0.7

PayBSwap Finance Process Quality Review

Score: 21%

Overview

This is a PayBSwap Finance Process Quality Review completed on August 31st 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 **21%**, a **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, Ethereum

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra

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:** 0%

None of the executing smart contract addresses (Governor, Controller, Vault, etc.) are provided in the PayBSwap documentation. Only the token address is listed, but we do not factor that in for the scoring of this metric.

Guidance:

100% Clearly labelled and on website, docs or repo, quick to find

70%	Clearly labelled and on website, docs or repo but takes a bit of looking
40%	Addresses in mainnet.json, in discord or sub graph, etc
20%	Address found but labeling not clear or easy to find
0%	Executing addresses could not be found

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 towards the final score.

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

> ⚠ **Answer:** 0%

Only the token address is listed, but we do not factor that in for the scoring of this metric.

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:** No

PayBSwap's GitHub only holds one repository: a trustwallet fork. Their source code is therefore private.

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:** 0%

PayBSwap's GitHub only holds one repository: a trustwallet fork. Therefore, we cannot check their source code's development history.

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

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:** Yes

Location: https://www.linkedin.com/search/results/people/?currentCompany=%5B%2271759000%22%5D&origin=COMPANY_PAGE_CANNED_SEARCH&sid=j~%3B.

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://paybswap.finance/static/media/Whitepaper.f1d731e1.pdf.

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

⚠ **Answer:** No

There are no software functions documented in any of the PayBSwap 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 are no software functions documented in any of the PayBSwap 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 be improved by adding content to the software functions 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:** 0%

As PayBSwap's source code is private, we cannot evaluate the CtC ratio.

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%

As PayBSwap's source code is private, we cannot evaluate the traceability between the software documentation and its implementation in their source 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 documentation to code such that it is clear where each outlined function is coded in the source code. For reference, check the SecurEth guidelines on traceability.

Testing

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

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

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

>

> ⚠ **Answer:** 0%

As PayBSwap's source code is private, we cannot evaluate the TtC ratio.

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 improved 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:** 0%

There is no evidence of any PayBSwap code coverage in their documentation or GitHub repository.

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:** No

We cannot currently verify this due to the private GitHub.

How to improve this score:

Add the scripts to the repository and ensure they work. Ask an outsider to create the environment and run the tests. Improve the scripts and docs based on their feedback.

14) Report of the results (%)

> ⚠ **Answer:** 0%

We cannot currently verify this due to the private GitHub.

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 a PayBSwap Formal Verification test in any of their documentation or in further web searches.

16) Stress Testing environment (%)

> ⚠ **Answer:** 0%

There is no evidence of any PayBSwap testnet smart contract usage.

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:** 35%

Someone published a PayBSwap audit report on an undisclosed date. Author is also not specified. Also the audit was just on the token contract.

Quillhash team published a PayBSwap token audit report on an undisclosed date. However, the audit was completed on April 1st, which is before the launch of the token on April 16th 2021.

Note: No indication as to the fix recommendations' implementation.

Note 2: Private repo = -25%.

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
60% Audit(s) performed before deployment an changes needed but not implemented

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:** 0%

No PayBSwap bug bounty program was found in any of their documentation or in further web searches.

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:** 40%

In the PayBSwap whitepaper, you need to go manually find the governance section by scrolling through it, so it already takes a bit of looking. In addition, there are two governance sections, and you can only find that out if you keep scrolling after the first one as there is some distance between the two. Roles are also under "Protocol actors".

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) Contracts labelled as upgradeable

b) MultiSig mentioned but no proof thereof. However, the roles are defined.

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%

There is practically no tangible information about user investments' safety in the PayBSwap documentation.

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 evidence of a Pause Control or similar function was found in any of the PayBSwap 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	PayBSwap Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		55.5
Code and Team			**21%**
1) Are the executing code addresses readily available? (%)	**20**	0%	0
2) Is the code actively being used? (%)	**5**	0%	0
3) Is there a public software repository? (Y/N)	**5**	N	0
4) Is there a development history visible? (%)	**5**	0%	0
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**	N	0
8) Does the software function documentation fully (100%) cov	**15**	0%	0
9) Are there sufficiently detailed comments for all functions w	**5**	0%	0
10) Is it possible to trace from software documentation to the	**10**	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	0%	0
12) Code coverage (Covers all the deployed lines of code, or ex	**5**	0%	0
13) Scripts and instructions to run the tests? (Y/N)	**5**	0	0
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**	35%	24.5
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	**5**	40%	2
20) Is the information clear and complete	**10**	60%	6
21) Is the information in non-technical terms	**10**	30%	3
22) Is there Pause Control documentation including records of	**10**	0%	0
Section Scoring			
Code and Team	50	30%	
Documentation	45	11%	
Testing	50	0%	
Security	80	31%	
Access Controls	35	31%	

Executing Code Appendix

N/A

Code Used Appendix

N/A

Example Code Appendix

```
contract BEP20TokenPAYB is Context, IBEP20, Ownable {
    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    /**
     * @dev Sets the values for {name} and {symbol}.
     *
     * The defaut value of {decimals} is 18. To select a different value for
     * {decimals} you should overload it.
     *
     * All three of these values are immutable: they can only be set once during
     * construction.
     */
    constructor (address _owner) Ownable(_owner) {
        require(_owner != address(0), "BEP20: transfer from the zero address");
        _totalSupply = 1_000_000_000 * 10 ** uint256(decimals());
        _name = "Paybswap";
        _symbol = "PAYB";
```

```
        _balances[_owner] = _totalSupply;
        emit Transfer(address(0) , _owner, _totalSupply);
    }

    /**
     * @dev Returns the name of the token.
     */
    function name() external view virtual override returns (string memory) {
        return _name;
    }

    /**
     * @dev Returns the symbol of the token, usually a shorter version of the
     * name.
     */
    function symbol() external view virtual override returns (string memory) {
        return _symbol;
    }

    /**
     * @dev Returns the number of decimals used to get its user representation.
     * For example, if `decimals` equals `2`, a balance of `505` tokens should
     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
     *
     * Tokens usually opt for a value of 18, imitating the relationship between
     * Ether and Wei. This is the value {BEP20} uses, unless this function is
     * overloaded;
     *
     * NOTE: This information is only used for _display_ purposes: it in
     * no way affects any of the arithmetic of the contract, including
     * {IBEP20-balanceOf} and {IBEP20-transfer}.
     */
    function decimals() public view virtual override returns (uint8) {
        return 18;
    }

    /**
     * @dev Returns the bep token owner.
     * https://github.com/binance-chain/BEPs/blob/master/BEP20.md
     */
    function getOwner() external virtual override view returns (address) {
        return owner();
    }

    /**
     * @dev See {IBEP20-totalSupply}.
     */
    function totalSupply() external view virtual override returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev See {IBEP20-balanceOf}.
```

```
     */
    function balanceOf(address account) external view virtual override returns (uint256) {
        return _balances[account];
    }

    /**
     * @dev See {IBEP20-transfer}.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
     */
    function transfer(address recipient, uint256 amount) external virtual override returns
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    /**
     * @dev See {IBEP20-allowance}.
     */
    function allowance(address owner, address spender) external view virtual override retu
        return _allowances[owner][spender];
    }

    /**
     * @dev See {IBEP20-approve}.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 amount) external virtual override returns (b
        _approve(_msgSender(), spender, amount);
        return true;
    }

    /**
     * @dev See {IBEP20-transferFrom}.
     *
     * Emits an {Approval} event indicating the updated allowance. This is not
     * required by the EIP. See the note at the beginning of {BEP20}.
     *
     * Requirements:
     *
     * - `sender` and `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     * - the caller must have allowance for ``sender``'s tokens of at least
     * `amount`.
     */
    function transferFrom(address sender, address recipient, uint256 amount) external virtu
        _transfer(sender, recipient, amount);

```

```
131         require(_allowances[sender][_msgSender()] >= amount, "BEP20: transfer amount exceed
132         _approve(sender, _msgSender(), _allowances[sender][_msgSender()] - amount);
133 
134         return true;
135     }
136 
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	N/A	N/A	N/A	N/A	N/A	N/A

Comments to Code = N/A

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	N/A	N/A	N/A	N/A	N/A	N/A

Tests to Code = N/A