0.7

DFYN Process Quality Review

Score: 56%

Overview

This is a DFYN Process Quality Review completed on June 21st 2021. It was performed using the Process Review process (version 0.7.2) and is documented here. The review was performed by of DeFiSafety. Check out our Telegram.

The final score of the review is 56%, a close fail. The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is 70%.

Summary of the Process

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

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

Disclaimer

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

Chain

This section indicates the blockchain used by this protocol.

✓ **Chain: Polygon**

Guidance:
Ethereum
Binance Smart Chain
Polygon

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is here. This review will answer the questions;

1) Are the executing code addresses readily available? (%)
2) Is the code actively being used? (%)
3) Is there a public software repository? (Y/N)
4) Is there a development history visible? (%)
5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

✓ Answer: 100%

They are available at website https://docs.dfyn.network/technical/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 3500 transactions a day on contract *UniSwapV2Router02*, 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/dfyn.

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: 0%

With 5 commits and 2 branches, this is an unhealthy software repository.

This checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:
100%   Any one of 100+ commits, 10+branches
70%   Any one of 70+ commits, 7+branches
50%   Any one of 50+ commits, 5+branches
30%   Any one of 30+ commits, 3+branches
0%   Less than 2 branches or less than 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: Yes

Team info can be found on page 30 of https://dfyn.network/assets/docs/Dfyn-Litepaper.pdf.

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: Litepaper: https://dfyn.network/assets/docs/Dfyn-Litepaper.pdf. Gitbooks:
https://docs.dfyn.network/.

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

> ⚠ Answer: No

No basic software function documentation was found in their Litepaper, GitHub, or GitBooks.

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%

No software documentation was found in their Litepaper, GitHub, or GitBooks.

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: 0%

Code examples are in the Appendix. As per the SLOC, there is 6%% 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%

No software documentation was founds, and therefore code implementation traceability is non-existent.

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 85% 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 evidence of code coverage was found in their audit 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 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

 Instructions to run tests can be found at the bottom of https://github.com/dfyn/dfyn-exchange.

14) Report of the results (%)

⚠ Answer: 0%

No test report was found in their GitHub repository.

Guidance:
100%   Detailed test report as described below
70%   GitHub Code coverage report visible
0%   No test report evident

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

15) Formal Verification test done (%)

> ⚠ Answer: 0%

No evidence of a DFYN formal verification was found in their documentation or on the web.

16) Stress Testing environment (%)

> ✓ Answer: 100%

There is evidence of DFYN using the Router testnet network at https://docs.dfyn.network/current-status#routers-cross-chain-liquidity-protocol-xclp.

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: 90%

QuillAudits did a DFYN audit in May 2021.

Note: Audit does not state that any of the proposed fixes were implemented.

Note 2: DFYN's Phase 1 farming release date could not be found anywhere, but we can assume it's before May as that is when they released Phase 2.

Note 3: DFYN was released in December 2020.

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: 0%

There is a "report bug" function in their "Get Involved" section on their website that is referred to as a bug bounty. However, the link leads to an email you send to them and there is no other available bug bounty information in any of their documentation.

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: 0%

No admin control information was found in their Litepaper or GitBooks.

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%

No information at all, other than the fact that users can vote on contract upgrades. However, this is not a clear statement that the DFYN contracts are upgradeable.

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: 0%

No admin control info was found in any of their 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 (%)

No pause control info or tests were found in any of their documentation.

Guidance:

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

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

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

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

0%   Pause control not documented or explained

How to improve this score

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

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

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

I was coaxed to go to EthDenver 2018 and there I started SecuEth.org with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got EthFoundation funding to assist in their development.

Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix

PQ Audit Scoring Matrix (v0.7)	Total Points	DYFN Answer	Points
Total	260		145.5
Code and Team			**56%**
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	0%	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%) cover the deployed contracts? (%)	15	0%	0
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	5	0%	0
10) Is it possible to trace from software documentation to the implementation in code (%)	10	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
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	100%	5
Security			
17) Did 3rd Party audits take place? (%)	70	90%	63
18) Is the bug bounty acceptable high? (%)	10	0%	0
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	5	0%	0
20) Is the information clear and complete	10	0%	0
21) Is the information in non-technical terms	10	0%	0
22) Is there Pause Control documentation including records of tests	10	0%	0

Section Scoring

Code and Team	50	90%
Documentation	45	11%
Testing	50	65%
Security	80	79%
Access Controls	35	0%

Executing Code Appendix

Dfyn

(Dfyn)swapV2Factory

0xE7Fb3e833eFE5F9c441105EB65Ef8b261266423B

(Dfyn)swapV2Router02

0xA102072A4C07F06EC3B4900FDC4C7B80b6c57429

> ⓘ **(Dfyn)swapV2Pair Code Hash**
> f187ed688403aa4f7acfada758d8d53698753b998a3071b06f1b777f4330eaf3

Dfyn Farms

DAI/USDT Pool

0x84D0640Cd8c366BcA7Abc3492fa3CA99C8e32615

ETHER/USDC Pool

0x28Bf5111B86D41427c02DFB9E98E55E5BB57d692

Code Used Appendix

Token Transfers for 0xA102072A4C07F06EC3B4900FDC4C7B80b6c57429

Example Code Appendix

```
1  contract UniswapV2Router02 is IUniswapV2Router02 {
2      using SafeMathUniswap for uint;
3
4      address public immutable override factory;
5      address public immutable override WETH;
6
7      modifier ensure(uint deadline) {
8          require(deadline >= block.timestamp, 'UniswapV2Router: EXPIRED');
9          _;
10     }
11
12     constructor(address _factory, address _WETH) public {
13         factory = _factory;
14         WETH = _WETH;
15     }
16
17     receive() external payable {
18         assert(msg.sender == WETH); // only accept ETH via fallback from the WETH contract
19
```

```solidity
        }

    // **** ADD LIQUIDITY ****
    function _addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin
    ) internal virtual returns (uint amountA, uint amountB) {
        // create the pair if it doesn't exist yet
        if (IUniswapV2Factory(factory).getPair(tokenA, tokenB) == address(0)) {
            IUniswapV2Factory(factory).createPair(tokenA, tokenB);
        }
        (uint reserveA, uint reserveB) = UniswapV2Library.getReserves(factory, tokenA, tok
        if (reserveA == 0 && reserveB == 0) {
            (amountA, amountB) = (amountADesired, amountBDesired);
        } else {
            uint amountBOptimal = UniswapV2Library.quote(amountADesired, reserveA, reservel
            if (amountBOptimal <= amountBDesired) {
                require(amountBOptimal >= amountBMin, 'UniswapV2Router: INSUFFICIENT_B_AMOU
                (amountA, amountB) = (amountADesired, amountBOptimal);
            } else {
                uint amountAOptimal = UniswapV2Library.quote(amountBDesired, reserveB, rese
                assert(amountAOptimal <= amountADesired);
                require(amountAOptimal >= amountAMin, 'UniswapV2Router: INSUFFICIENT_A_AMOU
                (amountA, amountB) = (amountAOptimal, amountBDesired);
            }
        }
    }
    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint amountA, uint amountB, uint
        (amountA, amountB) = _addLiquidity(tokenA, tokenB, amountADesired, amountBDesired,
        address pair = UniswapV2Library.pairFor(factory, tokenA, tokenB);
        TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, amountA);
        TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, amountB);
        liquidity = IUniswapV2Pair(pair).mint(to);
    }
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
```

```solidity
72
        uint deadline
73    ) external virtual override payable ensure(deadline) returns (uint amountToken, uint am
74        (amountToken, amountETH) = _addLiquidity(
75            token,
76            WETH,
77            amountTokenDesired,
78            msg.value,
79            amountTokenMin,
80            amountETHMin
81        );
82        address pair = UniswapV2Library.pairFor(factory, token, WETH);
83        TransferHelper.safeTransferFrom(token, msg.sender, pair, amountToken);
84        IWETH(WETH).deposit{value: amountETH}();
85        assert(IWETH(WETH).transfer(pair, amountETH));
86        liquidity = IUniswapV2Pair(pair).mint(to);
87        // refund dust eth, if any
88        if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - a
89    }
90
91    // **** REMOVE LIQUIDITY ****
92    function removeLiquidity(
93        address tokenA,
94        address tokenB,
95        uint liquidity,
96        uint amountAMin,
97        uint amountBMin,
98        address to,
99        uint deadline
100    ) public virtual override ensure(deadline) returns (uint amountA, uint amountB) {
101        address pair = UniswapV2Library.pairFor(factory, tokenA, tokenB);
102        IUniswapV2Pair(pair).transferFrom(msg.sender, pair, liquidity); // send liquidity
103        (uint amount0, uint amount1) = IUniswapV2Pair(pair).burn(to);
104        (address token0,) = UniswapV2Library.sortTokens(tokenA, tokenB);
105        (amountA, amountB) = tokenA == token0 ? (amount0, amount1) : (amount1, amount0);
106        require(amountA >= amountAMin, 'UniswapV2Router: INSUFFICIENT_A_AMOUNT');
107        require(amountB >= amountBMin, 'UniswapV2Router: INSUFFICIENT_B_AMOUNT');
108    }
109    function removeLiquidityETH(
110        address token,
111        uint liquidity,
112        uint amountTokenMin,
113        uint amountETHMin,
114        address to,
115        uint deadline
116    ) public virtual override ensure(deadline) returns (uint amountToken, uint amountETH)
117        (amountToken, amountETH) = removeLiquidity(
118            token,
119            WETH,
120            liquidity,
121            amountTokenMin,
122            amountETHMin,
123            address(this),
124            deadline
```

```solidity
124            deadline
125
        );
126        TransferHelper.safeTransfer(token, to, amountToken);
127        IWETH(WETH).withdraw(amountETH);
128        TransferHelper.safeTransferETH(to, amountETH);
129    }
130    function removeLiquidityWithPermit(
131        address tokenA,
132        address tokenB,
133        uint liquidity,
134        uint amountAMin,
135        uint amountBMin,
136        address to,
137        uint deadline,
138        bool approveMax, uint8 v, bytes32 r, bytes32 s
139    ) external virtual override returns (uint amountA, uint amountB) {
140        address pair = UniswapV2Library.pairFor(factory, tokenA, tokenB);
141        uint value = approveMax ? uint(-1) : liquidity;
142        IUniswapV2Pair(pair).permit(msg.sender, address(this), value, deadline, v, r, s);
143        (amountA, amountB) = removeLiquidity(tokenA, tokenB, liquidity, amountAMin, amountB
144    }
145    function removeLiquidityETHWithPermit(
146        address token,
147        uint liquidity,
148        uint amountTokenMin,
149        uint amountETHMin,
150        address to,
151        uint deadline,
152        bool approveMax, uint8 v, bytes32 r, bytes32 s
153    ) external virtual override returns (uint amountToken, uint amountETH) {
154        address pair = UniswapV2Library.pairFor(factory, token, WETH);
155        uint value = approveMax ? uint(-1) : liquidity;
156        IUniswapV2Pair(pair).permit(msg.sender, address(this), value, deadline, v, r, s);
157        (amountToken, amountETH) = removeLiquidityETH(token, liquidity, amountTokenMin, amo
158    }
159
160    // **** REMOVE LIQUIDITY (supporting fee-on-transfer tokens) ****
161    function removeLiquidityETHSupportingFeeOnTransferTokens(
162        address token,
163        uint liquidity,
164        uint amountTokenMin,
165        uint amountETHMin,
166        address to,
167        uint deadline
168    ) public virtual override ensure(deadline) returns (uint amountETH) {
169        (, amountETH) = removeLiquidity(
170            token,
171            WETH,
172            liquidity,
173            amountTokenMin,
174            amountETHMin,
175            address(this),
176            deadline
```

```
177          );
178          TransferHelper.safeTransfer(token, to, IERC20Uniswap(token).balanceOf(address(this
179          IWETH(WETH).withdraw(amountETH);
180          TransferHelper.safeTransferETH(to, amountETH);
181      }
182      function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
183          address token,
184          uint liquidity,
185          uint amountTokenMin,
186          uint amountETHMin,
187          address to,
188          uint deadline,
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	17	1260	127	59	1074	106

Comments to Code 59/1074 = 6%%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	6	1082	153	13	916	22

Tests to Code 916/1074 = 85%