0.7

Mirror Finance Process Quality Review

Score: 86%

Overview

This is a Mirror Finance Process Quality Review completed on July 28th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is 86%, 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

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.

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain:** Terra

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra

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 following 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.mirror.finance/networks, 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 labeling not clear or easy to find
0% Executing addresses could not be found

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

> **Answer:** 100%

Activity is over 10 transactions a day on contract *Staking.rs*, as indicated in the Appendix.

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/Mirror-Protocol/mirror-contracts.

Is there a public software repository with the code at a minimum, but also normally test and scripts. Even if the repository was created just to hold the files and has just 1 transaction, it gets a **"Yes"**. For teams with private repositories, this answer is **"No"**.

4) Is there a development history visible? (%)

> **Answer:** 100%

With 317 commits and 5 branches, this is a very healthy repository.

This metric 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 30 commits

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

> **Answer:** Yes

Location: https://messari.io/person/do-kwon (founder).

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, 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.mirror.finance/.

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

> **Answer:** Yes

The Terra team has documented all of the Mirror Finance basic software functions in the "Developers" section of their documentation.

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

>

Answer: 100%

All Mirror Finance's contracts have their software functions documented in the "Smart Contracts" section of their documentation, as well as in the developer-tools repository.

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

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 9% commenting to code (CtC).

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

Note: To calculate the CtC, we used the "packages" file of the mirror-contracts repository as it contains all of the necessary core contracts of the protocol. We did not use any mocks, interface, migrations, multiples, or any third-party files into our calculations.

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 a clear association between code and documentation via non-explicit traceability to their implementations in the Mirror Finance source code. However, there is clear and explicit traceability in the README.md of their main contracts' GitHub repository, as well as in the developer-tools repository.

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:** 100%

Code examples are in the Appendix. As per the SLOC, there is 1009% 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.

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

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

> **Answer:** 75%

Mirror Finance does not have official code coverage but does have a robust testing suite as well as detailed testing in their DocSend v1 audit report. Although Mirror Finance is now at v2, the changes brought in the update have not changed the results of those tests (they mainly added governance).

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 improved by adding tests that achieve 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

Scrips/Instructions location: https://github.com/Mirror-Protocol/mirror-contracts/blob/master/README.md.

14) Report of the results (%)

> ⚠ **Answer:** 0%

No Mirror Finance test report was found in their documentation or in their various GitHub repositories.

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 Mirror Finance Formal Verification test was found in their documentation.

16) Stress Testing environment (%)

> **Answer:** 100%

There is evidence of Mirror Finance's test-net smart contract usage in their documentation.

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:** 100%

CyberUnit published a Mirror Finance audit report in October 2020.

Cryptonics published a Mirror Finance audit report on June 22nd 2021.

Mirror Finance Mainnet v1 was launched in December 2020

Mirror Finance Mainnet v2 was launched on June 25th 2021.

Note: All results are public and most fix recommendations were successfully implemented by the Terra team.

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

50% Audit(s) performed after deployment and changes needed but not implemented
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, (where question 1 is 0%)

Deduct 25% if code is in a private repo and no note from auditors that audit is applicable to deployed code

18) Is the bounty value acceptably high (%)

> **Answer:** 70%

Mirror Finance's Bug Bounty Program is active and rewards participating users with up to 150k for the most critical of bug finds.

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
60% Bounty is 100k or over
50% Bounty is 50k or over AND active program
40% Bounty is 50k or over
20% Bug bounty program bounty is less than 50k
0% No bug bounty program offered

An active program means that a third party (such as Immunefi) is actively driving hackers to the site. An inactive program would be static mentions 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?
21) 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 access controls (%)

> **Answer:** 100%

Mirror Finance has a clearly labelled "Governance" section 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:** 90%

a) The contracts are clearly labelled as upgradeable within Mirror Finance's voting process documentation.

b) Governance contract is OnlyOwner of itself and there are defined community roles when it comes to the communal voting process.

c) Capabilities for change in contracts through proposal voting is described in "Proposal Types".

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%

All governance parameter information is in user-friendly language that pertains investment safety.

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%

There is no evidence of Pause Control or a similar function in the Mirror Finance documentation or GitHub repositories.

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	Mirror Finance

PQ Audit Scoring Matrix (v0.7)	Points	Answer	Points
Total	260		223.75
Code and Team			**86%**
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	100%	15
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	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	75%	3.75
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	100%	70
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	100%	5
20) Is the information clear and complete	10	90%	9
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	89%	
Testing	50	68%	
Security	80	96%	
Access Controls	35	66%	

Executing Code Appendix

Mainnet (v2)

Network chain ID: `columbus-4`

Core Contracts

Contract	Address
Collector	terra1s4fllut0e6vw0k3fxsg4fs6fm2ad6hn0prqp3s
Community	terra1x35fvy3sy47drd3qs288sm47fjzjnksuwpyl9k
Factory	terra1mzj9nsxx0lxlaxnekleqdv8xnvw2qrh3uz6h8p

Gov	terra1wh39swv7nq36pnefnupttm2nr96kz7jjddyt2x
Mint	terra1wfz7h3aqf4cjmjcvc6s8lxdhh7k30nkczyf0mj
Oracle	terra1t6xe0txzywdg85n6k8c960cuwgh6l8esw6lau9
Staking	terra17f7zu97865jmknk7p2glqvxzhduk78772ezac5

Code Used Appendix

Transactions

Tx hash	Type	Block	Amount (Out)	Amount (In)	Timestamp	Fee
D2E94341...7A433E66 ✓	MsgExecuteContract	3923142(columbus-4)	-	+6.800702 MIR	2021.07.28 16:35:28-04:00	0.100000 UST
42B68755...AAB9CCD5 ✓	MsgExecuteContract	3923141(columbus-4)	-	+54.053673 MIR	2021.07.28 16:35:20-04:00	0.100000 UST
2AB76D3C...7A80B691 ✓	MsgExecuteContract	3923135(columbus-4)	-	+102.458636 MIR	2021.07.28 16:34:40-04:00	0.100000 UST
7507C875...ECB2CDE4 ✓	MsgExecuteContract	3923132(columbus-4)	-218.263548 UST -0.500000 mSPY	+10.230356 mSPY-UST LP	2021.07.28 16:34:18-04:00	1.249819 UST
30B7598B...A59A4EFF ✓	MsgExecuteContract	3923132(columbus-4)	-251.476175 mQQQ-UST LP	+4,995.618306 UST +13.648606 mQQQ	2021.07.28 16:34:18-04:00	0.200000 UST
1CF792D5...DABFD641 ✓	MsgExecuteContract	3923128(columbus-4)	-	+22.119350 MIR	2021.07.28 16:33:44-04:00	0.100000 UST
E72BF20C...BDDA852C ✓	MsgExecuteContract	3923128(columbus-4)	-11.900682 UST -4.105559 MIR	+4.779798 MIR-UST LP	2021.07.28 16:33:44-04:00	0.257241 UST
7D1F1F25...ACA930FE ✓	MsgExecuteContract	3923127(columbus-4)	-	+139.115184 aUST +16,360.884816 aUST ...	2021.07.28 16:33:37-04:00	0.300000 UST

Example Code Appendix

```
pub fn auto_stake<S: Storage, A: Api, Q: Querier>(
    deps: &mut Extern<S, A, Q>,
    env: Env,
    assets: [Asset; 2],
    slippage_tolerance: Option<Decimal>,
) -> HandleResult {
    let config: Config = read_config(&deps.storage)?;
    let terraswap_factory: HumanAddr = deps.api.human_address(&config.terraswap_factory)?;

    let mut native_asset_op: Option<Asset> = None;
    let mut token_info_op: Option<(HumanAddr, Uint128)> = None;
    for asset in assets.iter() {
        match asset.info.clone() {
            AssetInfo::NativeToken { .. } => {
                asset.assert_sent_native_token_balance(&env)?;
                native_asset_op = Some(asset.clone())
```

```
            }
            AssetInfo::Token { contract_addr } => {
                token_info_op = Some((contract_addr, asset.amount))
            }
        }
    }

    // will fail if one of them is missing
    let native_asset: Asset = match native_asset_op {
        Some(v) => v,
        None => return Err(StdError::generic_err("Missing native asset")),
    };
    let (token_addr, token_amount) = match token_info_op {
        Some(v) => v,
        None => return Err(StdError::generic_err("Missing token asset")),
    };

    // query pair info to obtain pair contract address
    let asset_infos: [AssetInfo; 2] = [assets[0].info.clone(), assets[1].info.clone()];
    let terraswap_pair: PairInfo = query_pair_info(deps, &terraswap_factory, &asset_infos)

    // assert the token and lp token match with pool info
    let pool_info: PoolInfo =
        read_pool_info(&deps.storage, &deps.api.canonical_address(&token_addr)?)?;

    if pool_info.staking_token
        != deps
            .api
            .canonical_address(&terraswap_pair.liquidity_token)?
    {
        return Err(StdError::generic_err("Invalid staking token"));
    }

    // get current lp token amount to later compute the recived amount
    let prev_staking_token_amount = query_token_balance(
        &deps,
        &terraswap_pair.liquidity_token,
        &env.contract.address,
    )?;

    // compute tax
    let tax_amount: Uint128 = native_asset.compute_tax(deps)?;

    // 1. Transfer token asset to staking contract
    // 2. Increase allowance of token for pair contract
    // 3. Provide liquidity
    // 4. Execute staking hook, will stake in the name of the sender
    Ok(HandleResponse {
        messages: vec![
            CosmosMsg::Wasm(WasmMsg::Execute {
                contract_addr: token_addr.clone(),
                msg: to_binary(&Cw20HandleMsg::TransferFrom {
                    owner: env.message.sender.clone(),
```

```
                    recipient: env.contract.address.clone(),
                    amount: token_amount,
                })?,
                send: vec![],
            }),
            CosmosMsg::Wasm(WasmMsg::Execute {
                contract_addr: token_addr.clone(),
                msg: to_binary(&Cw20HandleMsg::IncreaseAllowance {
                    spender: terraswap_pair.contract_addr.clone(),
                    amount: token_amount,
                    expires: None,
                })?,
                send: vec![],
            }),
            CosmosMsg::Wasm(WasmMsg::Execute {
                contract_addr: terraswap_pair.contract_addr,
                msg: to_binary(&PairHandleMsg::ProvideLiquidity {
                    assets: [
                        Asset {
                            amount: (native_asset.amount.clone() - tax_amount)?,
                            info: native_asset.info.clone(),
                        },
                        Asset {
                            amount: token_amount,
                            info: AssetInfo::Token {
                                contract_addr: token_addr.clone(),
                            },
                        },
                    ],
                    slippage_tolerance,
                })?,
                send: vec![Coin {
                    denom: native_asset.info.to_string(),
                    amount: (native_asset.amount - tax_amount)?,
                }],
            }),
            CosmosMsg::Wasm(WasmMsg::Execute {
                contract_addr: env.contract.address,
                msg: to_binary(&HandleMsg::AutoStakeHook {
                    asset_token: token_addr.clone(),
                    staking_token: terraswap_pair.liquidity_token,
                    staker_addr: env.message.sender,
                    prev_staking_token_amount,
                })?,
                send: vec![],
            }),
        ],
        log: vec![
            log("action", "auto_stake"),
            log("asset_token", token_addr.to_string()),
            log("tax_amount", tax_amount.to_string()),
        ],
        data: None,
```

```
123         })
124 }
125
126 pub fn auto_stake_hook<S: Storage, A: Api, Q: Querier>(
127     deps: &mut Extern<S, A, Q>,
128     env: Env,
129     asset_token: HumanAddr,
130     staking_token: HumanAddr,
131     staker_addr: HumanAddr,
132     prev_staking_token_amount: Uint128,
133 ) -> HandleResult {
134     // only can be called by itself
135     if env.message.sender != env.contract.address {
136         return Err(StdError::unauthorized());
137     }
138
139     // stake all lp tokens received, compare with staking token amount before liquidity pr
140     let current_staking_token_amount =
141         query_token_balance(&deps, &staking_token, &env.contract.address)?;
142     let amount_to_stake = (current_staking_token_amount - prev_staking_token_amount)?;
143
144     bond(deps, env, staker_addr, asset_token, amount_to_stake)
145 }
146
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Rust	13	1272	129	94	1049	7

Comments to Code 94/1049 = 9%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
Rust	18	12137	1124	433	10580	5

Tests to Code 10580/1049 = 1009%