

PancakeSwap Process Quality Review

Score: 42%

Overview

This is a [PancakeSwap](#) Process Quality Review of PancakeSwap completed on 17 April 2021. It was performed using the Process Review process (version 0.7) and is documented [here](#). The review was performed by Rex of DeFiSafety. Check out our [Telegram](#).

The final score of the review is 42%, a hard fail. The breakdown of the scoring is in [Scoring Appendix](#). For our purposes, a pass is 70%.

Summary of the Process

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

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

Disclaimer

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and

financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

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

Chain

This section indicates the blockchain used by this protocol. The chains and their multiples are explained in this [document](#).

 **Chain: Binance**

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: 70%

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

They are available at website <https://github.com/pancakeswap/pancake-farm> as indicated in the [Appendix](#).

How to improve this score

Make the Ethereum addresses of the smart contract utilized by your application available on either your website or your GitHub (in the README for instance). Ensure the addresses is up to date. This is a very important question wrt to the final score.

2) Is the code actively being used? (%)

 Answer: 100%

Activity is 10,000 transactions a day on contract *MasterChef.sol*, as indicated in the [Appendix](#).

Percentage Score Guidance

- 100% More than 10 transactions a day
- 70% More than 10 transactions a week
- 40% More than 10 transactions a month
- 10% Less than 10 transactions a month
- 0% No activity

3) Is there a public software repository? (Y/N)

 Answer: Yes

GitHub: <https://github.com/pancakeswap>

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 104 commits, this is clearly a robust 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

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

The team for pancakeswap seem anon.

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.pancakeswap.finance/>

How to improve this score

Ensure the white paper is available for download from your website or at least the software repository. Ideally update the whitepaper to meet the capabilities of your present application.

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



Answer: No

There is no evidence of software function documentation.

How to improve this score

Write the document based on the deployed code. For guidance, refer to the [SecurEth System Description Document](#).

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



Answer: 0%

There is no evident software function documentation.

Guidance:

- 100% All contracts and functions documented
- 80% Only the major functions documented
- 79-1% Estimate of the level of software documentation
- 0% No software documentation

How to improve this score

This score can improve by adding content to the requirements document such that it comprehensively covers the requirements. For guidance, refer to the [SecurEth System Description Document](#). Using tools that aid traceability detection will help.

9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)



Answer: 26%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 26% 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%

There is no evidence of software function documentation

Guidance:

100% - Clear explicit traceability between code and documentation at a requirement level for all code

60% - Clear association between code and documents via non explicit traceability

40% - Documentation lists all the functions and describes their functions

0% - No connection between documentation and code

How to improve this score


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

Testing

This section looks at the software testing available. It is explained in this [document](#). This section answers the following questions;

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

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

 Answer: 100%

While a TtC of 58% does not indicate a full test suite, the code coverage does, so we will give them the benefit of the doubt

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: 90%

According to [CodeCov.io](https://codecov.io) Pancakeswap has a 90% 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 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

The readme does explain.

14) Report of the results (%)

 Answer: 70%

Using the build and test records we have a basic test report which has a list of tests performed, basic status (pass/fail) but no coverage information.

(https://github.com/pancakeswap/pancake-farm/runs/2181505305?check_suite_focus=true)

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%

16) Stress Testing environment (%)

 Answer: 0%

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: 40%

Certik has done an audit on [PancakeSwap](#). The audit was done after deployment. The audit is not linked in the pancake swap website, github or docs. A major flaw in the syrup tokens was found and is not resolved in the deployed code, though they deprecated the pools. For these reasons a score of 40% is given.

Guidance:

1. Multiple Audits performed before deployment and results public and implemented or not required (100%)
2. Single audit performed before deployment and results public and implemented or not required (90%)
3. Audit(s) performed after deployment and no changes required. Audit report is public. (70%)
4. No audit performed (20%)
5. Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, question 1 (0%)

18) Is the bounty value acceptably high (%)

 Answer: 100%

Critical is up to \$1M.

Bug Bounty Location: <https://docs.pancakeswap.finance/code/bug-bounty>

Guidance:

- 100% Bounty is 10% TVL or at least 1M
- 90% Bounty is 5% TVL or at least 500k
- 70% Bounty is 100k or over
- 40% Bounty is 50k or over
- 20% Bug bounty program bounty is less than 50k
- 0% No bug bounty program offered

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 admin controls (%)

 Answer: 0%

While there is a voting section. I could not find any docs on what upgradability does or how.

Location:

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: 0%

With no information, no points awarded.

Guidance:

All the contracts are immutable -- 100% OR

All contracts are clearly labelled as upgradeable (or not) -- 30% AND

The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND

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: 0%

No admin control docs.

Guidance:

100% All the contracts are immutable

90% Description relates to investments safety and updates in clear, complete non-software 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 documentation on pause capability, though I suspect it exists.

Guidance:

100% All the contracts are immutable or no pause control needed and this is explained OR

100% Pause control(s) are clearly documented and there is records of at least one test within 3 months

80% Pause control(s) explained clearly but no evidence of regular tests

40% Pause controls mentioned with no detail on capability or tests

0% Pause control not documented or explained

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An [example](#) is enclosed.

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : [@defisafety](https://twitter.com/defisafety)

I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started [SecuEth.org](https://www.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

PQ Audit Scoring Matrix (v0.7)	Total	Pancake Swap		
	Points	Answer	Points	
Code and Team				42%
1) Are the executing code addresses readily available? (%)	20	8%	70%	14
2) Is the code actively being used? (%)	5	2%	100%	5
3) Is there a public software repository? (Y/N)	5	2%	y	5
4) Is there a development history visible? (%)	5	2%	100%	5
5) Is the team public (not anonymous)? (Y/N)	15	6%	N	0
Code Documentation				19%
6) Is there a whitepaper? (Y/N)	5	2%	y	5
7) Are the basic software functions documented? (Y/N)	10	4%	N	0
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	15	6%	0%	0
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	5	2%	26%	1.3
10) Is it possible to trace from software documentation to the implementation in code (%)	10	4%	0%	0
Testing				17%
11) Full test suite (Covers all the deployed code) (%)	20	8%	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	2%	90%	4.5
13) Scripts and instructions to run the tests? (Y/N)	5	2%	Y	5
14) Report of the results (%)	10	4%	70%	7
15) Formal Verification test done (%)	5	2%	0%	0
16) Stress Testing environment (%)	5	2%	0%	0
Security				19%
17) Did 3rd Party audits take place? (%)	70	27%	40%	28
18) Is the bug bounty acceptable high? (%)	10	4%	100%	10
Access Controls				31%
19) Can a user clearly and quickly find the status of the admin controls	5	2%	0%	0
20) Is the information clear and complete	10	4%	0%	0
21) Is the information in non-technical terms	10	4%	0%	0
22) Is there Pause Control documentation including records of tests	10	4%	0%	0
				13%
Section Scoring				
Code and Team	50		58%	
Documentation	45		14%	
Testing	50		73%	
Audits	80		40%	
Access Controls	35		0%	

Executing Code Appendix

PanCake Farming

CI passing
codecov 90%

<https://pancakeswap.finance>. Feel free to read the code. More details coming soon.

Deployed Contracts / Hash

BSCMAINNET

- CakeToken - <https://bscscan.com/token/0x0e09fabb73bd3ade0a17ecc321fd13a19e81ce82>
- MasterChef - <https://bscscan.com/address/0x73feaa1eE314F8c655E354234017bE2193C9E24E>
- (Uni|Cake)swapV2Factory - <https://bscscan.com/address/0xBCfCcbde45cE874adCB698cC183deBcF17952812>
- (Uni|Cake)swapV2Router02 - <https://bscscan.com/address/0x05ff2B0DB69458A0750badebc4f9e13aDd608C7F>
- (Uni|Cake)swapV2Pair init code hash - `0xd0d4c4cd0848c93cb4fd1f498d7013ee6bfb25783ea21593d5834f5d250ece66`
- MultiCall - `0xE1dDc30f691CA671518090931e3bFC1184BFa4Aa`

Code Used Appendix



Example Code Appendix

```

1 pragma solidity =0.5.16;
2
3 import './interfaces/IPancakeERC20.sol';
4 import './libraries/SafeMath.sol';
5
6 contract PancakeERC20 is IPancakeERC20 {
7     using SafeMath for uint;
8
9     string public constant name = 'Pancake LPs';
10    string public constant symbol = 'Cake-LP';
11    uint8 public constant decimals = 18;
12    uint public totalSupply;
13    mapping(address => uint) public balanceOf;
14    mapping(address => mapping(address => uint)) public allowance;
15
16    bytes32 public DOMAIN_SEPARATOR;
17    // keccak256("Permit(address owner,address spender,uint256 value,uint256
18    bytes32 public constant PERMIT_TYPEHASH = 0x6e71edae12b1b97f4d1f60370fe
19    mapping(address => uint) public nonces;
20
21    event Approval(address indexed owner, address indexed spender, uint value);
22    event Transfer(address indexed from, address indexed to, uint value);
23
24    constructor() public {

```



```

25     uint chainId;
26     assembly {
27         chainId := chainid
28     }
29     DOMAIN_SEPARATOR = keccak256(
30         abi.encode(
31             keccak256('EIP712Domain(string name,string version,uint256 (
32             keccak256(bytes(name)),
33             keccak256(bytes('1')),
34             chainId,
35             address(this)
36         )
37     );
38 }
39
40 function _mint(address to, uint value) internal {
41     totalSupply = totalSupply.add(value);
42     balanceOf[to] = balanceOf[to].add(value);
43     emit Transfer(address(0), to, value);
44 }
45
46 function _burn(address from, uint value) internal {
47     balanceOf[from] = balanceOf[from].sub(value);
48     totalSupply = totalSupply.sub(value);
49     emit Transfer(from, address(0), value);
50 }
51
52 function _approve(address owner, address spender, uint value) private {
53     allowance[owner][spender] = value;
54     emit Approval(owner, spender, value);
55 }
56
57 function _transfer(address from, address to, uint value) private {
58     balanceOf[from] = balanceOf[from].sub(value);
59     balanceOf[to] = balanceOf[to].add(value);
60     emit Transfer(from, to, value);
61 }
62
63 function approve(address spender, uint value) external returns (bool) {
64     _approve(msg.sender, spender, value);
65     return true;
66 }
67
68 function transfer(address to, uint value) external returns (bool) {
69     _transfer(msg.sender, to, value);
70     return true;
71 }
72
73 function transferFrom(address from, address to, uint value) external returns (bool) {
74     if (allowance[from][msg.sender] != uint(-1)) {
75         allowance[from][msg.sender] = allowance[from][msg.sender].sub(value);
76     }
77     _transfer(from, to, value);
78     return true;
79 }

```

```

80
81     function permit(address owner, address spender, uint value, uint deadli
82         require(deadline >= block.timestamp, 'Pancake: EXPIRED');
83         bytes32 digest = keccak256(
84             abi.encodePacked(
85                 '\x19\x01',
86                 DOMAIN_SEPARATOR,
87                 keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, value
88             )
89         );
90         address recoveredAddress = ecrecover(digest, v, r, s);
91         require(recoveredAddress != address(0) && recoveredAddress == owner
92             _approve(owner, spender, value);
93     }
94 }

```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	12	1715	264	300	1151	140

Comments to Code 300/1151 = 26%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	7	773	83	15	675	0

Tests to Code 675/1151 = 58%