0.7

Pickle Finance V2 Process Quality Review

Scoring: 66%

Overview

This is a Pickle Finance V2 Process Quality Review completed on June 14th 2021. A Pickle Finance V1 review was written previously, and you can find it here. 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 66%, almost 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**

Guidance:
Ethereum
Binance

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://github.com/pickle-finance/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 800 transactions a day on contract *MasterChefV2*, 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/pickle-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: 100%

With 255 commits and 25 branches, this is a very healthy 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 10 commits

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

⚠ Answer: No

Team is anonymous.

For a yes in this question the real names of some team members must be public on the website or other documentation. 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.pickle.finance/.

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

✓ Answer: Yes

Basic capability and requirements are documented at https://docs.pickle.finance/farms/farms. This document mentions some basic functions, but does not describe them fully.

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

⚠ Answer: 20%

Only top level requirements are documented at https://docs.pickle.finance/farms/farms so a 20% overall score is given.

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: 75%

While the CtC is 28%, reading MasterChef shows excellent quality. Score adapted to 75%

Code examples are in the Appendix. As per the SLOC, there is 28% 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%

Some traceability in https://docs.pickle.finance/jars/strategies, but not much traceability anywhere else.

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: 80%

Code examples are in the Appendix. As per the SLOC, there is 90% 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: 50%

No code coverage tests or reports found in their repo, but there are clearly a robust set of tests.

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 instructions to run tests can be found at https://github.com/pickle-finance/protocol.

14) Report of the results (%)

> ⚠ Answer: 0%

There are no apparent test reports in their software 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 evidence of a formal verification for Pickle Finance.

16) Stress Testing environment (%)

⚠ Answer: 0%

I could not find any test-net smart contract addresses.

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%

PeckShield did a Pickle Finance audit on March 4th 2021.

MixBytes did a Pickle Finance audit on October 9th 2020.

MixBytes released a second Pickle Finance audit on November 5th 2020.

Haechi released a Pickle Finance audit on November 9th 2020.

Pickle was launched on September 10th 2020.

Notes: most changes were implemented.

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: 50%

Bug Bounty program found at https://github.com/pickle-finance/protocol/blob/master/SECURITY.md#bug-bounty-program.

Rewards go up to 50k USD.

Note: Currently there is no set end date for the program, therefore it is currently active.

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%

Found at https://docs.pickle.finance/faqs/security.

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: 90%

a) The general DAO and voting system of Pickle Finance indicates the capabilities of contract upgradeability.

b) Voting roles and MultiSig roles are clearly defined.

c) The capabilities for admin change in contract is described in their MasterChef contract.

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: 30%

It's basically written in code, and code commenting, as they link their MasterChef contract for a description of admin control capabilities.

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 documentation or test were found.

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	Pickle Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		172.25
Code and Team			**66%**
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**	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**	20%	3
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	75%	3.75
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**	80%	16
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	50%	2.5
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**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	50%	5
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**	90%	9
21) Is the information in non-technical terms	**10**	30%	3
22) Is there Pause Control documentation including records of tests	**10**	0%	0
Section Scoring			
Code and Team	50	70%	
Documentation	45	48%	
Testing	50	47%	
Security	80	94%	
Access Controls	35	49%	

Executing Code Appendix

DILL Contracts

Main Contracts

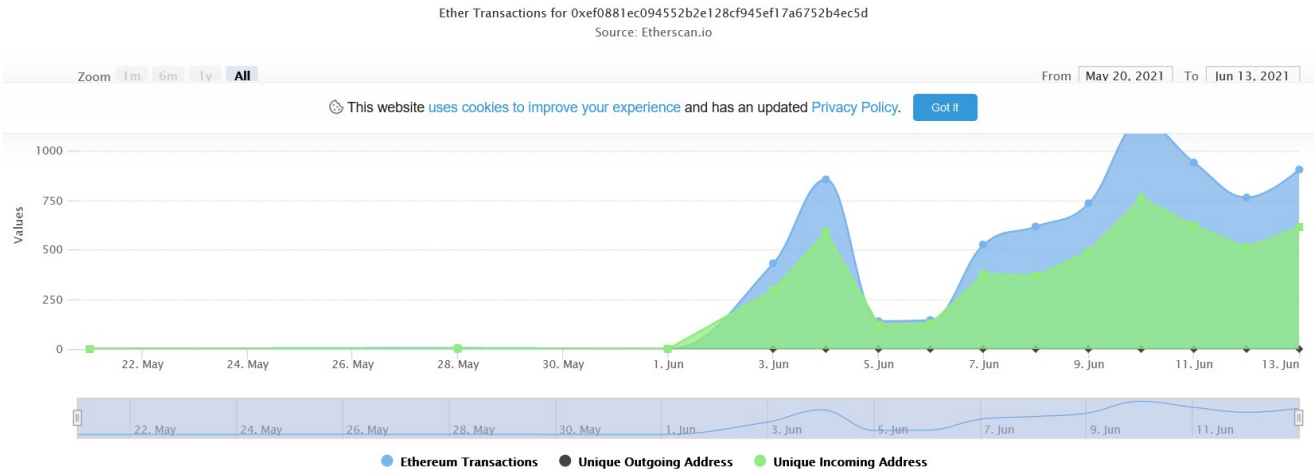
DILL: 0xbBCf169eE191A1Ba7371F30A1C344bFC498b29Cf

GaugeProxy: 0x2e57627ACf6c1812F99e274d0ac61B786c19E74f

mDILL: 0x45F7fa97BD0e0C212A844BAea35876C7560F465B

FeeDistributor: 0x74C6CadE3eF61d64dcc9b97490d9FbB231e4BdCc

Code Used Appendix



Example Code Appendix

```
pragma solidity 0.6.7;

import "../lib/enumerableSet.sol";
import "../lib/safe-math.sol";
import "../lib/erc20.sol";
import "../lib/ownable.sol";
import "./pickle-token.sol";

// MasterChef was the master of pickle. He now governs over PICKLES. He can make Pickles a
//
// Note that it's ownable and the owner wields tremendous power. The ownership
// will be transferred to a governance smart contract once PICKLES is sufficiently
// distributed and the community can show to govern itself.
//
// Have fun reading it. Hopefully it's bug-free. God bless.

```

```
contract MasterChef is Ownable {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

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

    // Info of each pool.
    struct PoolInfo {
        IERC20 lpToken; // Address of LP token contract.
        uint256 allocPoint; // How many allocation points assigned to this pool. PICKLEs t
        uint256 lastRewardBlock; // Last block number that PICKLEs distribution occurs.
        uint256 accPicklePerShare; // Accumulated PICKLEs per share, times 1e12. See below
    }

    // The PICKLE TOKEN!
    PickleToken public pickle;
    // Dev fund (2%, initially)
    uint256 public devFundDivRate = 50;
    // Dev address.
    address public devaddr;
    // Block number when bonus PICKLE period ends.
    uint256 public bonusEndBlock;
    // PICKLE tokens created per block.
    uint256 public picklePerBlock;
    // Bonus muliplier for early pickle makers.
    uint256 public constant BONUS_MULTIPLIER = 10;

    // Info of each pool.
    PoolInfo[] public poolInfo;
    // Info of each user that stakes LP tokens.
    mapping(uint256 => mapping(address => UserInfo)) public userInfo;
    // Total allocation points. Must be the sum of all allocation points in all pools.
    uint256 public totalAllocPoint = 0;
    // The block number when PICKLE mining starts.
    uint256 public startBlock;

    // Events
    event Recovered(address token, uint256 amount);
```

```
    event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
    event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
    event EmergencyWithdraw(
        address indexed user,
        uint256 indexed pid,
        uint256 amount
    );

    constructor(
        PickleToken _pickle,
        address _devaddr,
        uint256 _picklePerBlock,
        uint256 _startBlock,
        uint256 _bonusEndBlock
    ) public {
        pickle = _pickle;
        devaddr = _devaddr;
        picklePerBlock = _picklePerBlock;
        bonusEndBlock = _bonusEndBlock;
        startBlock = _startBlock;
    }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	107	10677	1825	1919	6933	615

Comments to Code 1919/6933 = 28%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	63	8156	1567	329	6260	97

Tests to Code 6260/6933 = 90%