Armor.fi Process Quality review

Score:74%

Overview

This is a Process Quality Review of Armor.fi completed on May 13, 2021. It was performed using the Process Review process (version 0.7) and is documented here. The review was performed by Lucas 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

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: 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://github.com/ArmorFi/arNXM 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: 40%

Activity is 3 transactions a day on contract *arNFT.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/ArmorFi

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 209 commits and 7 branches, this is a healthy 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

The names of the team members are not public.

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://armorfi.gitbook.io/armor/

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

Answer: Yes

There is a small amount of function documentation in their GitHub Repositories.

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

Answer: 40%

Armor.fi documents some of the major contracts and functions.

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: 42%

Code examples are in the Appendix. As per the SLOC, there is 42% 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: 40%

The documentation lists 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;

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%

With a TtC of 165%, this is clearly a well-tested protocol.

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: 50%

There is no indication of code coverage, but there is clearly a reasonable 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

location: https://github.com/ArmorFi/arNXM

14) Report of the results (%)

Answer: 0%

There is no evident report of the test results.

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%

There is no evidence of formal verification testing having been done.

16) Stress Testing environment (%)

> ⚠ Answer: 0%

There are no published test net addresses.

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%

There have been 2 audits that have been performed on this protocol, with the earliest being released on the 16th of October, 2020.

Armor.fi was released on the 20th of January, 2021.

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: 100%

Bug Bounty Location: https://armorfi.gitbook.io/armor/developers/bug-bounties

The highest possible bug bounty is 1,150,000$.

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: 40%

The development team owns a team multisig.

Location: https://armorfi.gitbook.io/armor/products/armordao-hybrid-decentralization

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: 70%

All contracts are clearly labelled as behind a proxie --> 30%
The type of ownership is team MultiSig with TimeLock --> 30%
The capabilities for change are implied as fully upgradable --> 10%
30+30+10=70%

Guidance:
All the contracts are immutable -- 100% OR

All contracts are clearly labelled as upgradeable (or not) -- 30% AND
The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
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%

The descriptions are written in software specific language.

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 documentation indicating any pause control testing having been done on this platform.

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	Armor.fi	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		193.6
Code and Team			**74%**
1) Are the executing code addresses readily available? (%)	**20**	100%	20
2) Is the code actively being used? (%)	**5**	40%	2
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**	40%	6
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	42%	2.1
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	40%	4
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**	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**	0%	0
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	100%	10
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	40%	2
20) Is the information clear and complete	**10**	70%	7
21) Is the information in non-technical terms	**10**	30%	3
22) Is there Pause Control documentation including records of tests	**10**	0%	0
Section Scoring			
Code and Team	50	94%	
Documentation	45	60%	
Testing	50	55%	
Security	80	100%	
Access Controls	35	34%	

Executing Code Appendix

README.md

Testing

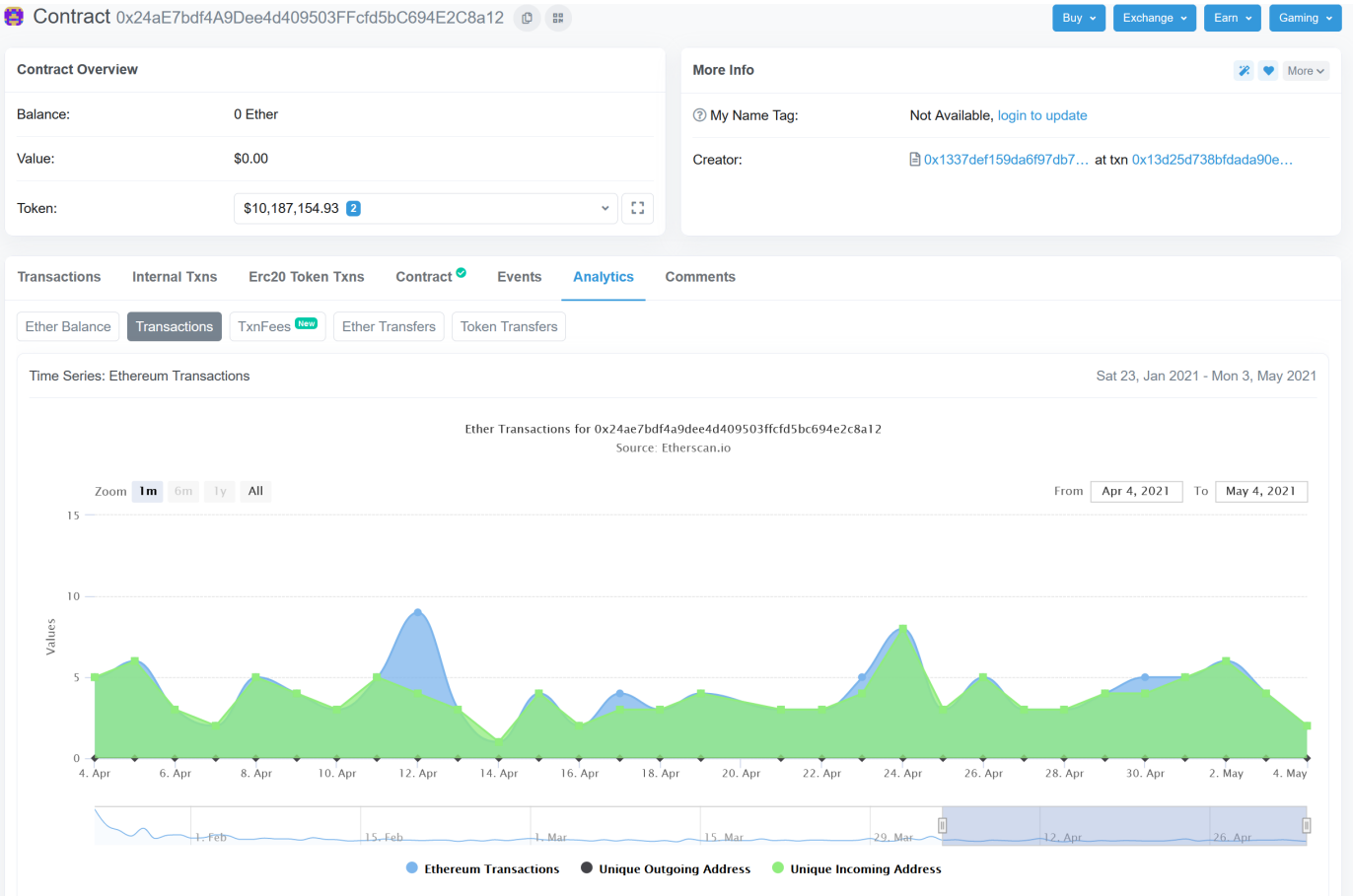
1. `git clone https://github.com/ArmorFi/arNXM.git`
2. `npm install --save-dev`
3. `npx hardhat test`

Contracts

arNXMVault Master: 0x7eFf1f18644b84A391788923d53400e8fe455687
ReferralRewards Master: 0xefF1CDc3CC01afAB104b00a7D9cd09619B94ae8F
arNXMVault Proxy: 0x1337DEF1FC06783D4b03CB8C1Bf3EBf7D0593FC4
ReferralRewards Proxy: 0x1337DEF1C79053dA23921a3634aDbD12f3b748A5
arNXM token: 0x1337DEF18C680aF1f9f45cBcab6309562975b1dD
arNFT token: 0x1337DEF1e9c7645352D93baf0b789D04562b4185
Old arNFT token: 0x57318daf32e1f208fb84af5413c4185b8f66104d
Multisig Admin: 0x1f28eD9D4792a567DaD779235c2b766Ab84D8E33
Timelock Owned: 0x1337DEF11D788e62A253feA846A505EE1b57623f
Armor token: 0x1337DEF16F9B486fAEd0293eb623Dc8395dFE46a
FarmController: 0x1337DEF159da6F97dB7c4D0E257dc689837b9E70
FarmController Master: 0x0Bdb7976c34aB05E5a9031F258B8956f68ee29cf

arNXM:ETH Uni: 0x24ae7bdf4a9dee4d409503ffcfd5bc694e2c8a12
arNXM:ETH Sushi: 0xcd1f8cda8be6a8c306a5b0ee759bad46a6f60cad
arNXM:ETH 1inch: 0x07aFD11985bFcAA8016eEb9b00534c0B3A70CCaC
arNXM:ETH Bal: 0x008F3DDE2Ed44BdC72800108d8309D16d55d6dD5
ARMOR:ETH Uni: 0xf991f1e1b8acd657661c89b5cd452d86de76a8c1
ARMOR:DAI Uni: 0xa659e66E116D354e779D8dbb35319AF67171ffb4
ARMOR:WBTC Uni: 0x01Acad2228F18598CD2b8611aCD37992BF27313C
ARMOR:ETH Sushi: 0x1b39d7f818aaf0318f6d0a66cd388c20c15fea94
ARMOR:DAI Sushi: 0x4529AAA39DE655c8B4715DEa8b1dACEbbA255C74
ARMOR:WBTC Sushi: 0x88aACE19997656F4eB1b8D3729226A4F97Ca6b2c
ARMOR:ETH 1inch: 0xfDF5709D44b26A7DD112556Dd1B1cE53c0eAF454
ARMOR:DAI 1inch: 0xD7b8Ef47C08F824ceA3d837afA61484e81d14BfB
ARMOR:WBTC 1inch: 0x8C7442Bd71A1464f50efb216407B59584a2bcfF5
ARMOR:DAI Bal: 0x148ac62a238a71D7fb8A5bA093B8BADF4DCc7DCC

Code Used Appendix



Example Code Appendix

```
pragma solidity ^0.6.6;

import '../general/Ownable.sol';
import '../libraries/SafeERC20.sol';
import '../interfaces/IWNXM.sol';
import '../interfaces/IERC20.sol';
import '../interfaces/INexusMutual.sol';
import '../interfaces/IRewardManager.sol';
import '../interfaces/IShieldMining.sol';
/**
 * @title arNXM Vault
 * @dev Vault to stake wNXM or NXM in Nexus Mutual while maintaining your l
 *      This is V2 which replaces V1 behind a proxy. Updated variables at tl
 * @author Armor.fi -- Robert M.C. Forster, Taek Lee
 * SPDX-License-Identifier: (c) Armor.Fi DAO, 2021
**/
contract arNXMVault is Ownable {

    using SafeMath for uint;
    using SafeERC20 for IERC20;

    uint256 constant private DENOMINATOR = 1000;

    // Amount of time between
    uint256 public restakePeriod;
```

```

    // Amount of time that rewards are distributed over.
    uint256 public rewardDuration;

    // This used to be unstake percent but has now been deprecated in favor
    // Paranoia results in this not being replaced but rather deprecated an
    uint256 public ____deprecated____;

    // Amount of wNXM (in token Wei) to reserve each period.
    // Overwrites reservePercent in update.
    uint256 public reserveAmount;

    // Withdrawals may be paused if a hack has recently happened. Timestamp
    uint256 public withdrawalsPaused;

    // Amount of time withdrawals may be paused after a hack.
    uint256 public pauseDuration;

    // Address that will receive administration funds from the contract.
    address public beneficiary;

    // Percent of funds to be distributed for administration of the contrac
    uint256 public adminPercent;

    // Percent of staking rewards that referrers get.
    uint256 public referPercent;

    // Timestamp of when the last restake took place--7 days between each.
    uint256 public lastRestake;

    // The amount of the last reward.
    uint256 public lastReward;

    // Uniswap, Maker, Compound, Aave, Curve, Synthetix, Yearn, RenVM, Bala
    address[] public protocols;

    // Amount to unstake each time.
    uint256[] private amounts;

    // Protocols being actively used in staking or unstaking.
    address[] private activeProtocols;

    // Nxm tokens.
    IERC20 public wNxm;
    IERC20 public nxm;
    IERC20 public arNxm;

    // Nxm Master address.
    INxmMaster public nxmMaster;

    // Reward manager for referrers.
    IRewardManager public rewardManager
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	52	4327	655	1104	2568	346

Comments to Code 1104/2568 = 42%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	25	4787	467	77	4243	213

Tests to Code 4243/2568 = 165%