0.7

unFederalReserve Process Quality Review

Score: 74%

Overview

This is a unFederalReserve Process Quality Review completed on June 7th 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 74%, 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%

They are available on https://lending-docs.unfederalreserve.com/docs/developer/smart-contract-address/ 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 labelling not clear or easy to find
0% Executing addresses could not be found

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.

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

> Answer: 100%

Activity is 60 transactions a day on contract *Unitroller.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/UnFederalReserve

Note: Github link does not exist on their website, it has to be manually searched on the web.

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: 50%

With 65 commits and 2 branches, this is a semi-acceptable 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.

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

> Answer: Yes

You can find team information at https://unfederalreserve.com/team

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://unfederalreserve.com/docs/Residual_Token_Whitepaper_091520_CONFIDE.pdf

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.

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

> ✓ Answer: Yes

Function documentation available at https://lending-docs.unfederalreserve.com/docs/developer/untokens/

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: 100%

All contracts and functions are covered at https://lending-docs.unfederalreserve.com/docs/developer/untokens/.

They also have APY documentation at https://lending-docs.unfederalreserve.com/docs/lending/interest-rate-model/

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: 63%

Code examples are in the Appendix. As per the SLOC, there is 63% commenting to code (CtC).

The Comments to Code (CtC) ratio is the primary metric for this score.

Note: This is essentially the CtC of their Compound fork, which is their only extensive repository.

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: 100%

There is clear explicit traceability found at https://lending-docs.unfederalreserve.com/docs/developer/untokens/.

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;

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%

As per the SLOC, there is 95% 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.

Note: This is essentially the TtC of their Compound fork.

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%

Scripts to analyze Code coverage at https://github.com/UnFederalReserve/compound-protocol/blob/master/script/coverage, but no value for coverage is indicated.

Note: This is once again based off of their Compound fork.

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

Instructions can be found at the bottom of this page: https://github.com/UnFederalReserve/compound-protocol

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.

14) Report of the results (%)

⚠ Answer: 0%

There is no evidence of a 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 formal verification of unFederalReserve was performed.

16) Stress Testing environment (%)

✓ Answer: 100%

They have their own testing server at https://ledger-stage.unfederalreservetesting.com/

Extensive test-net repository at https://github.com/UnFederalReserve/compound-protocol/tree/master/networks

official Rinkeby addresses found at https://lending-docs.unfederalreserve.com/docs/developer/smart-contract-address/

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%

Hacken did a unFederalReserve token audit on March 19th 2021. No indication if issues were resolved.

Coinspect did a unFederalReserve token audit in April 2021. Issues remain unfixed.

Both audits, the outstanding changes are very minor and should not result in penalization. The 70% score stands.

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: 70%

Bug bounty info found at https://lending-docs.unfederalreserve.com/docs/security/bug-bounty/

Reward as high as 150k, and active since May 20th 2021.

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: 70%

Admin controls can be found on page 42 of https://unfederalreserve.com/docs/Residual_Token_Whitepaper_091520_CONFIDE.pdf, and page 2 of https://unfederalreserve.com/docs/LitePaperRSDL.pdf.

Note: unFederalReserve is indicated as being immutable on their home page at https://unfederalreserve.com/, however there is contradictory information 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: 60%

Note: unFederalReserve is described as immutable on their home page at https://unfederalreserve.com/, but is described as being upgradeable in their whitepaper on page 42 at https://unfederalreserve.com/docs/Residual_Token_Whitepaper_091520_CONFIDE.pdf.

Note 2: Their token contract is once again described as immutable on page 2 of https://unfederalreserve.com/docs/LitePaperRSDL.pdf.

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	MacaronSwap	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		129.15
Code and Team			**50%**
1) Are the executing code addresses readily available? (%)	**20**	20%	4
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**	50%	2.5
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**	N	0
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	**15**	0%	0
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	63%	3.15
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	0%	0
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**	60%	3
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**	50%	35
18) Is the bug bounty acceptable high? (%)	**10**	70%	7
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	70%	3.5
20) Is the information clear and complete	**10**	60%	6

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	63%	
Documentation	45	18%	
Testing	50	58%	
Security	80	53%	
Access Controls	35	53%	

Executing Code Appendix

unFederalReserve
eRSDL

Search

- Introduction
  - How to Borrow
  - How to Claim Rewards
  - How to Connect a Wallet
  - How to Repay a Borrow
  - How to Supply
  - How to Withdraw
- Security
  - April 2021
  - March 2021
  - Bug Bounty
- Lending
  - Price Oracle
  - CompSpeed
  - Interest Rate Model
- Developer
  - unToken
  - Contract Address
  - Error codes
  - Key Events
- Faq
- Github

Contract Address

eRSDL

Contract

Mainnet addresses

Duration	Address
Unitroller	0x3105D328c66d8d55092358cF595d54608178E9B5
Comptroller	0xBb509D78d31fE327c18e58c4c83621BAE0CFE4FC
Timelock	0x1676174ee643cd5020a711d91e4d0f3b99354c13
Price oracle	0x9c48aaee3243a2e0c3b991386dc39f0027cf2ad7
uneRDSL	0xE4cC5A22B39fFB0A56d67F94f9300db20D786a5F
eRDSL	0x5218e472cfcfe0b64a064f055b43b4cdc9efd3a6

Rinkeby addresses

Duration	Address
Unitroller	0x44abc8395f35b6290af32601234fe11954808011
Comptroller	0x3E419553fa0477e6D3B0dBB6d88d270cA29bca1e
Timelock	0xbcbddc327585a277fb8c0a8018fc9e8b8e7f7960
Price oracle	0x305e44243b53a71e92cf1a20594298d9c695aa2d
uneRDSL	0xe2BBB422caaCFcACc2Daa3Aee26b3f49591Db764

Code Used Appendix

Ether Transactions for 0x3105D328c66d8d55092358cF595d54608178E9B5
Source: Etherscan.io

Example Code Appendix

```
pragma solidity ^0.5.16;

import "./ErrorReporter.sol";
import "./ComptrollerStorage.sol";
/**
 * @title ComptrollerCore
 * @dev Storage for the comptroller is at this address, while execution is delegated to th
 * CTokens should reference this contract as their comptroller.
 */
contract Unitroller is UnitrollerAdminStorage, ComptrollerErrorReporter {

    /**
      * @notice Emitted when pendingComptrollerImplementation is changed
      */
    event NewPendingImplementation(address oldPendingImplementation, address newPendingImp

    /**
      * @notice Emitted when pendingComptrollerImplementation is accepted, which means com
      */
    event NewImplementation(address oldImplementation, address newImplementation);

    /**
      * @notice Emitted when pendingAdmin is changed
      */
    event NewPendingAdmin(address oldPendingAdmin, address newPendingAdmin);

    /**
      * @notice Emitted when pendingAdmin is accepted, which means admin is updated
      */
    event NewAdmin(address oldAdmin, address newAdmin);

    constructor() public {
        // Set admin to caller
        admin = msg.sender;
    }

    /*** Admin Functions ***/
```

```
38      function _setPendingImplementation(address newPendingImplementation) public returns (u
39
40          if (msg.sender != admin) {
41              return fail(Error.UNAUTHORIZED, FailureInfo.SET_PENDING_IMPLEMENTATION_OWNER_CH
42          }
43
44          address oldPendingImplementation = pendingComptrollerImplementation;
45
46          pendingComptrollerImplementation = newPendingImplementation;
47
48          emit NewPendingImplementation(oldPendingImplementation, pendingComptrollerImplemen
49
50          return uint(Error.NO_ERROR);
51      }
52
53      /**
54      * @notice Accepts new implementation of comptroller. msg.sender must be pendingImplemen
55      * @dev Admin function for new implementation to accept it's role as implementation
56      * @return uint 0=success, otherwise a failure (see ErrorReporter.sol for details)
57      */
58      function _acceptImplementation() public returns (uint) {
59          // Check caller is pendingImplementation and pendingImplementation ≠ address(0)
60          if (msg.sender != pendingComptrollerImplementation || pendingComptrollerImplementa
61              return fail(Error.UNAUTHORIZED, FailureInfo.ACCEPT_PENDING_IMPLEMENTATION_ADDR
62          }
63
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	44	15331	2428	4993	7910	1569

Comments to Code 4993/7910 = 63%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	48	9138	1352	244	7542	313

Tests to Code 7542/7910 = 95%