dHEDGE Process Quality Review

Score: 70%

This is a Process Quality Review of dHedge completed on April 23/2021. It was performed using the Process Review process (version 0.6.2) and is documented here. The review was performed by Lucas of DeFiSafety. Check out our Telegram.

The final score of the review is 70%, a bare 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**

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://docs.dhedge.org/dhedge-protocol/contract-details as indicated in the Appendix.

Is the code actively being used? (%)

Answer: 40%

Activity is 2 transactions a day on contract adminupgradabilityproxy.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/dhedge/dhedge-token-solidity

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.

Is there a development history visible? (%)

Answer: 30%

With 27 commits and 6 branches, this is a moderately healthy 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.

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

> Answer: Yes

Some of the team member's names can be found in their medium article titles.

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;

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://docs.dhedge.org/

How to improve this score

Ensure the white paper is available for download from your website or at least the software repository. Ideally update the whitepaper to meet the capabilities of your present application.

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

Answer: No

There is no evidence of any software function documentation.

How to improve this score

Write the document based on the deployed code. For guidance, refer to the SecurEth System Description Document.

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

Answer: 0%

There is no evidence of any software function 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 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.

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

> Answer: 45%

Code examples are in the Appendix. As per the SLOC, there is 25% commenting to code (CtC). A lot of the comments

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

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

> Answer: 0%

There is no evidence of any software function 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%

With a TtC of 181%, there is clearly a complete set of tests present.

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: 88%

The code coverage is included in the audit.

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/dhedge/dhedge-token-solidity

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

How to improve this score

Improving this score requires redeployment of the code, with the tests. This score gives credit to those who test their code before deployment and release them together. If a developer adds tests after deployment they can gain full points for all test elements except this one.

Report of the results (%)

Answer: 0%

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.

Formal Verification test done (%)

> ⚠ Answer: 0%

There is no evidence of formal verification having been done.

Stress Testing environment (%)

> ⚠ Answer: 0%

There is no evidence of any stress-testing having been done on this protocol.

Audits

> ✓ Answer: 90%

Iosiro has done an audit on DHedge completed on the 9th of October, 2020.

dHedge was released Oct 22nd, 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 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

Executing Code Appendix

Contract Details

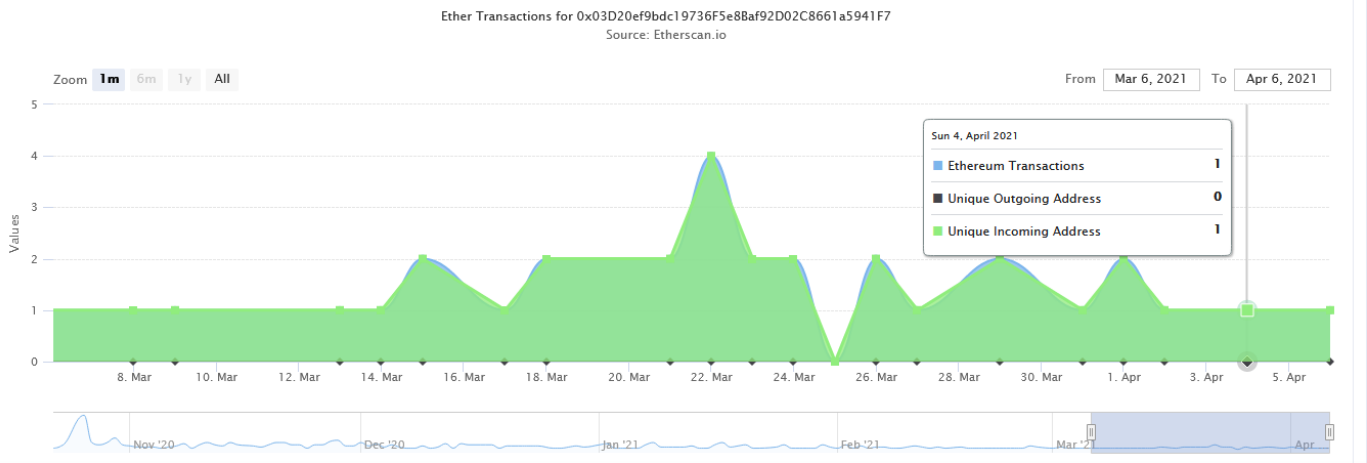
dHEDGE Factory on Ethereum mainnet: 0x03D20ef9bdc19736F5e8Baf92D02C8661a5941F7

dHEDGE Factory on Ethereum Kovan testnet: 0x44226Fc17074A4F019FAA6412E85eff3e01b8368

dHEDGE ABI: https://drive.google.com/file/d/1_ogouiodlesS0bqXgqfTo7qrY4i6-9XQ/view?usp=sharing

dHEDGE Factory ABI: https://drive.google.com/file/d/1NCRm-OsU7EX87yaUoC-uyvL6zlq2kTrI/view?usp=sharing

Code Used Appendix



Example Code Appendix

```
1
2  pragma solidity ^0.6.2;
3
4  import "./ISynthetix.sol";
5  import "./IExchangeRates.sol";
6  import "./IAddressResolver.sol";
7  import "./DHedge.sol";
8  import "./upgradability/ProxyFactory.sol";
9  import "./IHasDaoInfo.sol";
10 import "./IHasFeeInfo.sol";
11 import "./IHasAssetInfo.sol";
12
```

```
13  contract DHedgeFactory is ProxyFactory, IHasDaoInfo, IHasFeeInfo, IHasAsset
14      event FundCreated(
15          address fundAddress,
16          bool isPoolPrivate,
17          string fundName,
18          string managerName,
19          address manager,
20          uint256 time,
21          uint256 managerFeeNumerator,
22          uint256 managerFeeDenominator
23      );
24
25      event DaoAddressSet(address dao);
26      event DaoFeeSet(uint256 numerator, uint256 denominator);
27
28      event ExitFeeSet(uint256 numerator, uint256 denominator);
29      event ExitFeeCooldownSet(uint256 cooldown);
30
31      event MaximumSupportedAssetCountSet(uint256 count);
32
33      IAddressResolver public addressResolver;
34
35      address[] public deployedFunds;
36
37      address internal _daoAddress;
38      uint256 internal _daoFeeNumerator;
39      uint256 internal _daoFeeDenominator;
40
41      mapping (address => bool) public isPool;
42
43      uint256 private _MAXIMUM_MANAGER_FEE_NUMERATOR;
44      uint256 private _MANAGER_FEE_DENOMINATOR;
45      mapping (address => uint256) public poolManagerFeeNumerator;
46      mapping (address => uint256) public poolManagerFeeDenominator;
47
48      uint256 internal _exitFeeNumerator;
49      uint256 internal _exitFeeDenominator;
50      uint256 internal _exitFeeCooldown;
51
52      uint256 internal _maximumSupportedAssetCount;
53
54      bytes32 internal _trackingCode;
55
56      function initialize(IAddressResolver _addressResolver, address _poolLog
57          ProxyFactory.__ProxyFactory_init(_poolLogic);
58
59          addressResolver = _addressResolver;
60
61          _setDaoAddress(daoAddress);
62
63          _setMaximumManagerFee(5000, 10000);
64
65          _setDaoFee(10, 100); // 10%
66          _setExitFee(5, 1000); // 0.5%
67          _setExitFeeCooldown(1 days);
```

```
68
69          _setMaximumSupportedAssetCount(10);
70
71          _setTrackingCode(
72              0x4448454447450000000000000000000000000000000000000000000000000(
73          );
74      }
75
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	12	1469	289	239	941	66

Comments to Code 239/941= 25%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	9	2148	328	115	1705	17

Tests to Code 1705/941=181%