0.7

Ellipsis Finance Process Quality Review

Score: 60%

Overview

This is a Ellipsis Finance Process Quality Review completed on July 8th 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 60%, a close 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://docs.ellipsis.finance/deployment-links 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 300 transactions a day on contract *LpTokenStaker.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/ellipsis-finance.

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: 30%

With 38 commits and 1 branch, this is an acceptable 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 member 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://docs.ellipsis.finance/.

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

> ✓ Answer: Yes

Basic software functions found at https://docs.ellipsis.finance/dev/overview.

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

> ✓ Answer: 100%

All contract functions detailed in https://docs.ellipsis.finance/dev/overview.

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: 0%

Code examples are in the Appendix. As per the SLOC, there is 8% 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: 100%

Ellipsis Finance provides an explanation for every function, example code, ande GitHub links to the source code for a majority of the software functions at https://docs.ellipsis.finance/dev/overview.

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

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 22% 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 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: 30%

No code coverage but a clear presence of some testing.

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: Yes

Scripts and test instructions can be found at https://github.com/ellipsis-finance/ellipsis.

14) Report of the results (%)

Answer: 0%

No test report was found in their 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 Ellipsis Finance Formal Verification test was found in their documentation or on the web.

16) Stress Testing environment (%)

Answer: 0%

No evidence of any test-net smart contract use.

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%

Hacken published a Ellipsis Finance audit report on April 1st 2021.

Note 1: Ellipsis was launched in March 2021.

Note 2: Most fix recommendations were implemented by the devs.

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%

No Ellipsis Finance 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?
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: 100%

Clearly labelled as "Admin Multisig" in their documentation.

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) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
b) Multisig function and info is clearly detailed.

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%

Multisig info is all written in comprehensive language.

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%

No Pause Control or similar function was found in their documentation.

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	Ellipsis Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		155
Code and Team			**60%**
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**	30%	1.5
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**	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**	0%	0
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	100%	10
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**	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**	100%	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

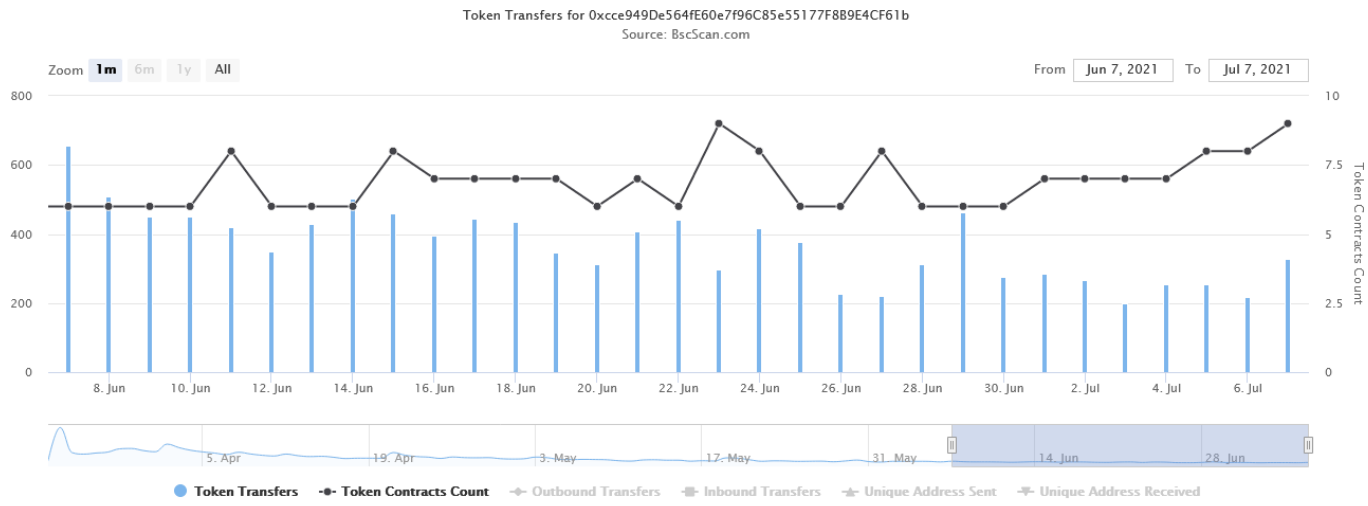
Section Scoring		
Code and Team	50	63%
Documentation	45	89%
Testing	50	29%
Security	80	61%
Access Controls	35	57%

Executing Code Appendix

Protocol

Name	Address
LpTokenStaker	0xcce949De564fE60e7f96C85e55177F8B9E4CF61b
EpsStaker	0x4076CC26EFeE47825917D0feC3A79d0bB9a6bB5c
MerkleDistributor	0x60a8AD8470189033789c1053B0c6F89eB27Bca18
FeeConverter	0x604348a7e98113E14c318605C458fBfB3B0Dd3f6
MetapoolFeeConverter	0xDd6dF5ffeD7b770355De53A9B60577b795A27B66

Code Used Appendix



Example Code Appendix

```
// SPDX-License-Identifier: MIT

pragma solidity 0.7.6;

```

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";
import "@openzeppelin/contracts/math/SafeMath.sol";
import "@openzeppelin/contracts/access/Ownable.sol";


interface Minter {
    function mint(address _receiver, uint256 _amount) external;
}

interface Oracle {
    function latestAnswer() external view returns (int256);
}

// LP token staking contract for http://ellipsis.finance/
// LP tokens are staked within this contract to generate EPS, Ellipsis' value-capture toke
// Based on the Sushi MasterChef contract by Chef Nomi - https://github.com/sushiswap/sush
contract LpTokenStaker is Ownable {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;
        uint256 rewardDebt;
    }
    // Info of each pool.
    struct PoolInfo {
        IERC20 lpToken; // Address of LP token contract.
        uint256 oracleIndex; // Index value for oracles array indicating which price multi
        uint256 allocPoint; // How many allocation points assigned to this pool.
        uint256 lastRewardTime; // Last second that reward distribution occurs.
        uint256 accRewardPerShare; // Accumulated rewards per share, times 1e12. See below
    }
    // Info about token emissions for a given time period.
    struct EmissionPoint {
        uint128 startTimeOffset;
        uint128 rewardsPerSecond;
    }

    Minter public rewardMinter;
    uint256 public rewardsPerSecond;
    // Info of each pool.
    PoolInfo[] public poolInfo;
    // Data about the future reward rates. emissionSchedule stored in reverse chronologica
    // whenever the number of blocks since the start block exceeds the next block offset a
    // reward rate is applied.
    EmissionPoint[] public emissionSchedule;
    // Info of each user that stakes LP tokens.
    mapping(uint256 => mapping(address => UserInfo)) public userInfo;
    // Total allocation poitns. Must be the sum of all allocation points in all pools.
    uint256 public totalAllocPoint = 0;
    // The block number when reward mining starts.
```

```
58
    uint256 public startTime;
59
60      // List of Chainlink oracle addresses.
61      Oracle[] public oracles;
62
63      event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
64      event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
65      event EmergencyWithdraw(
66          address indexed user,
67          uint256 indexed pid,
68          uint256 amount
69      );
70
71      constructor(
72          uint128[] memory _startTimeOffset,
73          uint128[] memory _rewardsPerSecond,
74          IERC20 _fixedRewardToken
75      ) public {
76          uint256 length = _startTimeOffset.length;
77          for (uint256 i = length - 1; i + 1 != 0; i--) {
78              emissionSchedule.push(
79                  EmissionPoint({
80                      startTimeOffset: _startTimeOffset[i],
81                      rewardsPerSecond: _rewardsPerSecond[i]
82                  })
83              );
84          }
85          // Pool values are based on USD so the first oracle is 0x00 and the price is always
86          oracles.push(Oracle(0));
87          // The first pool receives special treatment, it always has 20% of the totalAllocPo
88          poolInfo.push(
89              PoolInfo({
90                  lpToken: _fixedRewardToken,
91                  oracleIndex: 0,
92                  allocPoint: 0,
93                  lastRewardTime: block.timestamp,
94                  accRewardPerShare: 0
95              })
96          );
97      }
98
99      // Start the party
100     function start() public onlyOwner {
101         require(startTime == 0);
102         startTime = block.timestamp;
103     }
104
105     function setMinter(Minter _rewardMinter) public onlyOwner {
106         require(rewardMinter == Minter(0));
107         rewardMinter = _rewardMinter;
108     }
109
110     // Add a new lp to the pool. Can only be called by the owner
```

```
110     // Add a new lp to the pool. Can only be called by the owner.
111
        // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do
112     function addPool(IERC20 _lpToken, uint256 _oracleIndex) public onlyOwner {
113         require(_oracleIndex < oracles.length);
114         _massUpdatePools();
115         poolInfo.push(
116             PoolInfo({
117                 lpToken: _lpToken,
118                 oracleIndex: _oracleIndex,
119                 allocPoint: 0,
120                 lastRewardTime: block.timestamp,
121                 accRewardPerShare: 0
122             })
123         );
124     }
125
126     // Add a new oracle address. Should only be added if required by pool.
127     function addOracle(Oracle _oracle) external onlyOwner {
128         _oracle.latestAnswer();  // Validates that this is actually an oracle!
129         oracles.push(_oracle);
130     }
131
132     // Calculate the final allocation points for each pool.
133     // This is the main logical deviation from the original MasterChef contract.
134     // The pool at pid 0 always recieves exactly 20% of the allocation points.
135     // All remaining pools receive an "equal" allocation, based on their rough value in USD
136     // For pools handling USD-based Ellipsis LP tokens the value per token is assumed as $1
137     // for non-USD pools a rate is queried from a Chainlink oracle.
138     function _getAllocPoints() internal view returns (uint256[] memory allocPoints, uint256
139         // Get the oracle prices. Oracle[0] is USD and fixed at $1 (100000000)
140         uint256[] memory latestPrices = new uint256[](oracles.length);
141         latestPrices[0] = 100000000;
142         for (uint256 i = 1; i < oracles.length; i++) {
143             latestPrices[i] = uint256(oracles[i].latestAnswer());
144         }
145
146         // Apply oracle prices to calculate final allocation points for each pool
147         uint256 length = poolInfo.length;
148         allocPoints = new uint256[](length);
149         for (uint256 pid = 1; pid < length; ++pid) {
150             PoolInfo storage pool = poolInfo[pid];
151             allocPoints[pid] = pool.allocPoint.mul(latestPrices[pool.oracleIndex]);
152             totalAP = totalAP.add(allocPoints[pid]);
153         }
154         // Special treatment for pool 0 to always have 20%
155         totalAP = totalAP.mul(100).div(80);
156         allocPoints[0] = totalAP.div(5);
157
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	6	1360	189	114	1057	151
Vyper	6	4486	393	291	3802	196

Comments to Code 405/4859 = 8%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
Python	7	1428	362	0	1066	30

Tests to Code 1066/4859 = 22%