# **B Protocol PQ Review**

Score: 80%

This is a Process Quality Review of B protocol completed on 2/3/2021. It was performed using the Process Review process (version 0.6.1) and is documented here. The review was performed by Lucas of DeFiSafety. Check out our Telegram.

The final score of the review is 80%, a clear pass. 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.

This completed report is copyright (c) DeFiSafety 2021. Permission is given to copy in whole, retaining this copyright label.

### **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)



They are available from the GitHb readme; https://github.com/backstop-protocol/dss-cdp-manager/blob/update-readme/DEPLOYED.md as detailed in the Appendix.

# Is the code actively being used? (%)



For contract BCdpManager there are more than 10 transactions a week, 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)



GitHub: https://github.com/backstop-protocol/dss-cdp-manager

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? (%)



There are 488 commits and 17 branches making this a healthy repo.

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)



Information about the team can be found on their website.

## **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)



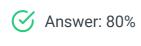
Location: https://github.com/backstop-protocol/whitepaper/blob/master/whitepaper.pdf

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



Basic docs are evident in the GitHub readme

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



#### 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 (%)



There are an extremely small set of comments present in the functions.

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.

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



Answer: 40%

The documentation lists all the functions and describes their functions.

#### 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? (%)



Test to Code ration is 915% so definitely a full 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:

TtC > 120% Both unit and system test visible
 TtC > 80% Both unit and system test visible
 TtC < 80% Some tests visible</li>
 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%

No code coverage results, so 50% die to a full test suite.

#### 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

No testing instructions found in the readme.

#### 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%

No report of testing was found.

#### 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%

No formal validation is evident.

# **Stress Testing environment (%)**



Answer: 0%

No stress testing network is evident.

### **Audits**



B. Protocol has an audit, as indicated below.

Location: https://github.com/solidified-platform/audits/blob/master/Audit%20Report%20-%20Backstop%20Protocol%20%5B02.10.2020%5D.pdf

#### 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.

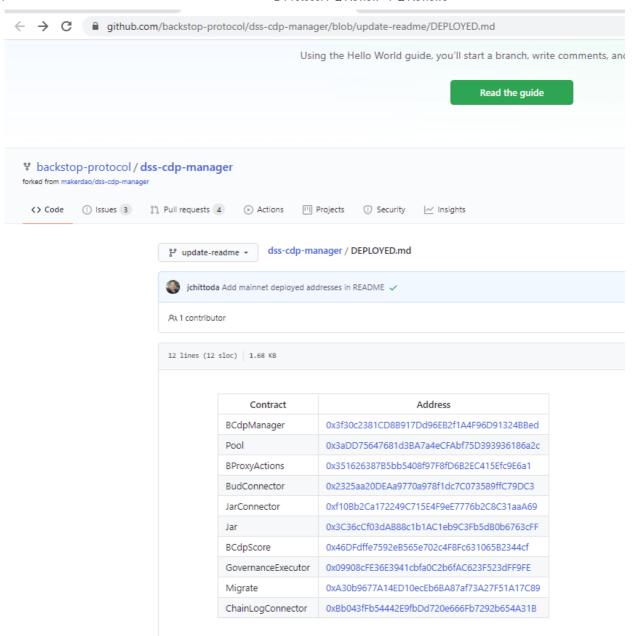
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

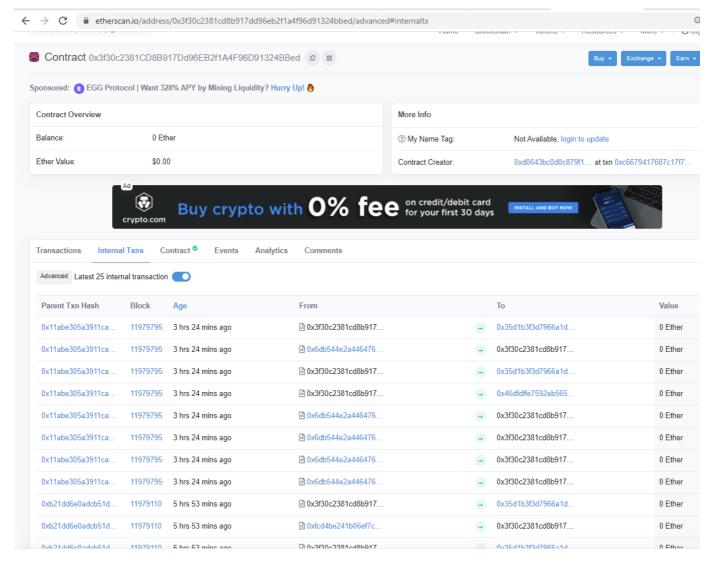
# **Scoring Appendix**

		B. Protocol	
PQ Audit Scoring Matrix (v0.6)	Points	Answer	Points
Tota	240		192.7
Code and Team			80%
Are the executing code addresses readily available? (Y/N)	30	Υ	30
2. Is the code actively being used? (%)	10	70%	7
3. Is there a public software repository? (Y/N)	5	Υ	5
4. Is there a development history visible? (%)	5	100%	5
Is the team public (not anonymous)? (Y/N)	20	Υ	20
Code Documentation			
1. Is there a whitepaper? (Y/N)	5	Y	5
2. Are the basic software functions documented? (Y/N)	10	Υ	10
3. Does the software function documentation fully (100%) cover the deployed contracts? (%)	15	80%	12
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)	10	12%	1.2
5 Is it possible to trace from software documentation to the implementation in code (%)	5	40%	2
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	Υ	5
5. Report of the results (%)	10	0%	0
6. Formal Verification test done (%)	5	0%	0
7. Stress Testing environment (%)	5	0%	0
Audits			
Audit done	70	90%	63
Section Scoring			
Code and Team	70	96%	
Documentation	45	67%	
Testing	55	59%	
Audits	70	90%	

# **Executing Addresses**



## **Code Used Appendix**



## **Example Code Appendix**

```
contract DssCdpManager is LibNote {
2
       address
                                   public vat:
                                   public cdpi;
                                                      // Auto incremental
3
                                                      // CDPId => UrnHandler
4
       mapping (uint => address) public urns;
       mapping (uint => List)
                                                      // CDPId => Prev & Next CDP:
5
                                  public list;
       mapping (uint => address) public owns;
                                                      // CDPId => Owner
6
7
       mapping (uint => bytes32) public ilks;
                                                      // CDPId => Ilk
8
9
       mapping (address => uint) public first;
                                                     // Owner => First CDPId
       mapping (address => uint) public last;
                                                     // Owner => Last CDPId
10
       mapping (address => uint) public count;
                                                     // Owner => Amount of CDPs
11
12
       mapping (
13
            address => mapping (
14
                uint => mapping (
15
                    address => uint
16
17
18
                                                      // Owner => CDPId => Allowed
19
       ) public cdpCan;
20
```

## **SLOC Appendix**

}

}

54

55 56

58

### **Solidity Contracts**

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	10	866	170	73	623	49

function sub(uint x, uint y) internal pure returns (uint z) {

require( $(z = x - y) \le x$ );

### **Javascript Tests**

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	13	7749	1695	355	5699	165

Tests to Code 5699/623 = 915%