0.7

Olympus Finance Process Quality Review

Score: 68%

Overview

This is a Olympus Finance Process Quality Review completed on July 26th 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 **68%**, 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:** Ethereum

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche

Code and Team

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

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

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

> **Answer:** 100%

They are available at website https://docs.olympusdao.finance/references/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 labeling not clear or easy to find
0% Executing addresses could not be found

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

> **Answer:** 100%

Activity is 300 transactions a day on contract *OlympusStaking.sol*, as indicated in the Appendix.

Guidance:

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

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

> **Answer:** Yes

GitHub: https://github.com/OlympusDAO/olympus-contracts.

Is there a public software repository with the code at a minimum, but also normally test and scripts. Even if the repository was created just to hold the files and has just 1 transaction, it gets a **"Yes"**. For teams with private repositories, this answer is **"No"**.

4) Is there a development history visible? (%)

> **Answer:** 100%

With 157 commits and 25 branches, this is a robust software repository.

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

Guidance:

100% Any one of 100+ commits, 10+branches
70% Any one of 70+ commits, 7+branches
50% Any one of 50+ commits, 5+branches

30% Any one of 30+ commits, 3+branches
0% Less than 2 branches or less than 30 commits

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

> ✓ **Answer:** Yes

The OlympusDAO team is public at https://www.linkedin.com/company/olympusdao.

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, then this question is a **"No"**.

Documentation

This section looks at the software documentation. The document explaining these questions is here.

Required questions are;

6) Is there a whitepaper? (Y/N)
7) Are the basic software functions documented? (Y/N)
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)
10) Is it possible to trace from software documentation to the implementation in code (%)

6) Is there a whitepaper? (Y/N)

> ✓ **Answer:** Yes

Location: https://docs.olympusdao.finance/.

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

> ⚠ **Answer:** No

There is no software function documentation in any of the Olympus Finance 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 software function documentation in any of the Olympus Finance 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:** 39%

Code examples are in the Appendix. As per the SLOC, there is 57% 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 software function documentation in any of the Olympus Finance 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 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:** 80%

Code examples are in the Appendix. As per the SLOC, there is 109% 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 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:** 50%

There is no evidence of code coverage in any of the Olympus Finance documentation or in their published audit reports. However, they did do a reasonable amount 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 improved by adding tests that achieve full code coverage. A clear report and scripts in the software repository will guarantee a high score.

13) Scripts and instructions to run the tests (Y/N)

> ✓ **Answer:** Yes

Location: https://github.com/OlympusDAO/olympus-contracts/tree/V3_migration/scripts.

14) Report of the results (%)

> ⚠ **Answer:** 0%

There is no evidence of any Olympus Finance test report or results in any of their documentation or GitHub repositories.

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 an Olympus Finance Formal Verification test.

16) Stress Testing environment (%)

> ✓ **Answer:** 100%

There is evidence of Olympus Finance test-net smart contract usage in the README.md at https://github.com/OlympusDAO/olympus-contracts/tree/V3_migration.

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%

These are two high quality audits where recommendations were implemented.

PeckShield published a Olympus audit report on April 9th 2021.

Omniscia published a Olympus audit report on May 28th 2021.

Most fix recommendations from both audit reports were successfully implemented by the Olympus team.

Olympus Finance was launched in early March 2021.

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

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%

There is currently no Olympus Finance Bug Bounty program.

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%

No official "Governance" section in the Olympus Finance documentation. There are brief excerpts of DAO governance and voting but the real information was found in one of their Medium articles called "The Genesis DAO".

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) Clearly labelled as upgradeable through voting on protocol implementations.

b) Defined DAO community voting roles

Source: https://olympusdao.medium.com/the-genesis-dao-70f0ee6b5b8.

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:** 90%

All access control-related information is written in user-friendly language.

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:** 40%

Pause control is mentioned in https://olympusdao.medium.com/the-genesis-dao-70f0ee6b5b8, but there are details on the capabilities or test results in their GitHub repository.

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	Olympus Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		176.45
Code and Team			**68%**
1) Are the executing code addresses readily available? (%)	**20**	100%	20
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	Y	5
4) Is there a development history visible? (%)	**5**	100%	5
5) Is the team public (not anonymous)? (Y/N)	**15**	y	15
Code Documentation			
6) Is there a whitepaper? (Y/N)	**5**	Y	5
7) Are the basic software functions documented? (Y/N)	**10**	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**	39%	1.95
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**	100%	5
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
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**	40%	2
20) Is the information clear and complete	**10**	60%	6
21) Is the information in non-technical terms	**10**	90%	9
22) Is there Pause Control documentation including records of tests	**10**	40%	4
Section Scoring			
Code and Team	50	100%	
Documentation	45	15%	
Testing	50	57%	
Security	80	88%	
Access Controls	35	60%	

Executing Code Appendix

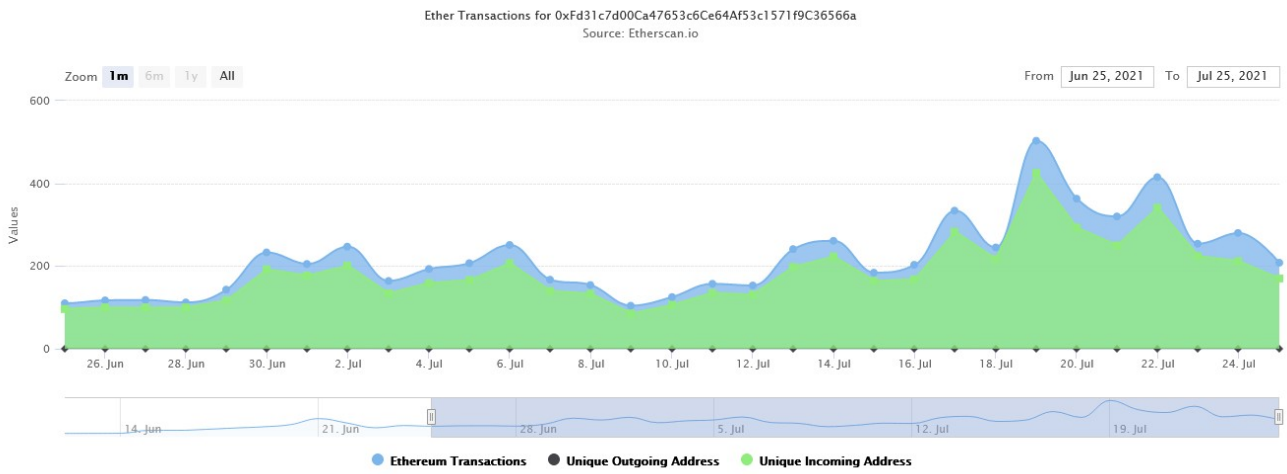
Staking

- new https://etherscan.io/address/0xFd31c7d00Ca47653c6Ce64Af53c1571f9C36566a
- helper https://etherscan.io/address/0xC8C436271f9A6F10a5B80c8b8eD7D0E8f37a612d
- old https://etherscan.io/address/0x0822F3C03dcc24d200AFF33493Dc08d0e1f274A2

LP

- V1 0x34d7...ef7c

Code Used Appendix



Example Code Appendix

```
contract OlympusStaking is Ownable {

    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    address public immutable OHM;
    address public immutable sOHM;

    struct Epoch {
        uint length;
        uint number;
        uint endBlock;
        uint distribute;
    }
    Epoch public epoch;
```

```
16
17      address public distributor;
18
19      address public locker;
20      uint public totalBonus;
21
22      address public warmupContract;
23      uint public warmupPeriod;
24
25      constructor (
26          address _OHM,
27          address _sOHM,
28          uint _epochLength,
29          uint _firstEpochNumber,
30          uint _firstEpochBlock
31      ) {
32          require( _OHM != address(0) );
33          OHM = _OHM;
34          require( _sOHM != address(0) );
35          sOHM = _sOHM;
36
37          epoch = Epoch({
38              length: _epochLength,
39              number: _firstEpochNumber,
40              endBlock: _firstEpochBlock,
41              distribute: 0
42          });
43      }
44
45      struct Claim {
46          uint deposit;
47          uint gons;
48          uint expiry;
49          bool lock; // prevents malicious delays
50      }
51      mapping( address => Claim ) public warmupInfo;
52
53      /**
54          @notice stake OHM to enter warmup
55          @param _amount uint
56          @return bool
57       */
58      function stake( uint _amount, address _recipient ) external returns ( bool ) {
59          rebase();
60
61          IERC20( OHM ).safeTransferFrom( msg.sender, address(this), _amount );
62
63          Claim memory info = warmupInfo[ _recipient ];
64          require( !info.lock, "Deposits for account are locked" );
65
66          warmupInfo[ _recipient ] = Claim ({
67              deposit: info.deposit.add( _amount ),
68              gons: info.gons.add( IsOHM( sOHM ).gonsForBalance( _amount ) ),
```

```
            expiry: epoch.number.add( warmupPeriod ),
            lock: false
        });

        IERC20( sOHM ).safeTransfer( warmupContract, _amount );
        return true;
    }

    /**
        @notice retrieve sOHM from warmup
        @param _recipient address
     */
    function claim ( address _recipient ) public {
        Claim memory info = warmupInfo[ _recipient ];
        if ( epoch.number >= info.expiry && info.expiry != 0 ) {
            delete warmupInfo[ _recipient ];
            IWarmup( warmupContract ).retrieve( _recipient, IsOHM( sOHM ).balanceForGons(
        }
    }

    /**
        @notice forfeit sOHM in warmup and retrieve OHM
     */
    function forfeit() external {
        Claim memory info = warmupInfo[ msg.sender ];
        delete warmupInfo[ msg.sender ];

        IWarmup( warmupContract ).retrieve( address(this), IsOHM( sOHM ).balanceForGons( i
        IERC20( OHM ).safeTransfer( msg.sender, info.deposit );
    }

    /**
        @notice prevent new deposits to address (protection from malicious activity)
     */
    function toggleDepositLock() external {
        warmupInfo[ msg.sender ].lock = !warmupInfo[ msg.sender ].lock;
    }

    /**
        @notice redeem sOHM for OHM
        @param _amount uint
        @param _trigger bool
     */
    function unstake( uint _amount, bool _trigger ) external {
        if ( _trigger ) {
            rebase();
        }
        IERC20( sOHM ).safeTransferFrom( msg.sender, address(this), _amount );
        IERC20( OHM ).safeTransfer( msg.sender, _amount );
    }

    /**
        @notice returns the sOHM index, which tracks rebase growth
```

```
122         @return uint
123      */
124     function index() public view returns ( uint ) {
125         return IsOHM( sOHM ).index();
126     }
127 
128     /**
129         @notice trigger rebase if epoch over
130      */
131     function rebase() public {
132         if( epoch.endBlock <= block.number ) {
133 
134             IsOHM( sOHM ).rebase( epoch.distribute, epoch.number );
135 
136             epoch.endBlock = epoch.endBlock.add( epoch.length );
137             epoch.number++;
138 
139             if ( distributor != address(0) ) {
140                 IDistributor( distributor ).distribute();
141             }
142 
143             uint balance = contractBalance();
144             uint staked = IsOHM( sOHM ).circulatingSupply();
145 
146             if( balance <= staked ) {
147                 epoch.distribute = 0;
148             } else {
149                 epoch.distribute = balance.sub( staked );
150             }
151         }
152     }
153 
154     /**
155         @notice returns contract OHM holdings, including bonuses provided
156         @return uint
157      */
158     function contractBalance() public view returns ( uint ) {
159         return IERC20( OHM ).balanceOf( address(this) ).add( totalBonus );
160     }
161 
162     /**
163         @notice provide bonus to locked staking contract
164         @param _amount uint
165      */
166     function giveLockBonus( uint _amount ) external {
167         require( msg.sender == locker );
168         totalBonus = totalBonus.add( _amount );
169         IERC20( sOHM ).safeTransfer( locker, _amount );
170     }
171 
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	16	3485	711	886	2248	365

Comments to Code 886/2248 = 57%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	11	2073	306	240	1527	57
TypeScript	4	1148	218	5	925	0

Tests to Code 2451/2248 = 109%