0.7

Inverse Finance Process Quality Review

Score: 47%

Overview

This is a Process Quality Review of Inverse Finance completed on May 17th 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 47%, a fail. 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 at website https://docs.inverse.finance/smart-contracts 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

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

> Answer: 100%

Activity is around 20 transactions a day on contract *GovernorAlpha.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/InverseFinance/inverse-protocol

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: 70%

With 73 commits and 3 branches, this is a semi-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.

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

> Answer: Yes

Team members public in their Medium.

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.inverse.finance/

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

> Answer: Yes

There is evident software function documentation at https://docs.inverse.finance/user-guides/yield-farming

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

> Answer: 80%

There is evident major software function documentation at https://docs.inverse.finance/user-guides/anchor-lending-and-borrowing/developer-guide

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: 30%

Code examples are in the Appendix. As per the SLOC, there is 30% 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: 100%

There is clear explicit traceability between code and documentation in their developer guides at

https://docs.inverse.finance/vaults/developer-guide

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: 22%

With a TtC of 22%, this is a tiny 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: 0%

There is no evident testing for code coverage.

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 test instructions available at https://github.com/InverseFinance/inverse-protocol

14) Report of the results (%)

⚠ Answer: 0%

No test report visible in their documentation.

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 any Inverse Finance formal verification found.

16) Stress Testing environment (%)

> ⚠ Answer: 0%

No testing smart contract addresses found in their documentation or software repo.

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: 20%

No audits of Inverse Finance were performed.

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: 0%

There is no evident bug bounty program.

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%

Location: https://docs.inverse.finance/governance.

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: 30%

a) some info give can change variables are they upgradeable? - 10%
b) They say they have an admin key - 20%

c) Virtually no info - 0%
Score 30%

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: 30%

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	inverse.finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		121.4
Code and Team			**47%**
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**	70%	3.5
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**	30%	1.5
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**	22%	4.4
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	0%	0
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**	0%	0

16) Stress Testing environment (%)	5	0%	0
Security			
17) Did 3rd Party audits take place? (%)	70	20%	14
18) Is the bug bounty acceptable high? (%)	10	0%	0
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	30%	3
21) Is the information in non-technical terms	10	30%	3
22) Is there Pause Control documentation including records of tests	10	0%	0

Executing Code Appendix

Deployer

Inverse deployer : 0x3fcb35a1cbfb6007f9bc638d388958bc4550cb28

Governance

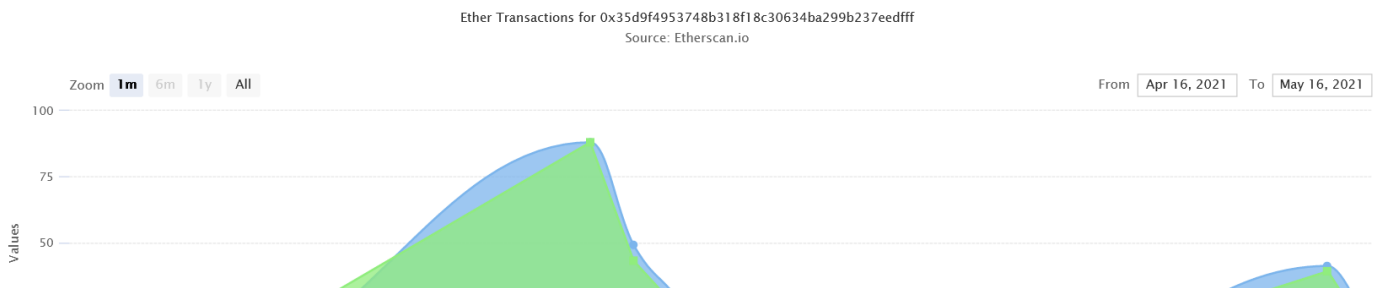
GovernorAlpha contract: 0x35d9f4953748b318f18c30634bA299b237eeDfff

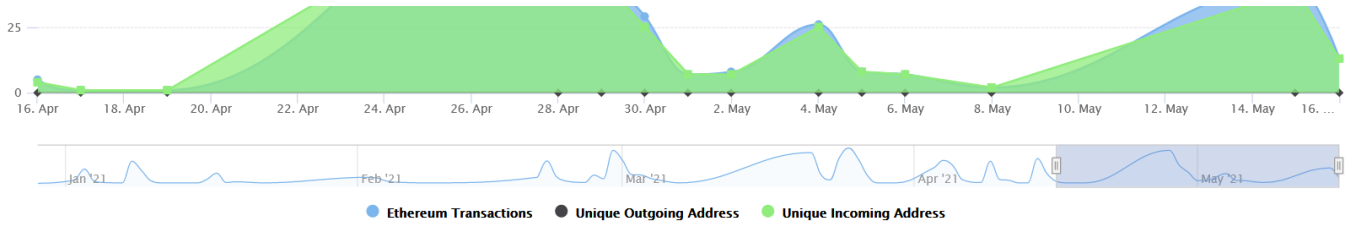
Tokens

INV contract : 0x41d5d79431a913c4ae7d69a668ecdfe5ff9dfb68

DOLA contract : 0x865377367054516e17014ccded1e7d814edc9ce4

Code Used Appendix





Example Code Appendix

```
pragma solidity ^0.5.16;
pragma experimental ABIEncoderV2;

import "./Timelock.sol";

contract GovernorAlpha {
    /// @notice The name of this contract
    string public constant name = "Inverse Governor Alpha";

    /// @notice The number of votes in support of a proposal required in order for a quorur
    function quorumVotes() public pure returns (uint) { return 4000e18; } // 4% of INV

    /// @notice The number of votes required in order for a voter to become a proposer
    function proposalThreshold() public pure returns (uint) { return 1000e18; } // 1% of IN

    /// @notice The maximum number of actions that can be included in a proposal
    function proposalMaxOperations() public pure returns (uint) { return 20; } // 10 actior

    /// @notice The delay before voting on a proposal may take place, once proposed
    function votingDelay() public pure returns (uint) { return 1; } // 1 block

    /// @notice The duration of voting on a proposal, in blocks
    function votingPeriod() public pure returns (uint) { return 17280; } // ~3 days in bloc

    /// @notice The address of the Protocol Timelock
    Timelock public timelock;

    /// @notice The address of the governance token
    InvInterface public inv;

    /// @notice The total number of proposals
    uint public proposalCount;

    struct Proposal {
        /// @notice Unique id for looking up a proposal
        uint id;

        /// @notice Creator of the proposal
        address proposer;

        /// @notice The timestamp that the proposal will be available for execution, set or
        uint eta;

```

```
44          /// @notice the ordered list of target addresses for calls to be made
45          address[] targets;
46
47          /// @notice The ordered list of values (i.e. msg.value) to be passed to the calls
48          uint[] values;
49
50          /// @notice The ordered list of function signatures to be called
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	26	2139	369	410	1360	193

Comments to Code 410/1360 = 30%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	3	395	57	44	294	0

Tests to Code 294/1360 = 22%