

0.7

ShapeShift Process Quality Review

Score: 90%

Overview

This is [ShapeShift](#) Process Quality Review completed on the 26th of October, 2021. It was performed using the Process Review process (version 0.7.3) and is documented [here](#). The review was performed by Nick of DeFiSafety. Check out our [Telegram](#).

The final score of the review is **90%**, 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.

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

Chain

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



Chain: Ethereum, Bitcoin, Binance Smart Chain

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum
Solana

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%

All of the executing ShapeShift smart contracts are available in their respective GitHub repositories, 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 10+ transactions a day on contract [FOX Staking Rewards](#), 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/shapeshift>

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: 100%

At 2,654 commits and 3 branches, the Fox staking repository is the extensive foxhole network the developer's would like it to be.

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 |

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



Answer: Yes

Location: <https://github.com/shapeshift/foxfarm/graphs/contributors>

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://github.com/shapeshift/docs>

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



Answer: Yes

ShapeShift documents their basic software functions at <https://raw.githack.com/shapeshift/fox-staking->

[unified-history/master/docs/index.html#/](https://raw.githubusercontent.com/shapeshift/fox-staking-unified-history/master/docs/index.html#/).

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

 **Answer:** 100%

All of ShapeShift's main executing contracts and functions are documented at

<https://raw.githubusercontent.com/shapeshift/fox-staking-unified-history/master/docs/index.html#/>.

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:** 30%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 30% 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:** 40%

ShapeShift lists and describes all their functions at <https://raw.githubusercontent.com/shapeshift/fox-staking-unified-history/master/docs/index.html#/>.

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:** 100%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 200% 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

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

 **Answer:** 91%

As per <https://raw.githack.com/shapeshift/fox-staking-unified-history/master/coverage/index.html>, ShapeShift has a 91% code coverage.

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 be 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

Scripts/Instructions location: <https://github.com/shapeshift/docs/blob/main/testing.md>

14) Report of the results (%)

 **Answer:** 70%

A code coverage report is visible at <https://raw.githack.com/shapeshift/fox-staking-unified-history/master/coverage/index.html>.

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 formal verification was undertaken.

16) Stress Testing environment (%)

 **Answer:** 100%

FOX Staking has been deployed to [Rinkeby](#).

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%

Multiple audits were performed before the public deployment of multiple of their smart contracts. Notably, the [token](#), and the [staker](#), and [airdrop](#) contracts have all been audited pre-launch.

Note: Most of the fix recommendations outlined by the auditing team were implemented by the ShapeShift 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, (where question 1 is 0%)

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:** 20%

According to <https://shapeshift.com/responsible-disclosure-program>, ShapeShift has given approximately 40k in rewards for bug finds.

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%

Immutability of contracts and absence of owner privileges are clearly described at <https://github.com/shapeshift/fox-staking-unified-history>.

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%

Immutability of contracts and absence of owner privileges are clearly described at <https://github.com/shapeshift/fox-staking-unified-history>.

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%

21) Is the information in non-technical terms that pertain to the investments (%)

 **Answer:** 100%

Immutability of contracts and absence of owner privileges are clearly described at <https://github.com/shapeshift/fox-staking-unified-history>.

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

22) Is there Pause Control documentation including records of tests (%)

 **Answer:** 100%

Although no pause control function exists in the ShapeShift code, their smart contracts are entirely immutable, and the protocol functionalities that do not make pause control functions a necessity are described at <https://github.com/shapeshift/fox-staking-unified-history/blob/master/README.md>

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	ShapeShift	
		Points	Answer
PQ Audit Scoring Matrix (v0.7)	260		234.05
Code and Team			90%
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	100%	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	y	10
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	15	100%	15
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	5	30%	1.5
10) Is it possible to trace from software documentation to the implementation in code (%)	10	40%	4
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	91%	4.55
13) Scripts and instructions to run the tests? (Y/N)	5	y	5
14) Report of the results (%)	10	70%	7
15) Formal Verification test done (%)	5	0%	0
16) Stress Testing environment (%)	5	100%	5
Security			
17) Did 3rd Party audits take place? (%)	70	100%	70
18) Is the bug bounty acceptable high? (%)	10	20%	2
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	100%	
Documentation	45	79%	
Testing	50	83%	
Security	80	90%	
Access Controls	35	100%	

Executing Code Appendix

Staker

Farming contract based on <https://github.com/Uniswap/liquidity-staker>; currently deployed at 0xc54B9F82C1c54E9D4d274d633c7523f2299c42A0 on Ethereum Mainnet. The contract code is immutable, and there are no special access privileges (i.e. contract ownership) that would permit them to be altered or the rewards shut down.

Code Used Appendix



Example Code Appendix

```
1 pragma solidity ^0.7.6;
2
3 import '../openzeppelin-solidity-3.4.0/contracts/token/ERC20/IERC20.sol';
4 import '../openzeppelin-solidity-3.4.0/contracts/access/Ownable.sol';
5
6 import './StakingRewards.sol';
7
8 contract StakingRewardsFactory is Ownable {
9     // immutables
10    address public rewardsToken;
11    uint public stakingRewardsGenesis;
12
13    // the staking tokens for which the rewards contract has been deployed
14    address[] public stakingTokens;
15
16    // info about rewards for a particular staking token
17    struct StakingRewardsInfo {
18        address stakingRewards;
19        uint rewardAmount;
20    }
21
22    // rewards info by staking token
23    mapping(address => StakingRewardsInfo) public stakingRewardsInfoByStakingToken;
24
25    constructor(
26        address _rewardsToken,
27        uint _stakingRewardsGenesis
28    ) Ownable() {
29        require(_stakingRewardsGenesis >= block.timestamp, 'StakingRewardsFactory::constructor: genesis timestamp must be in the future');
30
31        rewardsToken = _rewardsToken;
32        stakingRewardsGenesis = _stakingRewardsGenesis;
33    }
34
35    //// permissioned functions
36
37    // deploy a staking reward contract for the staking token, and store the reward amount
38    // the reward will be distributed to the staking reward contract no sooner than the genesis timestamp
39    function deploy(address stakingToken, uint rewardAmount) external onlyOwner {
40        StakingRewardsInfo storage info = stakingRewardsInfoByStakingToken[stakingToken];
41        require(info.stakingRewards == address(0), 'StakingRewardsFactory::deploy: already deployed');
42
43        info.stakingRewards = address(new StakingRewards(/*_rewardsDistribution=*/ address(0)));
44        info.rewardAmount = rewardAmount;
45        stakingTokens.push(stakingToken);
46    }
47
48    //// permissionless functions
49
50    // call notifyRewardAmount for all staking tokens.
```

```
51     function notifyRewardAmounts() external {
52         require(stakingTokens.length > 0, 'StakingRewardsFactory::notifyRewardAmounts: call to StakingRewards contract failed');
53         for (uint i = 0; i < stakingTokens.length; i++) {
54             notifyRewardAmount(stakingTokens[i]);
55         }
56     }
57
58     // notify reward amount for an individual staking token.
59     // this is a fallback in case the notifyRewardAmounts costs too much gas to call for all tokens.
60     function notifyRewardAmount(address stakingToken) public {
61         require(block.timestamp >= stakingRewardsGenesis, 'StakingRewardsFactory::notifyRewardAmount: genesis time has not been reached');
62
63         StakingRewardsInfo storage info = stakingRewardsInfoByStakingToken[stakingToken];
64         require(info.stakingRewards != address(0), 'StakingRewardsFactory::notifyRewardAmount: stakingRewards address is zero');
65
66         if (info.rewardAmount > 0) {
67             uint rewardAmount = info.rewardAmount;
68             info.rewardAmount = 0;
69
70             require(
71                 IERC20(rewardsToken).transfer(info.stakingRewards, rewardAmount),
72                 'StakingRewardsFactory::notifyRewardAmount: transfer failed'
73             );
74             StakingRewards(info.stakingRewards).notifyRewardAmount(rewardAmount);
75         }
76     }
77 }
78
79 pragma solidity ^0.7.6;
80
81 import "../openzeppelin-solidity-3.4.0/contracts/math/Math.sol";
82 import "../openzeppelin-solidity-3.4.0/contracts/math/SafeMath.sol";
83 import "../openzeppelin-solidity-3.4.0/contracts/token/ERC20/SafeERC20.sol";
84 import "../openzeppelin-solidity-3.4.0/contracts/utils/ReentrancyGuard.sol";
85
86 // Inheritance
87 import "./interfaces/IStakingRewards.sol";
88 import "./RewardsDistributionRecipient.sol";
89
90 contract StakingRewards is IStakingRewards, RewardsDistributionRecipient, ReentrancyGuard {
91     using SafeMath for uint256;
92     using SafeERC20 for IERC20;
93
94     /* ===== STATE VARIABLES ===== */
95
96     IERC20 public rewardsToken;
97     IERC20 public stakingToken;
98     uint256 public periodFinish = 0;
99     uint256 public rewardRate = 0;
100    uint256 public rewardsDuration = 135 days;
101    uint256 public lastUpdateTime;
102    uint256 public rewardPerTokenStored;
103}
```

```
104     mapping(address => uint256) public userRewardPerTokenPaid;
105     mapping(address => uint256) public rewards;
106
107     uint256 private _totalSupply;
108     mapping(address => uint256) private _balances;
109
110     /* ===== CONSTRUCTOR ===== */
111
112     constructor(
113         address _rewardsDistribution,
114         address _rewardsToken,
115         address _stakingToken
116     ) {
117         rewardsToken = IERC20(_rewardsToken);
118         stakingToken = IERC20(_stakingToken);
119         rewardsDistribution = _rewardsDistribution;
120     }
121
122     /* ===== VIEWS ===== */
123
124     function totalSupply() external view override returns (uint256) {
125         return _totalSupply;
126     }
127
128     function balanceOf(address account) external view override returns (uint256) {
129         return _balances[account];
130     }
131
132     function lastTimeRewardApplicable() public view override returns (uint256) {
133         return Math.min(block.timestamp, periodFinish);
134     }
135
136     function rewardPerToken() public view override returns (uint256) {
137         if (_totalSupply == 0) {
138             return rewardPerTokenStored;
139         }
140         return
141             rewardPerTokenStored.add(
142                 lastTimeRewardApplicable().sub(lastUpdateTime).mul(rewardRate).mul(1e18).div(_totalSupply));
143     }
144
145
146     function earned(address account) public view override returns (uint256) {
147         return _balances[account].mul(rewardPerToken()).sub(userRewardPerTokenPaid[account]);
148     }
149
150     function getRewardForDuration() external view override returns (uint256) {
151         return rewardRate.mul(rewardsDuration);
152     }
153
154     /* ===== MUTATIVE FUNCTIONS ===== */
155
156     function stakeWithPermit(uint256 amount, uint deadline, uint8 v, bytes32 r, bytes32 s)
```

```
157     require(amount > 0, "Cannot stake 0");
158     _totalSupply = _totalSupply.add(amount);
159     _balances[msg.sender] = _balances[msg.sender].add(amount);
160
161     // permit
162     IUniswapV2ERC20(address(stakingToken)).permit(msg.sender, address(this), amount, d
163
164     stakingToken.safeTransferFrom(msg.sender, address(this), amount);
165     emit Staked(msg.sender, amount);
166 }
167
168 function stake(uint256 amount) external override nonReentrant updateReward(msg.sender)
169     require(amount > 0, "Cannot stake 0");
170     _totalSupply = _totalSupply.add(amount);
171     _balances[msg.sender] = _balances[msg.sender].add(amount);
172     stakingToken.safeTransferFrom(msg.sender, address(this), amount);
173     emit Staked(msg.sender, amount);
174 }
175
176 function withdraw(uint256 amount) public override nonReentrant updateReward(msg.sender)
177     require(amount > 0, "Cannot withdraw 0");
178     _totalSupply = _totalSupply.sub(amount);
179     _balances[msg.sender] = _balances[msg.sender].sub(amount);
180     stakingToken.safeTransfer(msg.sender, amount);
181     emit Withdrawn(msg.sender, amount);
182 }
183
184 function getReward() public override nonReentrant updateReward(msg.sender) {
185     uint256 reward = rewards[msg.sender];
186     if (reward > 0) {
187         rewards[msg.sender] = 0;
188         rewardsToken.safeTransfer(msg.sender, reward);
189         emit RewardPaid(msg.sender, reward);
190     }
191 }
192
193 function exit() external override {
194     withdraw(_balances[msg.sender]);
195     getReward();
196 }
197
198 /* ===== RESTRICTED FUNCTIONS ===== */
199
200 function notifyRewardAmount(uint256 reward) external override onlyRewardsDistribution
201     if (block.timestamp >= periodFinish) {
202         rewardRate = reward.div(rewardsDuration);
203     } else {
204         uint256 remaining = periodFinish.sub(block.timestamp);
205         uint256 leftover = remaining.mul(rewardRate);
206         rewardRate = reward.add(leftover).div(rewardsDuration);
207     }
208 }
```

```

209     // Ensure the provided reward amount is not more than the balance in the contract.
210
211     // This keeps the reward rate in the right range, preventing overflows due to
212     // very high values of rewardRate in the earned and rewardsPerToken functions;
213     // Reward + leftover must be less than 2^256 / 10^18 to avoid overflow.
214     uint balance = rewardsToken.balanceOf(address(this));
215     require(rewardRate <= balance.div(rewardsDuration), "Provided reward too high");
216
217     lastUpdateTime = block.timestamp;
218     periodFinish = block.timestamp.add(rewardsDuration);
219     emit RewardAdded(reward);
220 }
221
222 /* ===== MODIFIERS ===== */
223
224 modifier updateReward(address account) {
225     rewardPerTokenStored = rewardPerToken();
226     lastUpdateTime = lastTimeRewardApplicable();
227     if (account != address(0)) {
228         rewards[account] = earned(account);
229         userRewardPerTokenPaid[account] = rewardPerTokenStored;
230     }
231 }
232
233 /* ===== EVENTS ===== */
234
235 event RewardAdded(uint256 reward);
236 event Staked(address indexed user, uint256 amount);
237 event Withdrawn(address indexed user, uint256 amount);
238 event RewardPaid(address indexed user, uint256 reward);
239 }
240
241 interface IUniswapV2ERC20 {
242     function permit(address owner, address spender, uint value, uint deadline, uint8 v, by-
243 }
```

SLOC Appendix

Solidity Contracts

Comments to Code 56/190 = 30%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	4	479	77	21	381	10

Tests to Code 380/190 = 200%

