0.7

Kyber Network 3.0 (0.7) Process Quality Review

76%

Overview

This is a Kyber Network Process Quality Review completed on 15 Sep 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by of DeFiSafety. Check out our Telegram.

The final score of the review is **76%**, a **PASS**. The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is **70%.**

Summary of the Process

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

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

Disclaimer

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 blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

> **Chain:** Ethereum, Binance Smart Chain, Polygon, Avalanche

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:** 100%

They are available at website https://developer.kyber.network/docs/Addresses-Mainnet/, 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 200+ transactions a day on contract 0xdd974d5c2e2928dea5f71b9825b8b646686bd200 , 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/dynamic-amm

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 300 commits and 10 branches, a rich development history is visible.

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

Location: https://files.kyber.network/DMM-Feb21.pdf

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://files.kyber.network/DMM-Feb21.pdf

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

> **Answer:** Yes

The protocol details its functions in an overview page.

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:** 60%

Some contracts are covered by documentation, though many contracts deployed are not covered.

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%

Code examples are in the Appendix. As per the SLOC, there is 12% 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:** 60%

Code is dissected in the software documentation though the traceability between the two is nonexplicit.

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:** 100%

Code examples are in the Appendix. As per the SLOC, there is 216% 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

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

> **Answer:** 50%

There is no indication of code coverage, but a 216% TtC ratio indicates good testing.

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

Scrips/Instructions location: https://github.com/KyberNetwork/smart-contracts

14) Report of the results (%)

> **Answer:** 0%

There is no test report.

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 proof of formal verification could be found.

16) Stress Testing environment (%)

> ✓ **Answer:** 100%

Kyber has been deployed to multiple testnets.

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%

A single audit has been completed this year; the results are public.

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, (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%

Aside from temporary bug bounties offered in 2019 at hackathons, no bug bounty program has been found.

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%

Access control information is present, but not well labelled.

Guidance:

100% Clearly labelled and on website, docs or repo, quick to find
70% Clearly labelled and on website, docs or repo but takes a bit of looking
40% Access control docs in multiple places and not well labelled

20% Access control docs in multiple places and not labelled
0% Admin Control information could not be found

20) Is the information clear and complete (%)

> **Answer:** 90%

a) All contracts are clearly labelled as upgradeable (or not) -- 30% -- the contracts are clearly explained as modifiable.

b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% --- three different ownership groups are clearly identified

c) The capabilities for change in the contracts are described -- 30% -- the capacity for change is explained well using both diagrams and text.

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%

The documentation details the information in technical 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:** 80%

Pause controls are mentioned with details, but there is no documented testing.

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.

PQ Audit Scoring Matrix (v0.7)	Total Points	Kyber 3.0 UPDATE Answer	Points
Total	260		197.5
Code and Team			**76%**
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	Y	10
8) Does the software function documentation fully (100%) cov	15	60%	9
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	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
12) Code coverage (Covers all the deployed lines of code, or ex	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	5	40%	2
20) Is the information clear and complete	10	90%	9
21) Is the information in non-technical terms	10	30%	3
22) Is there Pause Control documentation including records of	10	80%	8
Section Scoring			
Code and Team	50	100%	
Documentation	45	67%	
Testing	50	65%	
Security	80	79%	

Access Controls	35	63%	

Executing Code Appendix

Contract Addresses

KyberNetworkProxy

New: `0x9AAb3f75489902f3a48495025729a0AF77d4b11e`
Old: `0x818E6FECD516Ecc3849DAf6845e3EC868087B755`

KyberStorage

`0xC8fb12402cB16970F3C5F4b48Ff68Eb9D1289301`

KyberHintHandler (KyberMatchingEngine)

`0xa1C0Fa73c39CFBcC11ec9Eb1Afc665aba9996E2C`

KyberFeeHandler (ETH)

`0xd3d2b5643e506c6d9B7099E9116D7aAa941114fe`

KyberNetwork

`0x7C66550C9c730B6fdd4C03bc2e73c5462c5F7ACC`

KyberStaking

`0xECf0bdB7B3F349AbfD68C3563678124c5e8aaea3`

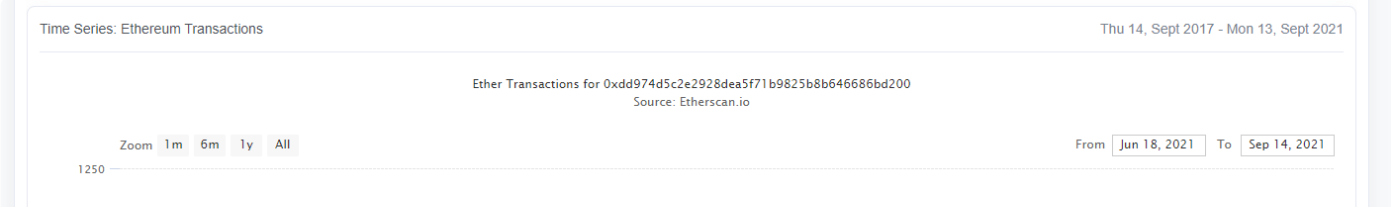
KyberDao

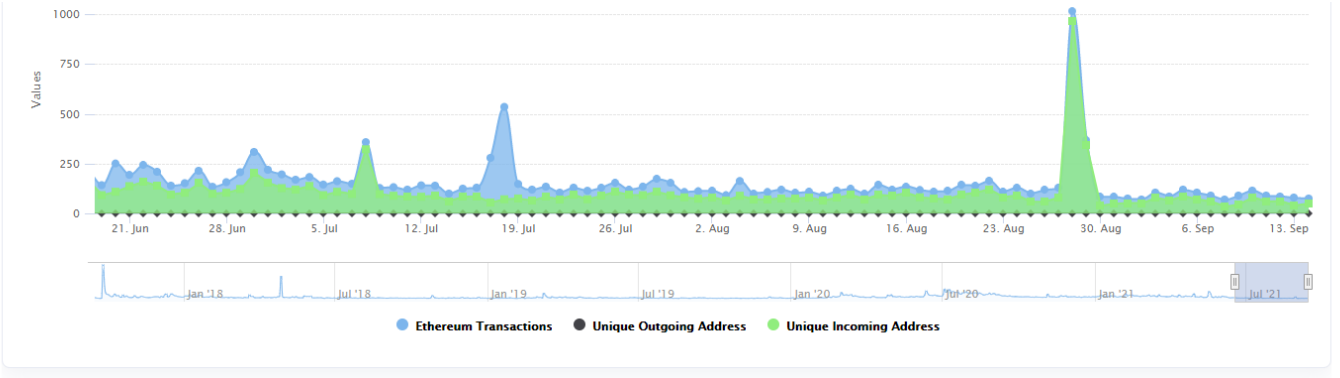
`0x49bdd8854481005bBa4aCEbaBF6e06cD5F6312e9`

KyberReserve

`0x63825c174ab367968EC60f061753D3bbD36A0D8F`

Code Used Appendix





Example Code Appendix

```
contract DMMFactory is IDMMFactory {
    using EnumerableSet for EnumerableSet.AddressSet;

    uint256 internal constant BPS = 10000;

    address private feeTo;
    uint16 private governmentFeeBps;
    address public override feeToSetter;

    mapping(IERC20 => mapping(IERC20 => EnumerableSet.AddressSet)) internal tokenPools;
    mapping(IERC20 => mapping(IERC20 => address)) public override getUnamplifiedPool;
    address[] public override allPools;

    event PoolCreated(
        IERC20 indexed token0,
        IERC20 indexed token1,
        address pool,
        uint32 ampBps,
        uint256 totalPool
    );
    event SetFeeConfiguration(address feeTo, uint16 governmentFeeBps);
    event SetFeeToSetter(address feeToSetter);

    constructor(address _feeToSetter) public {
        feeToSetter = _feeToSetter;
    }

    function createPool(
        IERC20 tokenA,
        IERC20 tokenB,
        uint32 ampBps
    ) external override returns (address pool) {
        require(tokenA != tokenB, "DMM: IDENTICAL_ADDRESSES");
        (IERC20 token0, IERC20 token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, toke
        require(address(token0) != address(0), "DMM: ZERO_ADDRESS");
        require(ampBps >= BPS, "DMM: INVALID_BPS");
        // only exist 1 unamplified pool of a pool.
        require(
            ampBps != BPS || getUnamplifiedPool[token0][token1] == address(0),
```

```
            "DMM: UNAMPLIFIED_POOL_EXISTS"
        );
        pool = address(new DMMPool());
        DMMPool(pool).initialize(token0, token1, ampBps);
        // populate mapping in the reverse direction
        tokenPools[token0][token1].add(pool);
        tokenPools[token1][token0].add(pool);
        if (ampBps == BPS) {
            getUnamplifiedPool[token0][token1] = pool;
            getUnamplifiedPool[token1][token0] = pool;
        }
        allPools.push(pool);

        emit PoolCreated(token0, token1, pool, ampBps, allPools.length);
    }

    function setFeeConfiguration(address _feeTo, uint16 _governmentFeeBps) external overri
        require(msg.sender == feeToSetter, "DMM: FORBIDDEN");
        require(_governmentFeeBps > 0 && _governmentFeeBps < 2000, "DMM: INVALID FEE");
        feeTo = _feeTo;
        governmentFeeBps = _governmentFeeBps;

        emit SetFeeConfiguration(_feeTo, _governmentFeeBps);
    }

    function setFeeToSetter(address _feeToSetter) external override {
        require(msg.sender == feeToSetter, "DMM: FORBIDDEN");
        feeToSetter = _feeToSetter;

        emit SetFeeToSetter(_feeToSetter);
    }

    function getFeeConfiguration()
        external
        override
        view
        returns (address _feeTo, uint16 _governmentFeeBps)
    {
        _feeTo = feeTo;
        _governmentFeeBps = governmentFeeBps;
    }

    function allPoolsLength() external override view returns (uint256) {
        return allPools.length;
    }

    function getPools(IERC20 token0, IERC20 token1)
        external
        override
        view
        returns (address[] memory _tokenPools)
    {
        uint256 length = tokenPools[token0][token1].length();
```

```
93          _tokenPools = new address[](length);
94          for (uint256 i = 0; i < length; i++) {
95              _tokenPools[i] = tokenPools[token0][token1].at(i);
96          }
97      }
98
99      function getPoolsLength(IERC20 token0, IERC20 token1) external view returns (uint256)
100         return tokenPools[token0][token1].length();
101     }
102
103     function getPoolAtIndex(
104         IERC20 token0,
105         IERC20 token1,
106         uint256 index
107     ) external view returns (address pool) {
108         return tokenPools[token0][token1].at(index);
109     }
110
111     function isPool(
112         IERC20 token0,
113         IERC20 token1,
114         address pool
115     ) external override view returns (bool) {
116         return tokenPools[token0][token1].contains(pool);
117     }
118 }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	11	2220	199	214	1807	153

Comments to Code 214/1807 = 12%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	15	4699	569	191	3912	91

Tests to Code 3912/1807 = 216%