0.7

Cyclone Protocol Process Quality Review

Score: 53%

Overview

This is a Cyclone Protocol Process Quality Review completed on July 26th 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 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:** Ethereum, Binance Smart Chain, Polygon

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche

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 following 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://docs.cyclone.xyz/deployment, 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 labeling not clear or easy to find
0% Executing addresses could not be found

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

> **Answer:** 100%

Activity is over 10 transactions a day on contract *AeolusV2dot1.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/cycloneprotocol/cyclone-contracts.

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"**.

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

> **Answer:** 0%

With 16 commits and 1 branch, this is an unhealthy software repository.

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:

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

The Cyclone Protocol team is anonymous.

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, 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.cyclone.xyz/.

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

> ⚠ **Answer:** No

Cyclone Protocol has no actual software documentation or even detailed description of algorithms.

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

> ⚠ **Answer:** 0%

Cyclone Protocol does not seem to have software 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

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

> ⓘ **Answer:** 38%

Code examples are in the Appendix. As per the SLOC, there is 38% commenting to code (CtC).

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

Note: The CtC was calculated using the main operating smart contract source code files from the Cyclone Protocol GitHub repository. Files that were not used: interface files, Cream, mock files, Uniswap files, Tornado Cash files, and all previous version files of important contracts (ex: AeolusV2.sol and AeolusV2dot2.sol).

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%

With no documentation, there can not be traceability.

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 documentation to code such that it is clear where each outlined function is coded in the source code. For reference, check the SecurEth guidelines on traceability.

Testing

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

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

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

> **Answer:** 40%

Code examples are in the Appendix. As per the SLOC, there is 57% 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 improved 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:** 30%

There is no evidence of test coverage in any of the Cyclone Protocol documentation or audit reports. However, they do have an acceptable testing suite.

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 improved by adding tests that achieve 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:** Yes

Scrips/Instructions location: https://docs.cyclone.xyz/development#test-contracts.

14) Report of the results (%)

> ⚠ **Answer: 0%**

There is no evidence of any test reports in the Cyclone Protocol documentation or software repositories.

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%

There is no evidence of a Cyclone Protocol Formal Verification test in any of their documentation or in web searches.

16) Stress Testing environment (%)

> ✓ **Answer:** 100%

Although there are no provided test-net addresses, Cyclone Protocol's documentation instructs users on how to interact with their test-net smart contracts, which is evidence that there is presence of stress testing via test-net smart contracts. Found at https://docs.cyclone.xyz/development#test-contracts and https://docs.cyclone.xyz/development#interaction-with-cyclone-contracts-anonymity-mining.

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%

ChainShield published a Cyclone V2 audit report on March 12th 2021.

Slowmist published a Cyclone V2 audit report on March 22nd 2021.

Most fix recommendations were implemented by the Cyclone team.

Cyclone Protocol V2 was launched on March 15th 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

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:** 0%

There is no evidence of a Cyclone Protocol 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
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

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.

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 access controls (%)

> **Answer:** 100%

Location: https://docs.cyclone.xyz/governance.

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:** 60%

a) Contracts are clearly upgradeable via the voting process.

c) The capabilities for contract changes via voting mechanisms is described here.

Note: Cyclone's full governance model is technically not fully launched yet. However, they still have this information ready and available in their current governance documentation.

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: 3**0%

There currently is not any information regarding investment safety due to the fact that the full Cyclone governance model has yet to be launched.

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:** 40%

Evidence of a Pause Control mechanism can be found in *Pausable.sol*.

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

PQ Audit Scoring Matrix (v0.7)	Total Points	Cyclone Protocol Answer	Cyclone Protocol Points
Total	260		137.4
Code and Team			**53%**
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	38%	1.9
10) Is it possible to trace from software documentation to the implementation in code (%)	10	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	40%	8
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	30%	1.5
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	100%	5
Security			
17) Did 3rd Party audits take place? (%)	70	90%	63
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	60%	6
21) Is the information in non-technical terms	10	30%	3
22) Is there Pause Control documentation including records of tests	10	40%	4
Section Scoring			
Code and Team	50	60%	
Documentation	45	15%	
Testing	50	39%	
Security	80	79%	
Access Controls	35	51%	

Executing Code Appendix

Ethereum Mainnet

- Cyclone Token: `0x8861cff2366c1128fd699b68304ad99a0764ef9a`
- Aeolus LP Mining v2.1: `0xdc71bc29d12960a3ee5452fac6f033a1b8e756fb`
- Hasher: `0x949452e32db13a5771445cf20b304474b866202b`
- Verifier: `0x7C994FB3a8C208C1750Df937d473040c604292D6`
- UniswapV2 Router: `0x602b40bf327c10370483ae5ecde15a7bb480dcca`

Anonymity Pools

- Latte - 100 ETH - Pool: `0xd619c8da0a58b63be7fa69b4cc648916fe95fa1b`
- Expresso - 100000 USDT - Pool: `0xa38b6742cef9573f7f97c387278fa31482539c3d`
- Cold Brew - 100 TORN Pool: `0x09f03488291063a8f3c67d2aab7002419d11c113`

Code Used Appendix

Advanced Latest 25 internal transaction

View All

Parent Txn Hash	Block	Age	From	To	Value
0xdf187756527e2f4e9b...	12901756	2 hrs 11 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0xdf187756527e2f4e9b...	12901756	2 hrs 11 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0xdf187756527e2f4e9b...	12901756	2 hrs 11 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0xdf187756527e2f4e9b...	12901756	2 hrs 11 mins ago	0xdc71bc29d12960a3ee...	0x37d9c7f451e5c619a7...	0 Ether
0x0821212fe736d612b0...	12901705	2 hrs 24 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x0821212fe736d612b0...	12901705	2 hrs 24 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x0821212fe736d612b0...	12901705	2 hrs 24 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x0821212fe736d612b0...	12901705	2 hrs 24 mins ago	0xdc71bc29d12960a3ee...	0x37d9c7f451e5c619a7...	0 Ether
0x4153557d80a19f1431...	12900953	5 hrs 22 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x4153557d80a19f1431...	12900953	5 hrs 22 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x4153557d80a19f1431...	12900953	5 hrs 22 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x4153557d80a19f1431...	12900953	5 hrs 22 mins ago	0xdc71bc29d12960a3ee...	0x37d9c7f451e5c619a7...	0 Ether
0x992cae809a2a90b20a...	12900898	5 hrs 34 mins ago	0xdc71bc29d12960a3ee...	0x37d9c7f451e5c619a7...	0 Ether
0x992cae809a2a90b20a...	12900898	5 hrs 34 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether
0x992cae809a2a90b20a...	12900898	5 hrs 34 mins ago	0xdc71bc29d12960a3ee...	0x8861cff2366c1128fd6...	0 Ether

Example Code Appendix

```
// Aeolus is the master of Cyclone tokens. He can distribute CYC and he is a fair guy.
//
// Note that it's ownable and the owner wields tremendous power. The ownership
```

```
// will be transferred to a governance smart contract once CYC is sufficiently
// distributed and the community can show to govern itself.
//
// Have fun reading it. Hopefully it's bug-free. God bless.
contract AeolusV2dot2 is Ownable {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;     // How many LP tokens the user has provided.
        uint256 rewardDebt; // Reward debt. See explanation below.
        //
        // We do some fancy math here. Basically, any point in time, the amount of CYCs
        // entitled to a user but is pending to be distributed is:
        //
        //   pending reward = (user.amount * accCYCPerShare) - user.rewardDebt
        //
        // Whenever a user deposits or withdraws LP tokens to a pool. Here's what happens:
        //   1. Update accCYCPerShare and lastRewardBlock
        //   2. User receives the pending reward sent to his/her address.
        //   3. User's `amount` gets updated.
        //   4. User's `rewardDebt` gets updated.
    }


    // Address of LP token contract.
    IERC20 public lpToken;
    // Accumulated CYCs per share, times 1e12. See below.
    uint256 public accCYCPerShare;
    // Last block reward block height
    uint256 public lastRewardBlock;
    // Reward per block
    uint256 public rewardPerBlock;
    // Reward to distribute
    uint256 public rewardToDistribute;
    // Entrance Fee Rate
    uint256 public entranceFeeRate;

    IExchange public exchange;
    // The Cyclone TOKEN
    IMintableToken public cycToken;

    // Info of each user that stakes LP tokens.
    mapping (address => UserInfo) public userInfo;

    event RewardAdded(uint256 amount, bool isBlockReward);
    event Deposit(address indexed user, uint256 amount, uint256 fee);
    event Withdraw(address indexed user, uint256 amount);
    event EmergencyWithdraw(address indexed user, uint256 amount);

    constructor(IMintableToken _cycToken, address payable _exchange) public {
        cycToken = _cycToken;
```

```
        lastRewardBlock = block.number;
        lpToken = IERC20(_exchange);
        exchange = IExchange(_exchange);
    }

    function() external payable {
    }

    function setEntranceFeeRate(uint256 _entranceFeeRate) public onlyOwner {
        require(_entranceFeeRate < 10000, "invalid entrance fee rate");
        entranceFeeRate = _entranceFeeRate;
    }

    function setRewardPerBlock(uint256 _rewardPerBlock) public onlyOwner {
        updateBlockReward();
        rewardPerBlock = _rewardPerBlock;
    }

    function rewardPending() internal view returns (uint256) {
        uint256 reward = block.number.sub(lastRewardBlock).mul(rewardPerBlock);
        uint256 cycBalance = cycToken.balanceOf(address(this)).sub(rewardToDistribute);
        if (cycBalance < reward) {
            return cycBalance;
        }
        return reward;
    }

    // View function to see pending reward on frontend.
    function pendingReward(address _user) external view returns (uint256) {
        UserInfo storage user = userInfo[_user];
        uint256 acps = accCYCPerShare;
        if (rewardPerBlock > 0) {
            uint256 lpSupply = lpToken.balanceOf(address(this));
            if (block.number > lastRewardBlock && lpSupply > 0) {
                acps = acps.add(rewardPending().mul(1e12).div(lpSupply));
            }
        }

        return user.amount.mul(acps).div(1e12).sub(user.rewardDebt);
    }

    // Update reward variables to be up-to-date.
    function updateBlockReward() public {
        if (block.number <= lastRewardBlock || rewardPerBlock == 0) {
            return;
        }
        uint256 lpSupply = lpToken.balanceOf(address(this));
        uint256 reward = rewardPending();
        if (lpSupply == 0 || reward == 0) {
            lastRewardBlock = block.number;
            return;
        }
        rewardToDistribute = rewardToDistribute.add(reward);
```

```
110
        emit RewardAdded(reward, true);
111         lastRewardBlock = block.number;
112         accCYCPerShare = accCYCPerShare.add(reward.mul(1e12).div(lpSupply));
113     }
114
115     // Deposit LP tokens to Aeolus for CYC allocation.
116     function deposit(uint256 _amount) public {
117         updateBlockReward();
118         UserInfo storage user = userInfo[msg.sender];
119         uint256 originAmount = user.amount;
120         uint256 acps = accCYCPerShare;
121         if (originAmount > 0) {
122             uint256 pending = originAmount.mul(acps).div(1e12).sub(user.rewardDebt);
123             if (pending > 0) {
124                 safeCYCTransfer(msg.sender, pending);
125             }
126         }
127         uint256 feeInCYC = 0;
128         if (_amount > 0) {
129             lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);
130             uint256 entranceFee = _amount.mul(entranceFeeRate).div(10000);
131             if (entranceFee > 0) {
132                 (uint256 iotxAmount, uint256 cycAmount) = exchange.removeLiquidity(entrance
133                 feeInCYC = cycAmount.add(exchange.iotxToTokenSwapInput.value(iotxAmount)(1
134                 require(cycToken.burn(feeInCYC), "failed to burn cyc token");
135                 _amount = _amount.sub(entranceFee);
136             }
137             user.amount = originAmount.add(_amount);
138         }
139         user.rewardDebt = user.amount.mul(acps).div(1e12);
140         emit Deposit(msg.sender, _amount, feeInCYC);
141     }
142
143     // Withdraw LP tokens from Aeolus.
144     function withdraw(uint256 _amount) public {
145         UserInfo storage user = userInfo[msg.sender];
146         uint256 originAmount = user.amount;
147         require(originAmount >= _amount, "withdraw: not good");
148         updateBlockReward();
149         uint256 acps = accCYCPerShare;
150         uint256 pending = originAmount.mul(acps).div(1e12).sub(user.rewardDebt);
151         if (pending > 0) {
152             safeCYCTransfer(msg.sender, pending);
153         }
154         if (_amount > 0) {
155             user.amount = originAmount.sub(_amount);
156             lpToken.safeTransfer(address(msg.sender), _amount);
157         }
158         user.rewardDebt = user.amount.mul(acps).div(1e12);
159         emit Withdraw(msg.sender, _amount);
160     }
161
```

```
162     // Withdraw without caring about rewards. EMERGENCY ONLY.
163
        function emergencyWithdraw() public {
164         UserInfo storage user = userInfo[msg.sender];
165         uint256 amount = user.amount;
166         user.amount = 0;
167         user.rewardDebt = 0;
168         lpToken.safeTransfer(address(msg.sender), amount);
169         emit EmergencyWithdraw(msg.sender, amount);
170     }
171
172     // Safe CYC transfer function, just in case if rounding error causes pool to not have e
173     function safeCYCTransfer(address _to, uint256 _amount) internal {
174         IMintableToken token = cycToken;
175         uint256 cycBalance = token.balanceOf(address(this));
176         if (_amount > cycBalance) {
177             _amount = cycBalance;
178         }
179         rewardToDistribute = rewardToDistribute.sub(_amount);
180         require(token.transfer(_to, _amount), "failed to transfer cyc token");
181     }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	14	2568	303	625	1640	209

Comments to Code 625/1640 = 38%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	5	1062	85	39	938	1

Tests to Code 938/1640 = 57%