0.7

Defi Saver Process Quality Review

Score: 71%

Overview

This is a Process Quality Review of Defi Saver completed on May 10th 2021. It was performed using the Process Review process (version 0.7) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is 71%, 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 blockchain used by this protocol.

> **Chain: Ethereum**

Guidance:
Ethereum
Binance

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%

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

They are available at https://docs.defisaver.com/protocol/deployed-contracts as indicated in the Appendix.

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

>

Answer: 100%

Activity is below 10 transactions a day on contract MCDMonitorV2.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

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

Answer: Yes

GitHub: https://github.com/DecenterApps/defisaver-v3-contracts/

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: 100%

With over 253 commits and 14 branches across many software repositories, they have an excellent development history.

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

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

Answer: Yes

https://medium.com/defi-saver

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: https://docs.defisaver.com/

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

> ✓ Answer: Yes

Software functions were found at https://docs.defisaver.com/protocol/core

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

> ✓ Answer: 80%

Major functions documented at https://docs.defisaver.com/protocol/core

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 18% 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: 80%

Each contract is explained at https://docs.defisaver.com/protocol/core.

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

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%

With a TtC of 109%, this is a clearly a complete test suite.

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 indication of code coverage, but there is clearly a reasonably complete 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.

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

> ✓ Answer: Yes

Scripts and instructions to run test were found at https://github.com/DecenterApps/defisaver-v3-contracts

14) 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.

15) Formal Verification test done (%)

> ⚠ Answer: 0%

No evidence of a formal verification was found on the web.

16) Stress Testing environment (%)

> ✓ Answer: 100%

Smart Contract Testing Addresses can be found on their Constants Directory.

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: 70%

Defi Saver has had an audit done by Debaub in Feb. 2021, an additional Debaub audit in Mar. 2021, and a Consensys audit in Mar. 2021. All of the audits had multiple significant issues. Many were fixed but given the concern we drop the 100% to 70%.

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
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: 20%

Bug Bounty Location: https://medium.com/defi-saver/defi-saver-bug-bounty-is-now-live-on-immunefi-56af32d0c220

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
50% Bounty is 100k or over
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 admin controls (%)

> Answer: 100%

Access controls are clearly labelled in their Documentation.

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: 35%

a) upgradability is indicated clearly for automation smart contracts, and vaguely for protocol smart contracts: 20%
b) Type of ownership is clearly indicated for Automation smart contracts, but not protocol action contracts: 15%
c) Capabiilities for change in the contracts is not described: 0%

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%

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: 0%

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

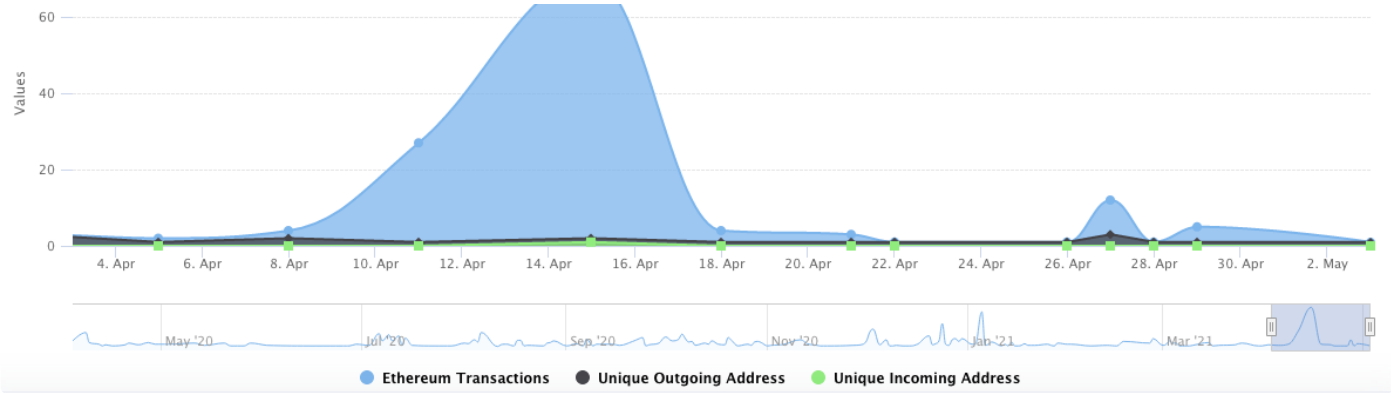
	Total	Defi Saver	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		184
Code and Team			**71%**
1) Are the executing code addresses readily available? (%)	**20**	100%	20
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	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%) cover the deployed contracts? (%)	**15**	80%	12
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**	100%	10
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**	70%	49
18) Is the bug bounty acceptable high? (%)	**10**	20%	2
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	100%	5
20) Is the information clear and complete	**10**	35%	3.5
21) Is the information in non-technical terms	**10**	90%	9
22) Is there Pause Control documentation including records of tests	**10**	0%	0
Section Scoring			
Code and Team	50	100%	
Documentation	45	82%	
Testing	50	57%	
Security	80	64%	
Access Controls	35	50%	

Deployed contracts

Mainnet addresses

```
[x] DFSRegistry - 0xD6049E1F5F3EfF1F921f5532aF1A1632bA23929C
[x] TaskExecutor - 0xb3e5371d55e1e84bfFE7D0b57Bd9c6A4C6b3C635

/********************** Actions ************************/

// utils
[x] WrapEth - 0x9E702937F42Db2cE58342Ca5F213Ef33D51AEF6b
[x] UnwrapEth - 0x925b0C0663aaef2C6F603aD8B992a15adA83f1Cc
[x] PullToken - 0xa8C63267e09F1b4Efc9F934ad81c09e021fc8103
[x] SendToken - 0xBbe0D7f2AF01aE678f8A873CB2d2EB73871C9b5A
[x] SumInputs - 0xCdef35425579fc566B6Ee0499A79256ac88d25cf

// UniswapV2
[x] UniSupply - 0x20051E428c9984B256EB73957cA59c256F01938c
[x] UniWithdraw - 0x5Abf6039fe574F82115742F7FADd0F7A21c2c4dc

// UniswapV3
[x] UniCollectV3 - 0x4D49FB2Cc0DF856b936CCA1816A9e6DD0ADFa232
[x] UniMintV3 - 0x6d61b5c47995DF6DB7e444240ceD3Ea898B2d408
[x] UniSupplyV3 - 0x1398Ea6151e8360Ed0515460214c84FeFf687C6F
[x] UniWithdrawV3 - 0xe5FAB89bdC01130c686B35540531EB1cBbaE8ff3

// DyDx
[x] DyDxSupply - 0xA8D1C1eea86573cBcC919fbf28Db542bDfE7Ed5b
[x] DyDxWithdraw - 0x55AA33F42D79DbD3f1885b410e4796d641549bC3

// Exchange
[x] DFSSell - 0x9A765623F9De2D7dB26afb5f7Bb85592DF094CDB
[x] DFSBuy - 0x939dCad6A3D1fEACccB60Af90876D904468CbF66

// Flash loans
[x] FLAaveV2 - 0xa290BDae65638c083e860b05009f290140fc0379
[x] FLDyDx - 0x505079b4E049B9e641deb7E04D55e9457B8ad8Bc

// AaveV2
[x] AaveBorrow - 0xEdBbF97C505309e5174E164fbec3cAE774d310Ea
[x] AavePayback - 0xA04dd7BbdA2DF9307654d3F91a252F911790773e
[x] AaveSupply - 0xC71113E9122465e8bCd42123f840Df99abaF29F1
[x] AaveWithdraw - 0xE06Fc1CBD78Eb0799d5B0ca62D51B065886e08FC

// Compound
[x] CompBorrow - 0xde1b05266f2D05Bf9216B76500583c2b7785b7e9
[x] CompClaim - 0xAA165C03fF61301b79f35649157f6738263739A3
```

Code Used Appendix



Example Code Appendix

```
pragma solidity ^0.5.0;

import "./interfaces/ERC20.sol";
import "./ReentrancyGuard.sol";
import "./DS/DSMath.sol";
import "./constants/ConstantAddresses.sol";

contract CTokenInterface is ERC20 {
    function mint(uint mintAmount) external returns (uint);
    function redeem(uint redeemTokens) external returns (uint);
    function redeemUnderlying(uint redeemAmount) external returns (uint);
    function borrow(uint borrowAmount) external returns (uint);
    function repayBorrow(uint repayAmount) external returns (uint);
    function repayBorrowBehalf(address borrower, uint repayAmount) external returns (uint)
    function exchangeRateCurrent() external returns (uint);
}

contract SaverProxy {
    function repay(bytes32 _cup, uint _amount, uint _minAmount) public;
    function boost(bytes32 _cup, uint _amount, uint _minAmount) public;
}

/// @title Contract will hold cDai and use it for users to borrow and return in the sam tx
contract DecenterLending is ReentrancyGuard, DSMath, ConstantAddresses {

    //Kovan
    CTokenInterface public cDai = CTokenInterface(CDAI_ADDRESS);
    ERC20 public Dai = ERC20(COMPOUND_DAI_ADDRESS);

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    address public owner;
    address public feeAddress;
    uint public feeAmount;
    uint public sanityBalance;

```

```
    address public sanityContractAddress;

    constructor(address _owner, uint _feeAmount, address _feeAddress) public {
        owner = _owner;
        feeAmount = _feeAmount;
        feeAddress = _feeAddress;
    }

    function borrow(uint _amountToBorrow, uint _type, bytes32 _cup, uint _amount, uint _mi

        require(cDai.redeemUnderlying(_amountToBorrow) == 0, "Reedem failed");

        uint prevDaiBalance = Dai.balanceOf(address(this));

        //Send money
        Dai.transfer(msg.sender, _amountToBorrow);

        if (_type == 1) {
            SaverProxy(msg.sender).repay(_cup, _amount, _minAmount);
        } else {
            SaverProxy(msg.sender).boost(_cup, _amount, _minAmount);
        }


        uint currentDaiBalance = Dai.balanceOf(address(this));

        // if feeAmount is 0, feeEarned will be 0
        uint feeEarned = _amountToBorrow / feeAmount;

        //Where my money bitch
        require(currentDaiBalance >= add(prevDaiBalance, feeEarned));

        // Transfer the fee earned to the feeAddress
        Dai.transfer(feeAddress, feeEarned);

        require(Dai.balanceOf(sanityContractAddress) >= sanityBalance, "Sanity check agains

        require(cDai.mint(_amountToBorrow) == 0, "Mint failed");
    }


    // ADMIN ONLY

    // Owner can get his money back
    function withdraw(uint _amount, address _tokenAddress) public onlyOwner {
        ERC20(_tokenAddress).transfer(owner, _amount);
    }

    function changeFee(uint _newFee) public onlyOwner {
        feeAmount = _newFee;
    }

    function setSanityAmount(uint _sanityAmount) public onlyOwner {
```

```
92
            sanityBalance = _sanityAmount;
93      }
94
95      function setSanityContractAddress(address _contractAddress) public onlyOwner {
96          sanityContractAddress = _contractAddress;
97      }
98  }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	85	6987	1357	875	4755	348

Comments to Code 875/4755 = 18%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	58	6809	1372	212	5225	186

Tests to Code 5225/4755 = 109%