Kyber Process Quality Review

Score 89%

This is a Process Quality Review on Kyber Network completed on 23 November, 2020. It was performed using the Process Review process (version 0.6) and is documented here. The review was performed by ShinkaRex of Caliburn Consulting. Check out our Telegram.

The final score of the review is 89%, a excellent score. The breakdown of the scoring is in Scoring Appendix.

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**

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.

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? (Y/N)
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)

Are the executing code addresses readily available? (Y/N)

> Answer: Yes

They are available at website https://developer.kyber.network/docs/Addresses-

Mainnet/ as indicated in the Appendix.

How to improve this score

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

Is the code actively being used? (%)

> Answer: 100%

Activity is *438 transactions a day on contract Kybernetworkproxy.sol*, as indicated in the Appendix.

Percentage Score Guidance

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

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

> Answer: Yes

GitHub: https://github.com/KyberNetwork

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.

How to improve this score

Maintain a public repo, at least for deployed code. Public repo's are in line with the vision of Ethereum where development is shared and public.

Is there a development history visible? (%)

> Answer: 100%

With 20 branches and 2009 commits, this is a healthy 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.

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

> Answer: Yes

The team was found here: https://icobench.com/ico/kybernetwork/team

There does not appear to be a direct team listing on their website.

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 seems a No.

Documentation

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

Required questions are;

1. Is there a whitepaper? (Y/N)
2. Are the basic software functions documented? (Y/N)
3. Does the software function documentation fully (100%) cover the deployed contracts? (%)
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)
5. Is it possible to trace from software documentation to the implementation in codee (%)

Is there a whitepaper? (Y/N)

> Answer: Yes

Location: https://files.kyber.network/Kyber_Protocol_22_April_v0.1.pdf

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

Answer: Yes

Location: https://developer.kyber.network/docs/API_ABI-Intro/

There is extensive documentation about the software functions on their API-ABI documentation.

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

Answer: 100%

There is extensive documentation outlining all the functions on their ABI-API documentation. This is through and well-written 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

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

Answer: 40%

There is some useful commenting in the code, and their CtC ratio is relatively low. Most of the function explanations are found in the API/ABI.

Code examples are in the Appendix. As per the SLOC, there is 17% 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.

Is it possible to trace from software documentation to the implementation in code (%)

> Answer: 60%

Although there is clear documentation of the functions within this code, there minimal code snippits in their documentation, leading to non-explicit tracability. Some code snippits can be found in their Restful API documentation.

Guidance:
100% - Clear explicit traceability between code and documentation at a requirement level for all code
60% - Clear association between code and documents via non explicit traceability
40% - Documentation lists all the functions and describes their functions
0% - No connection between documentation and code

How to improve this score

This score can improve by adding traceability from requirements to code such

that it is clear where each requirement is coded. For reference, check the SecurEth guidelines on traceability.

Testing

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

1. Full test suite (Covers all the deployed code) (%)
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)
3. Scripts and instructions to run the tests (Y/N)
4. Packaged with the deployed code (Y/N)
5. Report of the results (%)
6. Formal Verification test done (%)
7. Stress Testing environment (%)

Is there a Full test suite? (%)

> Answer: 100%

There are clearly a full set of tests, with an astounding TtC of 1079%.

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.

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

Answer: 50%

There is no evidence of a coverage report, but there is a clearly reasonable set 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 improve by adding tests achieving full code coverage. A clear report and scripts in the software repository will guarantee a high score.

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

Answer: Yes

Location: https://github.com/KyberNetwork/smart-contracts

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.

Packaged with the deployed code (Y/N)

Answer: Yes

Report of the results (%)

Answer: 0%

There is no evident report of the results.

How to improve this score

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

Formal Verification test done (%)

Answer: 0%

There is no evidence of formal verification testing.

Stress Testing environment (%)

Answer: 100%

Testnet addresses are published on the Kovan, Rinkeby and Ropsten networks.

Audits

> Answer: 100%

Multiple audits were preformed by ChainSecurity on June 29th, 2018, July 9th, 2018, January 9th, 2019, as well as Other audits were preformed by BlockchainLabs.nz, Kyber was made public in July 5th, 2018 and updated in 2020.

Guidance:

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

Appendices

Author Details

The author of this review is Rex of Caliburn Consulting.

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.

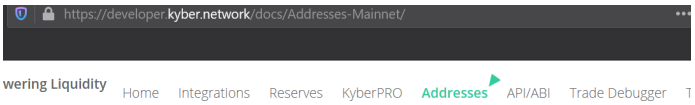
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

Career wise I am a business development manager for an avionics supplier.

Scoring Appendix

	Total	Kyber DEX	
PQ Audit Scoring Matrix (v0.6)	**Points**	**Answer**	**Points**
Total	240		214.5
Code and Team			**89%**
1. Are the executing code addresses readily available? (Y/N)	30	Y	30
2. Is the code actively being used? (%)	10	100%	10
3. Is there a public software repository? (Y/N)	5	Y	5
4. Is there a development history visible? (%)	5	100%	5
Is the team public (not anonymous)? (Y/N)	20	Y	20
Code Documentation			
1. Is there a whitepaper? (Y/N)	5	Y	5
2. Are the basic software functions documented? (Y/N)	10	Y	10
3. Does the software function documentation fully (100%) cover the deployed contracts? (%)	15	100%	15
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)	10	40%	4
5. Is it possible to trace from software documentation to the implementation in code (%)	5	60%	3
Testing			
1. Full test suite (Covers all the deployed code) (%)	20	100%	20
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	50%	2.5
3. Scripts and instructions to run the tests? (Y/N)	5	Y	5
4. Packaged with the deployed code (Y/N)	5	Y	5
5. Report of the results (%)	10	0%	0
6. Formal Verification test done (%)	5	0%	0
7. Stress Testing environment (%)	5	100%	5
Audits			
Audit done	70	100%	70
Section Scoring			
Executing Code Verification	70	100%	
Documentation	45	82%	
Testing	55	68%	
Audits	70	100%	

Executing Code Appendix



KyberNetworkProxy

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

KyberStorage

`0xC8fb12402cB16970F3C5F4b48Ff68Eb9D1289301`

KyberHintHandler (KyberMatchingEngine)

`0xa1C0Fa73c39CFBcC11ec9Eb1Afc665aba9996E2C`

KyberFeeHandler (ETH)

`0xd3d2b5643e506c6d9B7099E9116D7aAa941114fe`

KyberNetwork

`0x7C66550C9c730B6fdd4C03bc2e73c5462c5F7ACC`

KyberStaking

`0xECf0bdB7B3F349AbfD68C3563678124c5e8aaea3`

KyberDao

`0x49bdd8854481005bBa4aCEbaBF6e06cD5F6312e9`

KyberReserve

`0x63825c174ab367968EC60f061753D3bbD36A0D8F`

ConversionRates

`0x798AbDA6Cc246D0EDbA912092A2a3dBd3d11191B`

LiquidityConversionRates

`0x97D7126b6FF7C4D95601912f4Cdf790a3Cd1edaB`

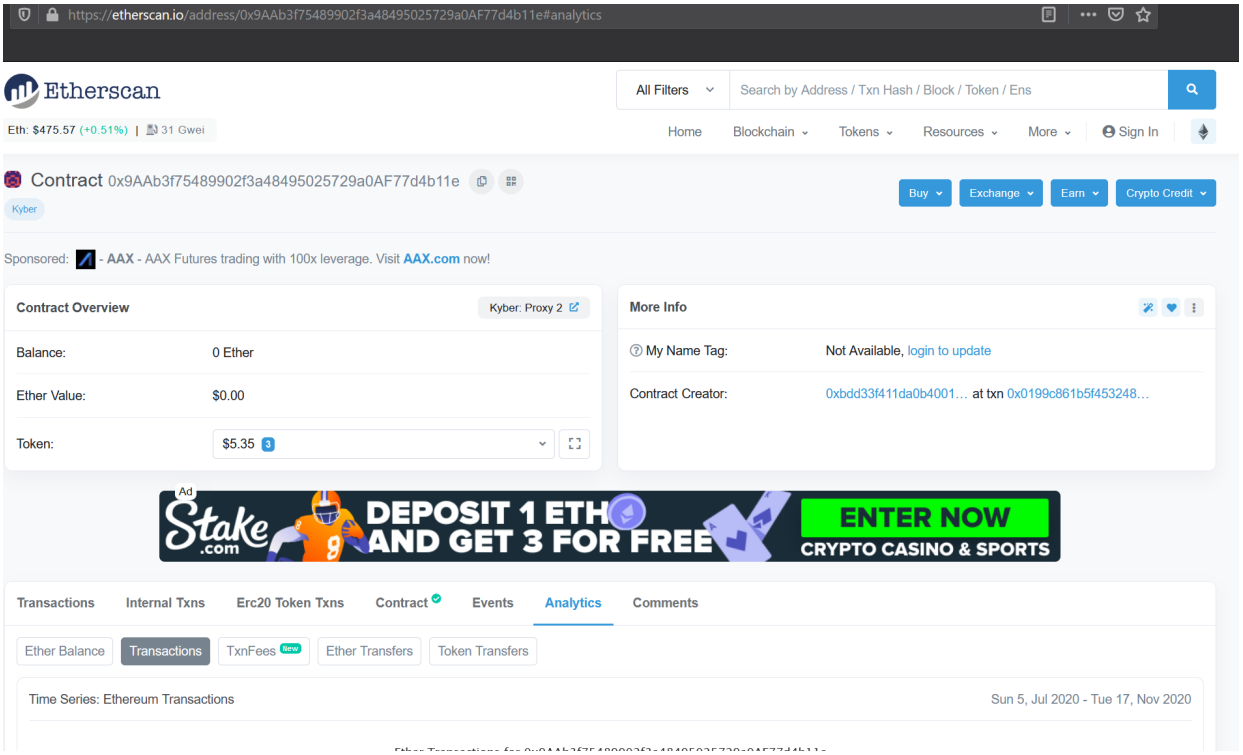
KyberNetworkProxy (V1)

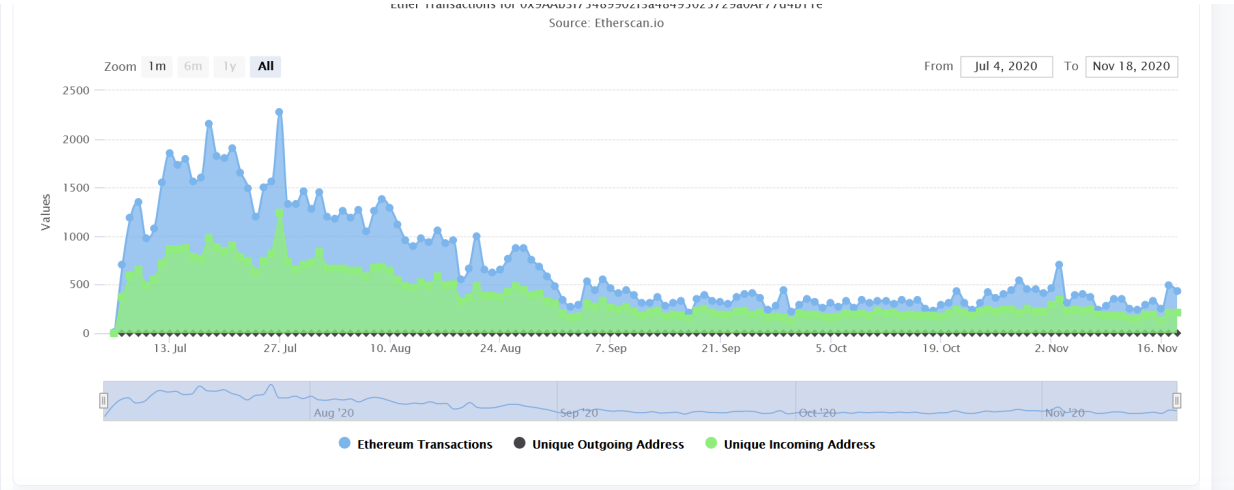
`0x818E6FECD516Ecc3849DAf6845e3EC868087B755`

KyberNetworkENSResolver

`0x1982131C7D6959ff7768EE39c023Ad002d8c9759`

Code Used Appendix





Example Code Appendix

```

    /// @notice Use token address ETH_TOKEN_ADDRESS for ether
    /// @dev Get expected rate for a trade from src to dest tokens, amoun
    /// @param src Source token
    /// @param dest Destination token
    /// @param srcQty Amount of src tokens in twei
    /// @param platformFeeBps Part of the trade that is allocated as fee
    /// @param hint Advanced instructions for running the trade
    /// @return expectedRate for a trade after deducting network + platfo
    ///             Rate = destQty (twei) / srcQty (twei) * 10 ** 18
    function getExpectedRateAfterFee(
        IERC20 src,
        IERC20 dest,
        uint256 srcQty,
        uint256 platformFeeBps,
        bytes calldata hint
    ) external view override returns (uint256 expectedRate) {
        (, expectedRate) = kyberNetwork.getExpectedRateWithHintAndFee(
            src,
            dest,
            srcQty,
            platformFeeBps,
            hint
        );
    }

    function maxGasPrice() external view returns (uint256) {
        return kyberNetwork.maxGasPrice();
    }

    function enabled() external view returns (bool) {
        return kyberNetwork.enabled();
    }
```

```

    /// helper structure for function doTrade
    struct UserBalance {
        uint256 srcTok;
        uint256 destTok;
    }

    function doTrade(
        IERC20 src,
        uint256 srcAmount,
        IERC20 dest,
        address payable destAddress,
        uint256 maxDestAmount,
        uint256 minConversionRate,
        address payable platformWallet,
        uint256 platformFeeBps,
        bytes memory hint
    ) internal returns (uint256) {
        UserBalance memory balanceBefore = prepareTrade(src, dest, srcAmo

        uint256 reportedDestAmount = kyberNetwork.tradeWithHintAndFee{val
            msg.sender,
            src,
            srcAmount,
            dest,
            destAddress,
            maxDestAmount,
            minConversionRate,
            platformWallet,
            platformFeeBps,
            hint
        );
        TradeOutcome memory tradeOutcome = calculateTradeOutcome(
            src,
            dest,
            destAddress,
            platformFeeBps,
            balanceBefore
        );

        require(
            tradeOutcome.userDeltaDestToken == reportedDestAmount,
            "kyberNetwork returned wrong amount"
        );
        require(
            tradeOutcome.userDeltaDestToken <= maxDestAmount,
            "actual dest amount exceeds maxDestAmount"
        );
        require(tradeOutcome.actualRate >= minConversionRate, "rate below

        emit ExecuteTrade(
            msg.sender,
```

```
            src,
            dest,
            destAddress,
            tradeOutcome.userDeltaSrcToken,
            tradeOutcome.userDeltaDestToken,
            platformWallet,
            platformFeeBps
        );

        return tradeOutcome.userDeltaDestToken;
    }

    /// helper structure for function prepareTrade
    struct TradeOutcome {
        uint256 userDeltaSrcToken;
        uint256 userDeltaDestToken;
        uint256 actualRate;
    }

    function prepareTrade(
        IERC20 src,
        IERC20 dest,
        uint256 srcAmount,
        address destAddress
    ) internal returns (UserBalance memory balanceBefore) {
        if (src == ETH_TOKEN_ADDRESS) {
            require(msg.value == srcAmount, "sent eth not equal to srcAmo
        } else {
            require(msg.value == 0, "sent eth not 0");
        }

        balanceBefore.srcTok = getBalance(src, msg.sender);
        balanceBefore.destTok = getBalance(dest, destAddress);

        if (src == ETH_TOKEN_ADDRESS) {
            balanceBefore.srcTok += msg.value;
        } else {
            src.safeTransferFrom(msg.sender, address(kyberNetwork), srcAm
        }
    }

    function calculateTradeOutcome(
        IERC20 src,
        IERC20 dest,
        address destAddress,
        uint256 platformFeeBps,
        UserBalance memory balanceBefore
    ) internal returns (TradeOutcome memory outcome) {
        uint256 srcTokenBalanceAfter;
        uint256 destTokenBalanceAfter;

        srcTokenBalanceAfter = getBalance(src, msg.sender);
```

```
        destTokenBalanceAfter = getBalance(dest, destAddress);

        //protect from underflow
        require(
            destTokenBalanceAfter > balanceBefore.destTok,
            "wrong amount in destination address"
        );
        require(balanceBefore.srcTok > srcTokenBalanceAfter, "wrong amoun

        outcome.userDeltaSrcToken = balanceBefore.srcTok - srcTokenBalanc
        outcome.userDeltaDestToken = destTokenBalanceAfter - balanceBefor

        // what would be the src amount after deducting platformFee
        // not protecting from platform fee
        uint256 srcTokenAmountAfterDeductingFee = (outcome.userDeltaSrcTo
            (BPS - platformFeeBps)) / BPS;

        outcome.actualRate = calcRateFromQty(
            srcTokenAmountAfterDeductingFee,
            outcome.userDeltaDestToken,
            getUpdateDecimals(src),
            getUpdateDecimals(dest)
        );
    }
}
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	68	8559	1268	1076	6215	732

Comments to Code 1076/ 6215 = 17%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	37	35429	5212	1993	28224	2080

Tests to Code 28224 / 6215 = 1079%