0.7

Pool Together 3.0 Process Quality Review

Score: 91%

Overview

This is a Pool Together Process Quality Review completed on 14/09/2021. 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 91%, 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.

Chain

This section indicates the blockchain used by this protocol.

> **Chain:** Ethereum, Polygon, Binance Smart Chain, Celo

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 websites https://docs.pooltogether.com/resources/networks/ethereum, https://docs.pooltogether.com/resources/networks/celo#celo, https://docs.pooltogether.com/resources/networks/xdai, https://docs.pooltogether.com/resources/networks/matic, https://docs.pooltogether.com/resources/networks/binance, as indicated in the Appendix.

Guidance: 100%

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 177 transactions a day on Polygon contract 0xEE06AbE9e2Af61cabcb13170e01266Af2DEFa946, 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/pooltogether

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%

With 878 commits and 30 branches, this is clearly a well-maintained 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

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

> **Answer:** Yes

Location: Protocol members are clearly listed in their medium articles.

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.pooltogether.com/

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

> **Answer:** Yes

https://docs.pooltogether.com/protocol/overview

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

> **Answer:** 100%

All contracts and functions are clearly explained with well-ordered and robust 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.

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:** 60%

There is nonexplicit traceability between the code and the documentation at a requirement level for all code.

Guidance:

100% Clear explicit traceability between code and documentation at a requirement level for all code
60% Clear association between code and documents via non explicit traceability
40% Documentation lists all the functions and describes their functions
0% No connection between documentation and code

How to improve this score

This score can improve by adding traceability from requirements to code such that it is clear where each requirement is coded. For reference, check the SecurEth guidelines on traceability.

Testing

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

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

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

> **Answer:** 100%

Code examples are in the Appendix. As per the SLOC, there is 171% 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:** 94%

Location: https://coveralls.io/github/pooltogether/pooltogether-pool-contracts?branch=master

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://github.com/pooltogether/pooltogether-pool-contracts

14) Report of the results (%)

> **Answer:** 70%

Coveralls test report can be found at https://coveralls.io/jobs/87058204.

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 formal verification.

16) Stress Testing environment (%)

> **Answer:** 100%

There is evidence of stress testing on the Rinkeby Testnet for many protocol versions.

https://docs.pooltogether.com/resources/subgraphs#lootbox-subgraphs

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%

PoolTogether has undergone three external audits. Open Zeppelin has audited the protocol twice, and ditCraft has once - though no proof could be found for this third audit.

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%

There is a bug bounty program offering up to $25,000.

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%

Controls are clearly detailed in the docs here: https://docs.pooltogether.com/governance/controls

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%

All Contracts are clearly labelled as non upgreadeable, AKA immutable.

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:** 100%

All contracts are clearly labelled as non-upgradeable.

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:** 100%

All contracts are clearly labelled as non-upgradeable.

Pause controls are mentioned in the audits performed by OpenZeppelin, but no evidence or detail on capabilities of tests.

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	Pool Together	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	260		236.6
Code and Team			**91%**
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%) c	15	100%	15
9) Are there sufficiently detailed comments for all function	5	38%	1.9
10) Is it possible to trace from software documentation to th	10	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
12) Code coverage (Covers all the deployed lines of code, or	5	94%	4.7

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 admi	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	10	100%	10
Section Scoring			
Code and Team	50	100%	
Documentation	45	84%	
Testing	50	83%	
Security	80	90%	
Access Controls	35	100%	

Executing Code Appendix

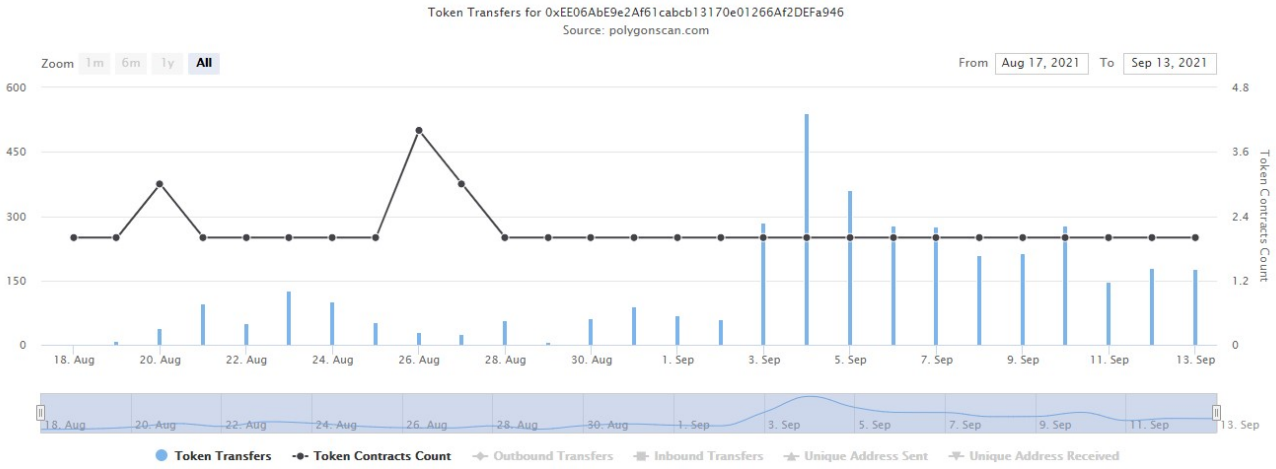
PoolTogether Pools & Supporting Contracts

@pooltogether/current-pool-data ^3.6.1 npm

Contract	Address
Dai Prize Pool	0xEBfb47A7ad0FD6e57323C8A42B2E5A6a4F68fc1a
Dai Prize Strategy	0x178969A87a78597d303C47198c66F68E8be67Dc2
Dai Pod	0x2f994e2E4F3395649eeE8A89092e63Ca526dA829
USDC Prize Pool	0xde9ec95d7708b8319ccca4b8bc92c0a3b70bf416
USDC Prize Strategy	0x3d9946190907ada8b70381b25c71eb9adf5f9b7b
USDC Pod	0x386EB78f2eE79AddE8Bdb0a0e27292755ebFea58
UNI Prize Pool	0x0650d780292142835F6ac58dd8E2a336e87b4393
UNI Prize Strategy	0xe8726B85236a489a8E84C56c95790d07a368f913
COMP Prize Pool	0xBC82221e131c082336cf698F0cA3EBd18aFd4ce7
COMP Prize Strategy	0x3ec4694b65e41f12d6b5d5ba7c2341f4d6859773
GUSD Prize Pool	0x65C8827229FbD63f9de9FDfd400C9D264066A336
GUSD Prize Strategy	0x821cF440654addD81493e1949F9ee078D65bb57f

POOL Prize Pool	0x396b4489da692788e327e2e4b2b0459a5ef26791
POOL Prize Strategy	0x21e5e62e0b6b59155110cd36f3f6655fbbcf6424
Loot Box ERC721	0x4d695c615a7AACf2d7b9C481B66045BB2457Dfde
Loot Box Prize Strategy Listener	0xfe7205DF55BA42c8801e44B55BF05F06cCe8565E
Aave USDT Prize Pool	0xc7d56c06F136EFff93e349C7BF8cc46bBF5D902c
Aave USDT Prize Strategy	0x2223d2e68e0990567f5e0451f4c027870ea07227

Code Used Appendix



Example Code Appendix

```
abstract contract PrizePool is PrizePoolInterface, OwnableUpgradeable, ReentrancyGuardUpgra
  using SafeMathUpgradeable for uint256;
  using SafeCastUpgradeable for uint256;
  using SafeERC20Upgradeable for IERC20Upgradeable;
  using SafeERC20Upgradeable for IERC721Upgradeable;
  using MappedSinglyLinkedList for MappedSinglyLinkedList.Mapping;
  using ERC165CheckerUpgradeable for address;

  /// @dev Emitted when an instance is initialized
  event Initialized(
    address reserveRegistry,
    uint256 maxExitFeeMantissa
  );

  /// @dev Event emitted when controlled token is added
  event ControlledTokenAdded(
    ControlledTokenInterface indexed token
```

```
  );

  /// @dev Emitted when reserve is captured.
  event ReserveFeeCaptured(
    uint256 amount
  );

  event AwardCaptured(
    uint256 amount
  );

  /// @dev Event emitted when assets are deposited
  event Deposited(
    address indexed operator,
    address indexed to,
    address indexed token,
    uint256 amount,
    address referrer
  );

  /// @dev Event emitted when interest is awarded to a winner
  event Awarded(
    address indexed winner,
    address indexed token,
    uint256 amount
  );

  /// @dev Event emitted when external ERC20s are awarded to a winner
  event AwardedExternalERC20(
    address indexed winner,
    address indexed token,
    uint256 amount
  );

  /// @dev Event emitted when external ERC20s are transferred out
  event TransferredExternalERC20(
    address indexed to,
    address indexed token,
    uint256 amount
  );

  /// @dev Event emitted when external ERC721s are awarded to a winner
  event AwardedExternalERC721(
    address indexed winner,
    address indexed token,
    uint256[] tokenIds
  );

  /// @dev Event emitted when assets are withdrawn instantly
  event InstantWithdrawal(
    address indexed operator,
    address indexed from,
    address indexed token,
```

```
    uint256 amount,
    uint256 redeemed,
    uint256 exitFee
  );

  event ReserveWithdrawal(
    address indexed to,
    uint256 amount
  );

  /// @dev Event emitted when the Liquidity Cap is set
  event LiquidityCapSet(
    uint256 liquidityCap
  );

  /// @dev Event emitted when the Credit plan is set
  event CreditPlanSet(
    address token,
    uint128 creditLimitMantissa,
    uint128 creditRateMantissa
  );

  /// @dev Event emitted when the Prize Strategy is set
  event PrizeStrategySet(
    address indexed prizeStrategy
  );

  /// @dev Emitted when credit is minted
  event CreditMinted(
    address indexed user,
    address indexed token,
    uint256 amount
  );

  /// @dev Emitted when credit is burned
  event CreditBurned(
    address indexed user,
    address indexed token,
    uint256 amount
  );

  /// @dev Emitted when there was an error thrown awarding an External ERC721
  event ErrorAwardingExternalERC721(bytes error);


  struct CreditPlan {
    uint128 creditLimitMantissa;
    uint128 creditRateMantissa;
  }

  struct CreditBalance {
    uint192 balance;
    uint32 timestamp;
```

```
    bool initialized;
  }

  /// @notice Semver Version
  string constant public VERSION = "3.4.5";

  /// @dev Reserve to which reserve fees are sent
  RegistryInterface public reserveRegistry;

  /// @dev An array of all the controlled tokens
  ControlledTokenInterface[] internal _tokens;

  /// @dev The Prize Strategy that this Prize Pool is bound to.
  TokenListenerInterface public prizeStrategy;

  /// @dev The maximum possible exit fee fraction as a fixed point 18 number.
  /// For example, if the maxExitFeeMantissa is "0.1 ether", then the maximum exit fee for
  uint256 public maxExitFeeMantissa;

  /// @dev The total funds that have been allocated to the reserve
  uint256 public reserveTotalSupply;

  /// @dev The total amount of funds that the prize pool can hold.
  uint256 public liquidityCap;

  /// @dev the The awardable balance
  uint256 internal _currentAwardBalance;

  /// @dev Stores the credit plan for each token.
  mapping(address => CreditPlan) internal _tokenCreditPlans;

  /// @dev Stores each users balance of credit per token.
  mapping(address => mapping(address => CreditBalance)) internal _tokenCreditBalances;

  /// @notice Initializes the Prize Pool
  /// @param _controlledTokens Array of ControlledTokens that are controlled by this Prize
  /// @param _maxExitFeeMantissa The maximum exit fee size
  function initialize (
    RegistryInterface _reserveRegistry,
    ControlledTokenInterface[] memory _controlledTokens,
    uint256 _maxExitFeeMantissa
  )
    public
    initializer
  {
    require(address(_reserveRegistry) != address(0), "PrizePool/reserveRegistry-not-zero")
    uint256 controlledTokensLength = _controlledTokens.length;
    _tokens = new ControlledTokenInterface[](controlledTokensLength);

    for (uint256 i = 0; i < controlledTokensLength; i++) {
      ControlledTokenInterface controlledToken = _controlledTokens[i];
      _addControlledToken(controlledToken, i);
    }
```

```
177       __Ownable_init();
178       __ReentrancyGuard_init();
179       _setLiquidityCap(uint256(-1));
180
181       reserveRegistry = _reserveRegistry;
182       maxExitFeeMantissa = _maxExitFeeMantissa;
183
184       emit Initialized(
185         address(_reserveRegistry),
186         maxExitFeeMantissa
187       );
188     }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	36	3762	612	869	2281	217

Comments to Code 869/2281 = 38%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	44	5125	1059	161	3905	65

Tests to Code 3905/2281 = 171%