

0.7

PrivacySwap Process Quality Review

Score: 41%

Overview

This is a [PrivacySwap](#) Process Quality Review completed on July 5th 2021. It was performed using the Process Review process (version 0.7.2) and is documented [here](#). The review was performed by Nic of DeFiSafety. Check out our [Telegram](#).

The final score of the review is 41%, a fail. The breakdown of the scoring is in [Scoring Appendix](#). For our purposes, a pass is 70%.

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

Disclaimer

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

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

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

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

Chain

This section indicates the blockchain used by this protocol.

✓ **Chain: Binance Smart Chain**

Guidance:

Ethereum

Binance 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://docs.privacyswap.finance/tokenomics/smart-contracts> 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 1800 transactions a day on contract *MasterChef.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/PrivacySwap>

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 4 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 PrivacySwap staff information 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://docs.privacyswap.finance/>

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

 Answer: No

No software functions are documented in the PrivacySwap 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%

No software functions are documented in the PrivacySwap 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: 59%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 59% 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%

No software functions are documented in the PrivacySwap documentation.

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%

There is no testing suite in the PrivacySwap GitHub.

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 evidence of code coverage was found in any of the PrivacySwap audits or GitHub repositories.

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

There is no testing suite in the PrivacySwap GitHub repositories.

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%

There is no test report in any of PrivacySwap's GitHub 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%

No evidence of a PrivacySwap Formal Verification test was found in their documentation or in web searches.

16) Stress Testing environment (%)

 Answer: 0%

No evidence of test-net smart contract usage was found in any PrivacySwap documentation or repository.


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: 70%

[TechRate published a PrivacySwap audit report in May 2021.](#)

Note 1: PrivacySwap was launched in March 2021.

Note 2: No changes required.

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

18) Is the bounty value acceptably high (%)

 Answer: 0%

No PrivacySwap Bug Bounty program was 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?
- 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: 70%


Robust access control documentation is found in the "[Security](#)" section of their documentation.

Note: Gave them a 70% because some access control information is found in the "Features" section, and takes a bit of looking to find. These include voting functions.

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) Initial liquidity locked for 6 months, after which it will become upgradeable, and this is clearly stated [here](#).
- c) Capabilities for change are described [here](#) and [here](#).

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: 90%

Information pertaining access controls is explained in very user-friendly language that is easy to decipher.

Guidance:

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

No evidence of Pause Control or similar functions in any PrivacySwap documentation or repository.

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](https://twitter.com/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](https://secur.eth.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

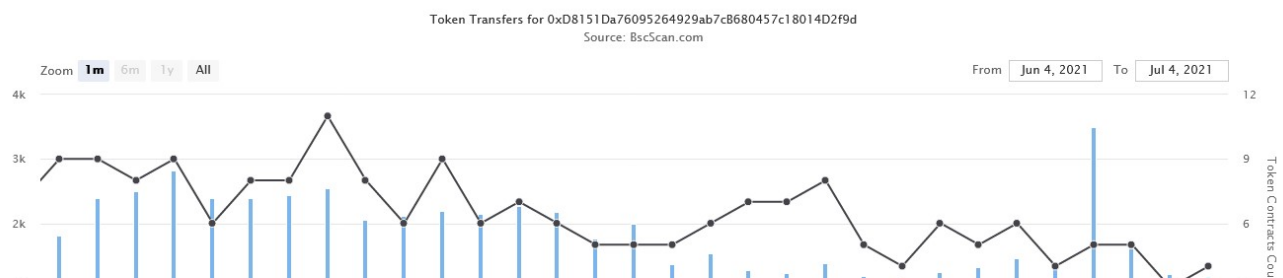
PQ Audit Scoring Matrix (v0.7)	Total	PrivacySwap	
	Points	Answer	Points
Total	260		105.45
Code and Team			41%
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	100%	5

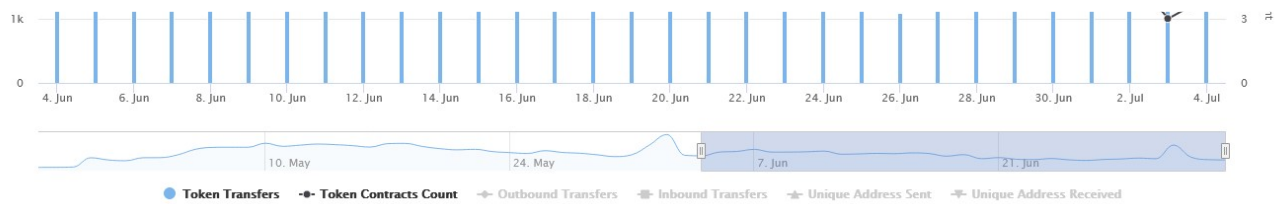
3) Is there a public software repository? (Y/N)	5	Y	5
4) Is there a development history visible? (%)	5	0%	0
5) Is the team public (not anonymous)? (Y/N)	15	N	0
Code Documentation			
6) Is there a whitepaper? (Y/N)	5	Y	5
7) Are the basic software functions documented? (Y/N)	10	N	0
8) Does the software 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	59%	2.95
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	70%	49
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	70%	3.5
20) Is the information clear and complete	10	60%	6
21) Is the information in non-technical terms	10	90%	9
22) Is there Pause Control documentation including records of tests	10	0%	0
Section Scoring			
Code and Team	50	60%	
Documentation	45	18%	
Testing	50	0%	
Security	80	61%	
Access Controls	35	53%	

Executing Code Appendix

- PRV Token Smart Contract: **0x7762a14082ab475c06d3868b385e46ae27017231**
- MasterChef: **0xD8151Da76095264929ab7cB680457c18014D2f9d**
- Deployer: **0x4067e1b64b1B31F7424bF60DaC671264244a76Bc**
- LP Timelock (6 months): **0x1bE4554bDec0A2Ca45737604abF9c04E9E70b8CF**
- MasterChef Timelock (24 hours): **0xa78515e2eD5ee06738DD6d014c29a75aF13Cb36**

Code Used Appendix





Example Code Appendix

```

1 pragma solidity ^0.6.12;
2
3 import "@openzeppelin/contracts/math/SafeMath.sol";
4 import "../libs/IBEP20.sol";
5 import "../libs/SafeBEP20.sol";
6 import "@openzeppelin/contracts/access/Ownable.sol";
7
8 import "../PrivacySwap.sol";
9
10 // MasterChef is the master of PRV. He can make PRV and he is a fair guy.
11 //
12 // Note that it's ownable and the owner wields tremendous power. The ownership
13 // will be transferred to a governance smart contract once PRV is sufficiently
14 // distributed and the community can show to govern itself.
15 //
16 // Have fun reading it. Hopefully it's bug-free. God bless.
17 contract MasterChef is Ownable {
18     using SafeMath for uint256;
19     using SafeBEP20 for IBEP20;
20
21     // Info of each user.
22     struct UserInfo {
23         uint256 amount;          // How many LP tokens the user has provided.
24         uint256 rewardDebt;      // Reward debt. See explanation below.
25         //
26         // We do some fancy math here. Basically, any point in time, the amount of PRVs
27         // entitled to a user but is pending to be distributed is:
28         //
29         // pending reward = (user.amount * pool.accPRVPerShare) - user.rewardDebt
30         //
31         // Whenever a user deposits or withdraws LP tokens to a pool. Here's what happens:
32         // 1. The pool's `accPRVPerShare` (and `lastRewardBlock`) gets updated.
33         // 2. User receives the pending reward sent to his/her address.
34         // 3. User's `amount` gets updated.
35         // 4. User's `rewardDebt` gets updated.
36     }
37
38     // Info of each pool.
39     struct PoolInfo {
40         IBEP20 lpToken;          // Address of LP token contract.
41         uint256 allocPoint;      // How many allocation points assigned to this pool. PRV:
42         uint256 lastRewardBlock; // Last block number that PRVs distribution occurs.
43         uint256 accPRVPerShare;  // Accumulated PRVs per share, times 1e12. See below.

```

```

44     uint16 depositFeeBP;        // Deposit fee in basis points
45 }
46
47 // The PRV TOKEN!
48 PrivacySwap public prv;
49 // Dev address.
50 address public devaddr;
51 // PRV tokens created per block.
52 uint256 public prvPerBlock;
53 // Bonus multiplier for early prv makers.
54 uint256 public constant BONUS_MULTIPLIER = 1;
55 // Deposit Fee address
56 address public feeAddress;
57
58 // Info of each pool.
59 PoolInfo[] public poolInfo;
60 // Info of each user that stakes LP tokens.
61 mapping (uint256 => mapping (address => UserInfo)) public userInfo;
62 // Total allocation points. Must be the sum of all allocation points in all pools.
63 uint256 public totalAllocPoint = 0;
64 // The block number when PRV mining starts.
65 uint256 public startBlock;
66
67 event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
68 event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
69 event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);
70
71 constructor(
72     PrivacySwap _prv,
73     address _devaddr,
74     address _feeAddress,
75     uint256 _prvPerBlock,
76     uint256 _startBlock
77 ) public {
78     prv = _prv;
79     devaddr = _devaddr;
80     feeAddress = _feeAddress;
81     prvPerBlock = _prvPerBlock;
82     startBlock = _startBlock;
83 }
84
85 function poolLength() external view returns (uint256) {
86     return poolInfo.length;
87 }
88
89 // Add a new lp to the pool. Can only be called by the owner.
90 // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
91 function add(uint256 _allocPoint, IBEP20 _lpToken, uint16 _depositFeeBP, bool _withUpdate)
92     public {
93     require(_depositFeeBP <= 10000, "add: invalid deposit fee basis points");
94     if (_withUpdate) {
95         massUpdatePools();
96     }
97     uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;

```

```

97     totalAllocPoint = totalAllocPoint.add(_allocPoint);
98     poolInfo.push(PoolInfo({
99         lpToken: _lpToken,
100         allocPoint: _allocPoint,
101         lastRewardBlock: lastRewardBlock,
102         accPRVPerShare: 0,
103         depositFeeBP: _depositFeeBP
104     }));
105 }
106
107 // Update the given pool's PRV allocation point and deposit fee. Can only be called by
108 function set(uint256 _pid, uint256 _allocPoint, uint16 _depositFeeBP, bool _withUpdate)
109     require(_depositFeeBP <= 10000, "set: invalid deposit fee basis points");
110     if (_withUpdate) {
111         massUpdatePools();
112     }
113     totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
114     poolInfo[_pid].allocPoint = _allocPoint;
115     poolInfo[_pid].depositFeeBP = _depositFeeBP;
116 }
117
118 // Return reward multiplier over the given _from to _to block.
119 function getMultiplier(uint256 _from, uint256 _to) public view returns (uint256) {
120     return _to.sub(_from).mul(BONUS_MULTIPLIER);
121 }
122
123 // View function to see pending PRVs on frontend.
124 function pendingPRV(uint256 _pid, address _user) external view returns (uint256) {
125     PoolInfo storage pool = poolInfo[_pid];
126     UserInfo storage user = userInfo[_pid][_user];
127     uint256 accPRVPerShare = pool.accPRVPerShare;
128     uint256 lpSupply = pool.lpToken.balanceOf(address(this));
129     if (block.number > pool.lastRewardBlock && lpSupply != 0) {
130         uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
131         uint256 prvReward = multiplier.mul(prvPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
132         accPRVPerShare = accPRVPerShare.add(prvReward.mul(1e12).div(lpSupply));
133     }
134     return user.amount.mul(accPRVPerShare).div(1e12).sub(user.rewardDebt);
135 }
136
137 // Update reward variables for all pools. Be careful of gas spending!
138 function massUpdatePools() public {
139     uint256 length = poolInfo.length;
140     for (uint256 pid = 0; pid < length; ++pid) {
141         updatePool(pid);
142     }
143 }
144
145 // Update reward variables of the given pool to be up-to-date.
146 function updatePool(uint256 _pid) public {
147     PoolInfo storage pool = poolInfo[_pid];
148     if (block.number <= pool.lastRewardBlock) {
149         return;

```

```
150      }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	9	1164	154	374	636	74

Comments to Code $374/636 = 59\%$

Javascript Tests

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

Tests to Code $0/0 = 0\%$