0.7

Impermax Finance Process Quality Review

Score: 65%

Overview

This is a Process Quality Review of Impermax Finance completed on May 17th 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 65%, almost a pass. 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.

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: 20%

Executing code addresses were found in the #resources channel in their discord, as seen 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 63 transactions a day on contract Router.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/Impermax-Finance/impermax-x-uniswapv2-core

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: 10%

with 17 commits and 1 branch, this is not a healthy 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: Yes

Full team members information can be found at https://impermax.finance/

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://impermax.finance/Whitepaper-Impermax-UniswapV2.pdf

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

> ⚠ Answer: No

There is no evident 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: 0%

Code examples are in the Appendix. As per the SLOC, there is 17% 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%

Documentation lists and describes all functions with a bit of code traceability for some of them.

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: 100%

With a TtC ratio of 265%, this is clearly a well-tested protocol.

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

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

> ✓ Answer: 100%

There is evident code coverage available at https://github.com/Impermax-Finance/impermax-x-uniswapv2-core/blob/main/audit/SC_impermax_core.pdf

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

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

> ✓ Answer: Yes

Instructions available at https://github.com/Impermax-Finance/impermax-x-uniswapv2-core

14) Report of the results (%)

> ⚠ Answer: 0%

No test report results visible in their documentation.

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 Impermax Finance formal verification on the web or documentation.

16) Stress Testing environment (%)

> ⚠ Answer: 0%

No testing smart contract addresses found in their documentation.

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 an audit on them on April 27th 2021.

CyberUnit did an audit on them January 18th 2020 that was updated February 8th 2021.

Impermax Finance was officially launched April 29th 2021.

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%

There is no evident bug bounty program.

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: 100%

The whitepaper indicates that their smart contracts are immutable.

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%

Their whitepaper indicates that all contracts are immutable.

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%

Their whitepaper describes that their contracts are immutable.

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: 100%

Code is indicated as immutable (therefore no pause control).

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.

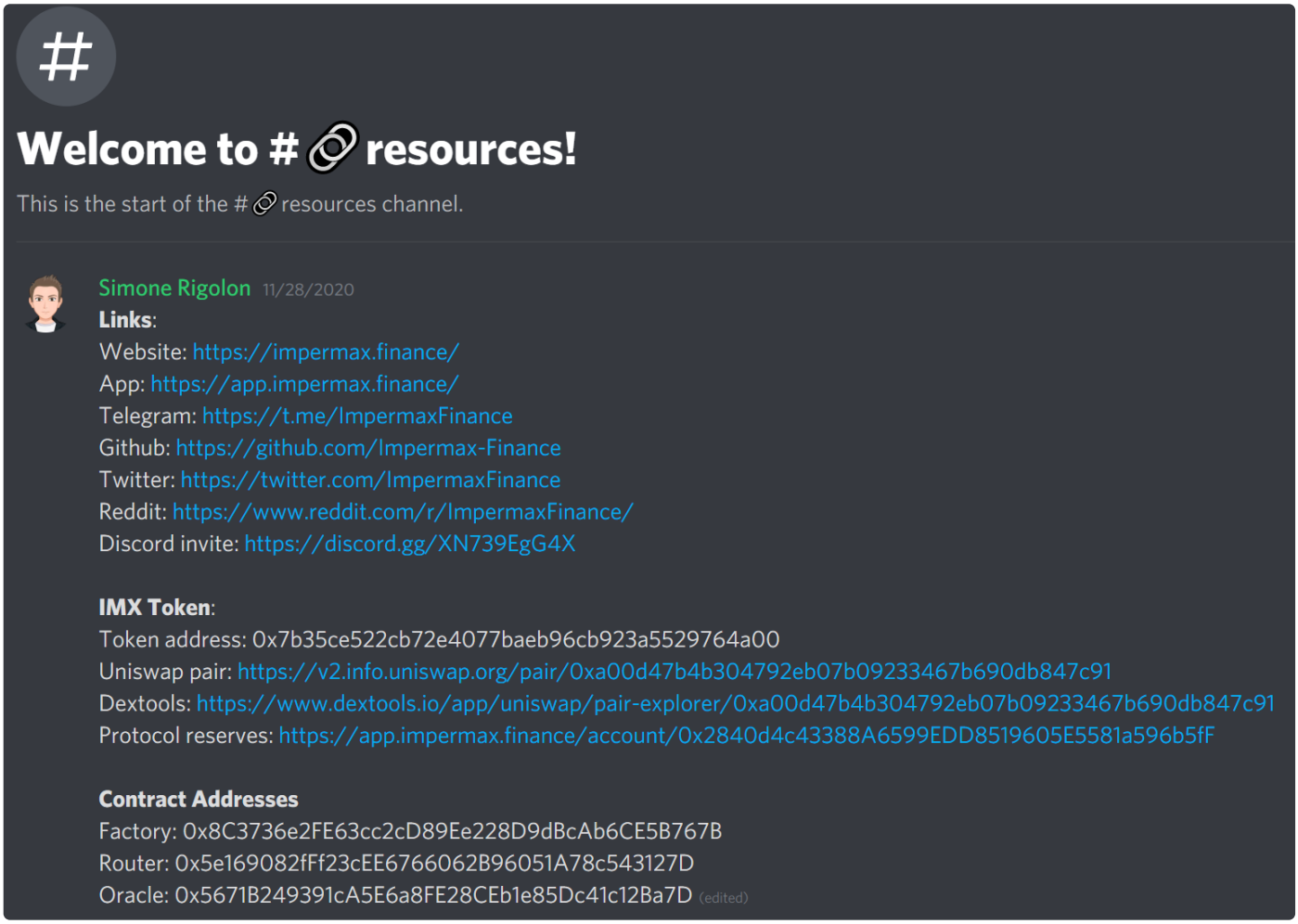
DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix

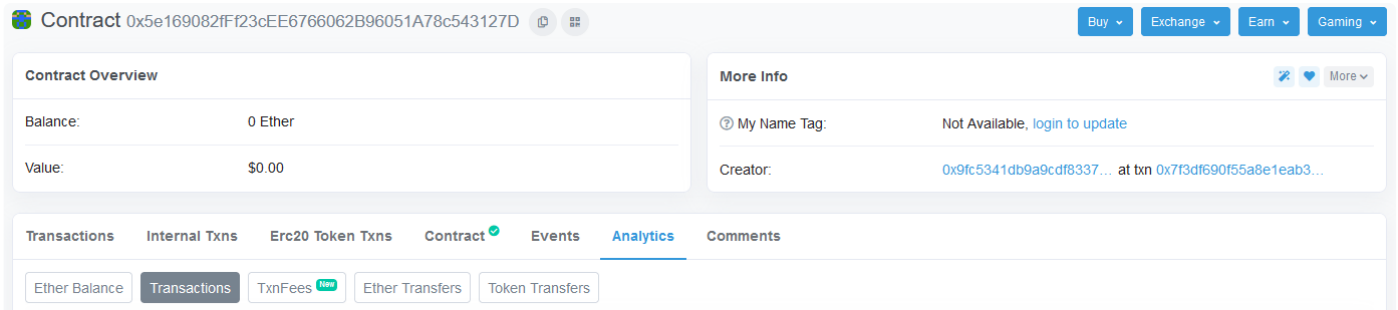
	Total	Impermax	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		169.5
Code and Team			**65%**
1) Are the executing code addresses readily available? (%)	**20**	20%	4
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	y	5
4) Is there a development history visible? (%)	**5**	10%	0.5
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**	0%	0
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**	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	100%	5
13) Scripts and instructions to run the tests? (Y/N)	**5**	Y	5
14) Report of the results (%)	**10**	0%	0
15) Formal Verification test done (%)	**5**	0%	0
16) Stress Testing environment (%)	**5**	0%	0
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	0%	0

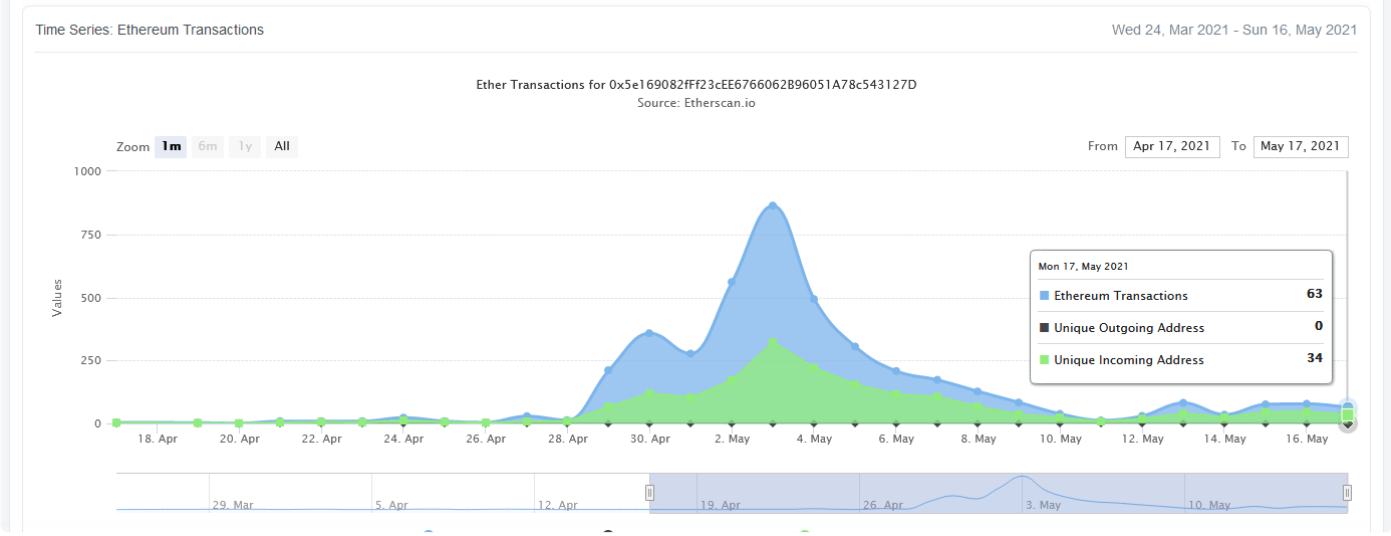
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	100%	5
20) Is the information clear and complete	**10**	100%	10
21) Is the information in non-technical terms	**10**	100%	10
22) Is there Pause Control documentation including records of tests	**10**	100%	10
Section Scoring			
Code and Team	50	59%	
Documentation	45	11%	
Testing	50	60%	
Security	80	88%	
Access Controls	35	100%	

Executing Code Appendix



Code Used Appendix





Example Code Appendix

```
pragma solidity =0.5.16;

import "./BStorage.sol";
import "./PoolToken.sol";

contract BInterestRateModel is PoolToken, BStorage {

        // When utilization is 100% borrowRate is kinkBorrowRate * KINK_MULTIPLIER
        // kinkBorrowRate relative adjustment per second belongs to [1-adjustSpeed, 1+adjus
        uint public constant KINK_MULTIPLIER = 5;
        uint public constant KINK_BORROW_RATE_MAX = 31.7097920e9; //100% per year
        uint public constant KINK_BORROW_RATE_MIN = 0.31709792e9; //1% per year

        event AccrueInterest(uint interestAccumulated, uint borrowIndex, uint totalBorrows)
        event CalculateKinkBorrowRate(uint kinkBorrowRate);
        event CalculateBorrowRate(uint borrowRate);

        function _calculateBorrowRate() internal {
                uint _kinkUtilizationRate = kinkUtilizationRate;
                uint _adjustSpeed = adjustSpeed;
                uint _borrowRate = borrowRate;
                uint _kinkBorrowRate = kinkBorrowRate;
                uint32 _rateUpdateTimestamp = rateUpdateTimestamp;

                // update kinkBorrowRate using previous borrowRate
                uint32 timeElapsed = getBlockTimestamp() - _rateUpdateTimestamp; // underf
                if(timeElapsed > 0) {
                        rateUpdateTimestamp = getBlockTimestamp();
                        uint adjustFactor;

                        if (_borrowRate < _kinkBorrowRate) {
                                // never overflows, _kinkBorrowRate is never 0
                                uint tmp = (_kinkBorrowRate - _borrowRate) * 1e18 / _kinkB
                                adjustFactor = tmp > 1e18 ? 0 : 1e18 - tmp;
                        } else {
```

```
                            // never overflows, _kinkBorrowRate is never 0
                            uint tmp = (_borrowRate - _kinkBorrowRate) * 1e18 / _kinkB
                            adjustFactor = tmp + 1e18;
                        }

                        // never overflows
                        _kinkBorrowRate = _kinkBorrowRate * adjustFactor / 1e18;
                        if(_kinkBorrowRate > KINK_BORROW_RATE_MAX) _kinkBorrowRate = KINK_
                        if(_kinkBorrowRate < KINK_BORROW_RATE_MIN) _kinkBorrowRate = KINK_

                        kinkBorrowRate = uint48(_kinkBorrowRate);
                        emit CalculateKinkBorrowRate(_kinkBorrowRate);
                }

                uint _utilizationRate;
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	28	1615	274	194	1147	112

Comments to Code 194/1147 = 17%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	28	3566	421	105	3040	125

Tests to Code 3040/1147 = 265%