0.7

Rabbit Finance Process Quality Review

Score: 42%

Overview

This is a Rabbit Finance Process Quality Review completed on July 12th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is 42%, 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.

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain: Binance Smart Chain**

Guidance:

Ethereum
Binance Smart Chain
Polygon

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://rabbitfinance.gitbook.io/homepage/resources/contract-information, 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 3500 transactions a day on contract *Boardroom.sol*, 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

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

> ✓ Answer: Yes

GitHub: https://github.com/RabbitFinanceProtocol

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: 0%

With 11 commits and 1 branch, this is an unhealthy software 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 30 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

No public team info was found.

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://rabbitfinance.gitbook.io/homepage/.

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

> ⚠ Answer: No

Although there are many financial and dApp parameters, the basic software functions (code) are not documented in their publicly-available 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%

Rabbit Finance's software functions (code) are not documented in their publicly-available 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: 40%

Code examples are in the Appendix. As per the SLOC, there is 102% commenting to code (CtC), however most were include files. Adapted to 40% based on boardroom.sol.

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

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

⚠ Answer: 0%

As there are no software functions (code) documented in their publicly-available documentation, it is impossible to trace them to their implementation in source code.

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%

As there is no test suite in Rabbit Finance's public GitHub, we cannot identify which files are test files. Therefore, we cannot properly evaluate the Test to Code ratio (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 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 code coverage report was found in Rabbit Finance's public GitHub or in any of their public audit reports by Certik and Chains Guard.

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

As there is no testing suite in Rabbit Finance's GitHub, there are no scripts and instructions to run them either.

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%

No test report was provided by Rabbit Finance in their public GitHub repository.

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 Rabbit Finance Formal Verification was found in any of their public documentation or in web searches.

16) Stress Testing environment (%)

⚠ Answer: 0%

No evidence of any Rabbit Finance test-net smart contract usage was found in any of their publicly-available 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: 90%

Certik has published a Rabbit Finance audit report on May 23rd 2021.

Chains Guard has published a Rabbit Finance audit report on June 2nd 2021.

Rabbit Finance was launched on May 24th, making it one audit report published before launch (Certik), and

one audit report published after launch (Chains Guard).

Note: Most of the fix recommendations provided by Certik and Chains Guard were successfully implemented by the Rabbit Finance team.

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

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

Deduct 25% if code is in a private repo and no note from auditors that audit is applicable to deployed code

18) Is the bounty value acceptably high (%)

> ⓘ Answer: 40%

Rabbit Finance's Bug Bounty program offers rewards up to 50,000$ for the most critical of issues found.

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

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 access controls (%)

> ⚠ Answer: 20%

The only semblance of admin access control information available in Rabbit Finance's documentation is about the TimeLock function. However, since this isn't really information about what the admins can and cannot access and/or change within the contract, they effectively have no access control documentation, other than some mentions that their contracts are upgradeable.

Note: The upcoming launch of their governance system will likely hold more information regarding access controls, and will likely change the score of this section of our review.

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: 30%

a) The different contracts are clearly labelled as upgradeable throughout Rabbit Finance's documentation.

b) ~~No ownership type is identified.~~

c) ~~No capabilities for contract changes are described.~~

Guidance:

All the contracts are immutable -- 100% OR

a) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
c) 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%

As no clear listings of Rabbit Finance's admin access control exist in their publicly-available documentation, we cannot evaluate how technical the information is.

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%

There is no evidence of Pause Control, Pause Control tests, or a similar function in any of Rabbit Finance's publicly-available documentation or GitHub.

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	Rabbit Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		108
Code and Team			**42%**
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**	0%	0
5) Is the team public (not anonymous)? (Y/N)	**15**	N	0
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**	40%	2
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**	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
Security			
17) Did 3rd Party audits take place? (%)	**70**	90%	63
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**	20%	1
20) Is the information clear and complete	**10**	30%	3
21) Is the information in non-technical terms	**10**	0%	0

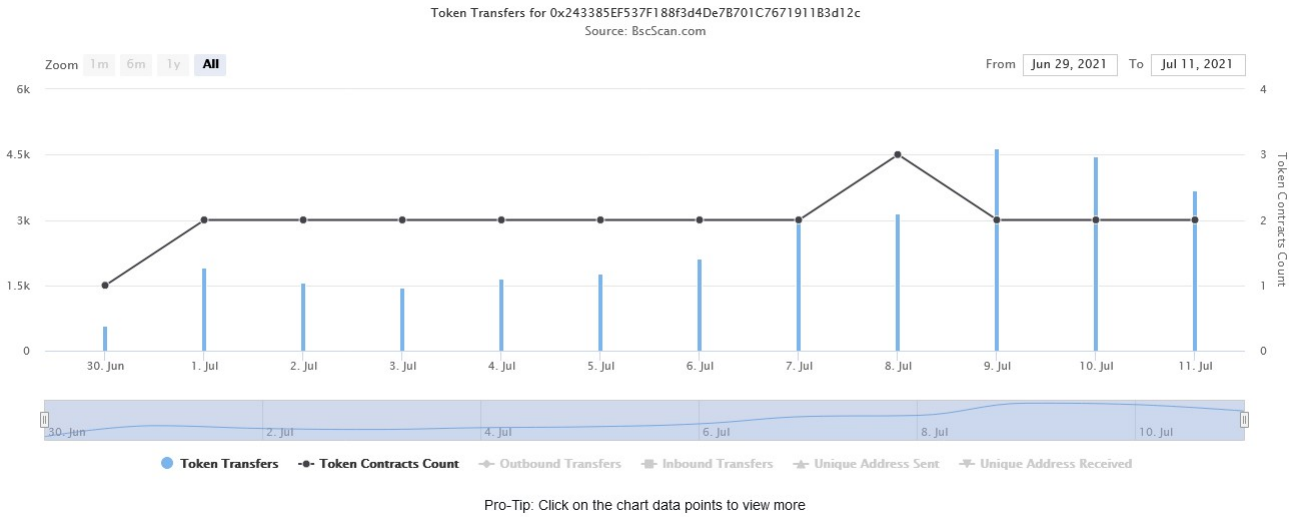
22) Is there Pause Control documentation including records of tests	**10**	0%	0
Section Scoring			
Code and Team	50	60%	
Documentation	45	16%	
Testing	50	0%	
Security	80	84%	
Access Controls	35	11%	

Executing Code Appendix

Important Contract

- ProxyAdmin Contract: 0x62C6a4396E306A5628BFDE974F33905954D48068
- FairLaunch Contract: 0x81C1e8A6f8eB226aA7458744c5e12Fc338746571
- Config: 0x371Da7799122F1105e928be2Dd36Cd1b89d60B32
- RateModel: 0x122B102aC3555A66C667E9E3E497c7683fF8FdC4
- CARROT Boardroom X3: 0x243385EF537F188f3d4De7B701C7671911B3d12c
- CARROT Boardroom X7: 0x178A256886DB96070a71D189Af365514f62b5ff6
- CARROT Treasury: 0x5555382b4f10f34c402C055C06CeD6137C65D421
- CARROT seigniorageOracle: 0xaf942551BB38cbd1963A4160E6D720Ac7E0e3aC2

Code Used Appendix



Example Code Appendix

```
1 contract Boardroom is ShareWrapper, ContractGuard, Operator {
2     using SafeERC20 for IERC20;
3     using Address for address;
4     using SafeMath for uint256;
```

```
    using Safe112 for uint112;

    /* ========== DATA STRUCTURES ========== */

    struct Boardseat {
        uint256 lastSnapshotIndex;
    }

    struct BoardSnapshot {
        uint256 time;
        uint256 rewardReceived;
        uint256 rewardPerShare;
    }

    /* ========== STATE VARIABLES ========== */

    IERC20 public cash;
    uint256 public timeLock;

    mapping(address => Boardseat) private directors;
    BoardSnapshot[] private boardHistory;

    /* ========== CONSTRUCTOR ========== */

    constructor(IERC20 _cash, IERC20 _share) public {
        cash = _cash;
        share = _share;

        BoardSnapshot memory genesisSnapshot = BoardSnapshot({
            time: block.number,
            rewardReceived: 0,
            rewardPerShare: 0
        });
        boardHistory.push(genesisSnapshot);
    }

    /* ========== Modifiers =============== */
    modifier directorExists {
        require(
            balanceOf(msg.sender) > 0,
            'Boardroom: The director does not exist'
        );
        _;
    }

    /* ========== VIEW FUNCTIONS ========== */

    // ========== Snapshot getters

    function latestSnapshotIndex() public view returns (uint256) {
        return boardHistory.length.sub(1);
    }

```

```
    function getLatestSnapshot() internal view returns (BoardSnapshot memory) {
        return boardHistory[latestSnapshotIndex()];
    }

    function getLastSnapshotIndexOf(address director)
        public
        view
        returns (uint256)
    {
        return directors[director].lastSnapshotIndex;
    }

    function getLastSnapshotOf(address director)
        internal
        view
        returns (BoardSnapshot memory)
    {
        return boardHistory[getLastSnapshotIndexOf(director)];
    }

    // ========== Director getters

    function rewardPerShare() public view returns (uint256) {
        return getLatestSnapshot().rewardPerShare;
    }

    function earned(address director) public view returns (uint256) {
        uint256 latestRPS = getLatestSnapshot().rewardPerShare;
        uint256 storedRPS = getLastSnapshotOf(director).rewardPerShare;

        return
            balanceOf(director).mul(latestRPS.sub(storedRPS)).div(1e18);
    }

    /* ========== MUTATIVE FUNCTIONS ========== */

    function stake(uint256 amount)
        public
        override
        onlyOneBlock
    {
        require(amount > 0, 'Boardroom: Cannot stake 0');
        claimReward();
        super.stake(amount);
        emit Staked(msg.sender, amount);
    }

    function withdraw(uint256 amount)
        public
        override
        onlyOneBlock
        directorExists
    {
```

```
    {
111
        require(amount > 0, 'Boardroom: Cannot withdraw 0');
112     require(block.timestamp >= timeLock,"Boardroom: Withdraw Time Lockd");
113     claimReward();
114     super.withdraw(amount);
115     emit Withdrawn(msg.sender, amount);
116 }
117
118 function claimReward() public {
119     uint256 reward = earned(msg.sender);
120     if (reward > 0) {
121         cash.safeTransfer(msg.sender, reward);
122         emit RewardPaid(msg.sender, reward);
123     }
124     directors[msg.sender].lastSnapshotIndex = latestSnapshotIndex();
125 }
126
127 function allocateSeigniorage(uint256 amount)
128     external
129     onlyOneBlock
130     onlyOperator
131 {
132     require(amount > 0, 'Boardroom: Cannot allocate 0');
133     require(
134         totalSupply() > 0,
135         'Boardroom: Cannot allocate when totalSupply is 0'
136     );
137
138     // Create & add new snapshot
139     uint256 prevRPS = getLatestSnapshot().rewardPerShare;
140     uint256 nextRPS = prevRPS.add(amount.mul(1e18).div(totalSupply()));
141
142     BoardSnapshot memory newSnapshot = BoardSnapshot({
143         time: block.number,
144         rewardReceived: amount,
145         rewardPerShare: nextRPS
146     });
147     boardHistory.push(newSnapshot);
148
149     cash.safeTransferFrom(msg.sender, address(this), amount);
150     emit RewardAdded(msg.sender, amount);
151 }
152
153 function burnReward()
154     external
155     onlyOneBlock
156     onlyOperator
157 {
158     for(uint256 i = 0;i<boardHistory.length;i++){
159         boardHistory[i].rewardPerShare = 0;
160     }
161     IERC20 _cash = IERC20(cash);
162     uint256 boardBalance = _cash.balanceOf(address(this));
```

```
163         if (boardBalance > 0){
164 
                // transfer treasury
165             _cash.transfer(operator(),boardBalance);
166         }
167     }
168 
169     function setTimeLock(uint256 time) external onlyOperator {
170         timeLock = time;
171     }
172 
173     /* ========== EVENTS ========== */
174     event Staked(address indexed user, uint256 amount);
175     event Withdrawn(address indexed user, uint256 amount);
176     event RewardPaid(address indexed user, uint256 reward);
177     event RewardAdded(address indexed user, uint256 reward);
178 }
179 
180 
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	18	12819	1719	5614	5486	556

Comments to Code 5614/5486 = 102%

Javascript Tests

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

Tests to Code 0/0 = 0%