater than 1 week.

0% - No Timelock information was documented OR no timelock length was identified.

How to improve this score

In your Timelock documentation, make sure to include the duration of it and an adequate description of why this specific timeframe was chosen. If you do not use Timelocks due to immutable contracts or not having a governance structure, make sure to clearly outline this in your docs and appropriately justify it.

Oracles

This section goes over the documentation that a protocol may or may not supply about their Oracle usage. Oracles are a fundamental part of DeFi as they are responsible for relaying tons of price data information to thousands of protocols using blockchain technology. Not only are they important for price feeds, but they are also an essential component of transaction verification and security.

The score for this section is derived from the individual scores of the following questions:

26) Is the Oracle sufficiently documented? (%)

27) Is front running mitigated by this protocol? (Y/N)

28) Can flashloan attacks be applied to the protocol, and if so, are those flashloan attack risks mitigated? (Y/N)

26) Is the protocol's Oracle sufficiently documented? (%)

> ✓ **Answer:**

Guidance:

100% - If it uses one, the Oracle is specified. The contracts dependent on the oracle are identified. Basic software functions are identified (if the protocol provides its own price feed data). Timeframe of price feeds are identified.

100% - The reason as to why the protocol does not use an Oracle is identified and explained.

75% - The Oracle documentation identifies both source and timeframe, but does not provide additional context regarding smart contracts.

50% - Only the Oracle source is identified.

0%: No oracle is named / no oracle information is documented.

How to improve this score

Include a section within protocol documentation explaining the oracle it employs. If no oracle is used, state this. Pancakeswap documentation identifies the oracles that it uses, and justifies why it chose them. Protocols might consider using their documentation as a starting point. On the other hand, a protocol may simply not need a data source and should this be explained full marks will be awarded.

27) Is front running mitigated by this protocol? (Y/N)

> ✓ **Answer:**

Guidance:

Yes - The protocol cannot be front run and there is an explanation as to why OR documented front running countermeasures are implemented.

No - The protocol does not mention front running or does not document any countermeasure against it

How to improve this score

Document the frontrunning countermeasures your protocol employs. If no frontrunning countermeasures are employed, state and justify this. For an example of documentation relating to frontrunning countermeasures, see mistX’s documentation. KeeperDAO also has good documentation on this matter.

28) Can flashloan attacks be applied to the protocol, and if so, are those flashloan attack risks mitigated? (Y/N)

> ✓ **Answer:**

Guidance:

Yes - The protocol's documentation includes information on how they mitigate the possibilities and extents of flash loan attacks.

No - The protocol's documentation does not include any information regarding the mitigation of flash loan attacks.

How to improve this score

Identify in your documentation what countermeasure the protocol employs to prevent a flashloan attack. An example of how this might be prevented can be found here and this should be documented in this way.

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

Smart Contract Address Appendix

Team Appendix

Code Used Appendix

Example Code Appendix

```

```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex

Solidity						

Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript						

Tests to Code / = %