

AAVE Protocol PQ Review

Score: 91%

This is a [AAVE Protocol V2](#) Process Quality Review completed on 27 December, 2020. [AAVE V1](#) was reviewed in July 2020. This review was performed using the Process Review process (version 0.6.1) and is documented [here](#). The review was performed by ShinkaRex of DeFiSafety. Check out our [Telegram](#).

The final score of the review is 91%, a great score. The breakdown of the scoring is in [Scoring Appendix](#).

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**

Disclaimer

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.


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

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is [here](#). This review will answer the questions;

1. Are the executing code addresses readily available? (Y/N)
2. Is the code actively being used? (%)
3. Is there a public software repository? (Y/N)
4. Is there a development history visible? (%)
5. Is the team public (not anonymous)? (Y/N)

Are the executing code addresses readily available? (Y/N)

 Answer: Yes

The contract addresses are in the “[Deployed Contracts](#)” section of the Developers docs, see Appendix: [Deployed Code](#).

Is the code actively being used? (%)

 Answer: 100%

Activity is in excess of 300 transactions a day on contract
0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9, as indicated in the [Appendix](#).

Percentage Score Guidance

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

Is there a public software repository? (Y/N)

 Answer: Yes

GitHub: <https://github.com/aave/protocol-v2>

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

Is there a development history visible? (%)

 Answer: 100%

With 1200+ commits (though only 3 branches) this is a very healthy repo.

This checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:

100% Any one of 100+ commits, 10+branches

70%	Any one of 70+ commits, 7+branches
50%	Any one of 50+ commits, 5+branches
30%	Any one of 30+ commits, 3+branches
0%	Less than 2 branches or less than 10 commits

How to improve this score

Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools. A public development history indicates clearly to the public the level of continued investment and activity by the developers on the application. This gives a level of security and faith in the application.

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

 Answer: Yes

Location: <https://www.linkedin.com/company/aaveaave>

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;

1. Is there a whitepaper? (Y/N)
2. Are the basic software functions documented? (Y/N)
3. Does the software function documentation fully (100%) cover the deployed contracts? (%)
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)
5. Is it possible to trace from software documentation to the implementation in codee (%)

Is there a whitepaper? (Y/N)

 Answer: Yes

Location: <https://docs.defisafety.com/old-reviews/aave-process-quality-audit>

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

 Answer: Yes

Location: <https://docs.aave.com/developers/>

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

 Answer: 90%

Location: <https://docs.aave.com/developers/the-core-protocol/lendingpool>

All external major functions are clearly documented with external variables clearly labelled. However internal functions are not documented fully. It is better than most API documentation, therefore 90%, not 80%.


Guidance:

- 100% All contracts and functions documented
- 80% Only the major functions documented
- 79-1% Estimate of the level of software documentation
- 0% No software documentation

How to improve this score

This score can improve by adding content to the requirements document such that it comprehensively covers the requirements. For guidance, refer to the [SecurEth System Description Document](#) . Using tools that aid traceability detection will help.

Are there sufficiently detailed comments for all functions within the deployed contract code (%)

 Answer: 43%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 43% commenting to code (CtC).

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

Guidance:

100% CtC > 100 Useful comments consistently on all code

90-70% CtC > 70 Useful comment on most code

60-20% CtC > 20 Some useful commenting

0% CtC < 20 No useful commenting

How to improve this score

This score can improve by adding comments to the deployed code such that it comprehensively covers the code. For guidance, refer to the [SecurEth Software Requirements](#).

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

 Answer: 80%

Very clear traceability for all major public functions.

Guidance:

100% - Clear explicit traceability between code and documentation at a requirement level for all code

60% - Clear association between code and documents via non explicit traceability

40% - Documentation lists all the functions and describes their functions

0% - No connection between documentation and code

How to improve this score

This score can improve by adding traceability from requirements to code such that it is clear where each requirement is coded. For reference, check the SecurEth guidelines on [traceability](#).

Testing

This section looks at the software testing available. It is explained in this [document](#). This section answers the following questions;

1. Full test suite (Covers all the deployed code) (%)
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)
3. Scripts and instructions to run the tests (Y/N)
4. Packaged with the deployed code (Y/N)
5. Report of the results (%)
6. Formal Verification test done (%)
7. Stress Testing environment (%)

Is there a Full test suite? (%)

 Answer: 100%

Test to Code ration is 254% as per the [SLOC](#).

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.

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

 Answer: 50%

No indication of coverage results.

Guidance:

100% - Documented full coverage

99-51% - Value of test coverage from documented results

50% - No indication of code coverage but clearly there is a reasonably complete set of tests

30% - Some tests evident but not complete

0% - No test for coverage seen

How to improve this score

This score can improve by adding tests achieving full code coverage. A clear report and scripts in the software repository will guarantee a high score.

Scripts and instructions to run the tests (Y/N)

 Answer: Yes

Instructions in the readme.

Packaged with the deployed code (Y/N)



Answer: Yes

Report of the results (%)



Answer: 0%

No indication 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.

Formal Verification test done (%)



Answer: 100%

Location: <https://github.com/aave/protocol-v2/blob/master/audits/Cetora-FV-aave-v2-03-12-2020.pdf>

Cetora formal verification report.

Stress Testing environment (%)



Answer: 100%

Location: <https://docs.aave.com/developers/deployed-contracts>

Kovan deployed contracts clearly indicated.

Audits

 Answer: 100%

Guidance:

1. Multiple Audits performed before deployment and results public and implemented or not required (100%)
2. Single audit performed before deployment and results public and implemented or not required (90%)
3. Audit(s) performed after deployment and no changes required. Audit report is public. (70%)
4. No audit performed (20%)
5. Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, question 1 (0%)

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

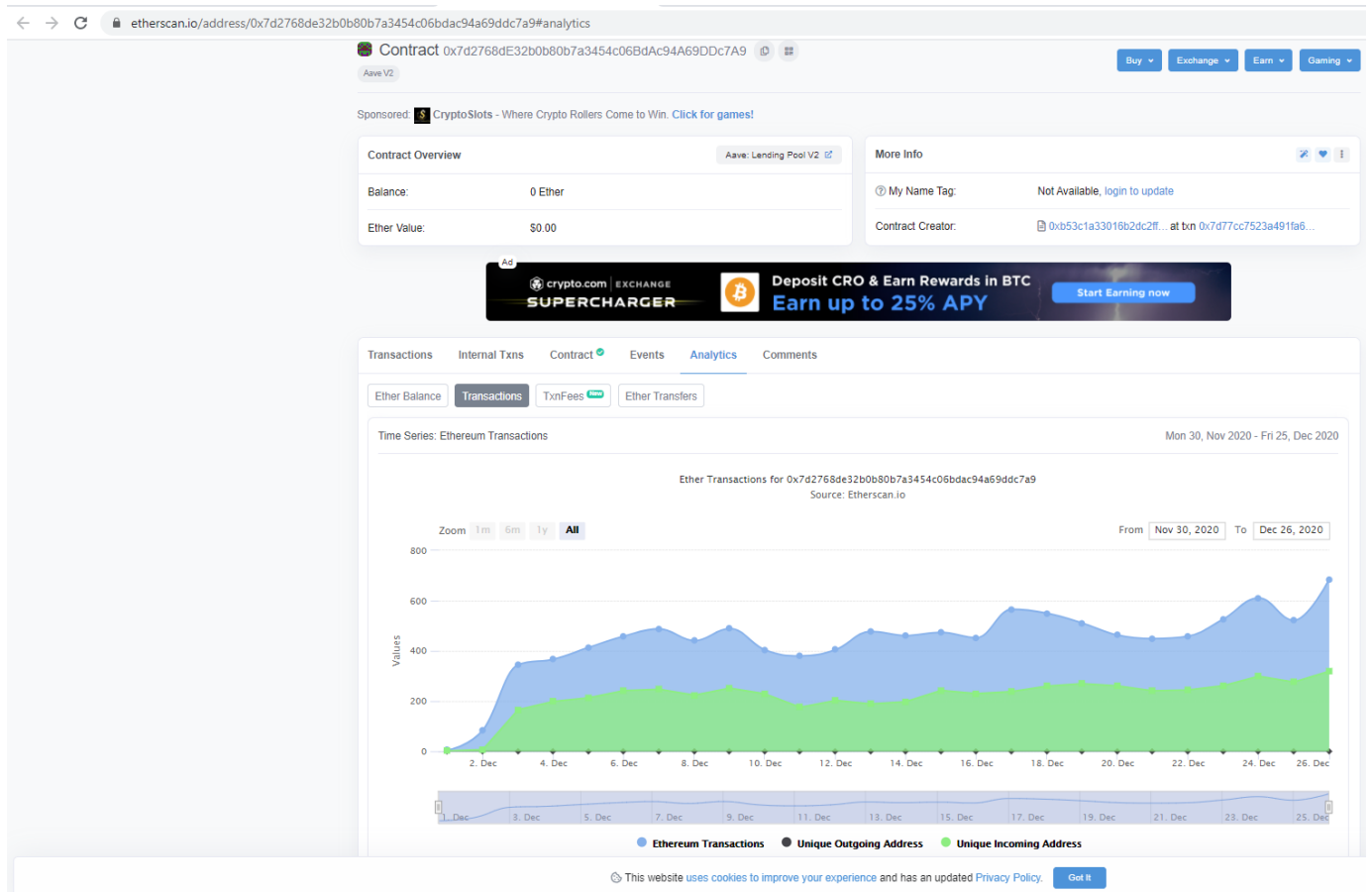
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started SecuEth.org with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got [EthFoundation funding](#) to assist in their development.

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

Career wise I am a business development manager for an avionics supplier.

Scoring Appendix



Executing Code Appendix

- v2 >
- Introduction
- THE CORE PROTOCOL
- Protocol Overview
- LendingPool >
- Addresses Provider >
- Addresses Provider Registry >
- Protocol Data Provider >
- aTokens >
- Debt Tokens >
- AAVE Token
- Price Oracle >
- Deployed Contracts**
- Changes from v1 to v2
- Security & Audits
- PROTOCOL GOVERNANCE
- Voting & Governance
- Staking AAVE
- GUIDES
- Credit Delegation
- Flash Loans >

Deployed Contracts

i You are viewing the V2 docs. For V1 contracts, [click here](#).

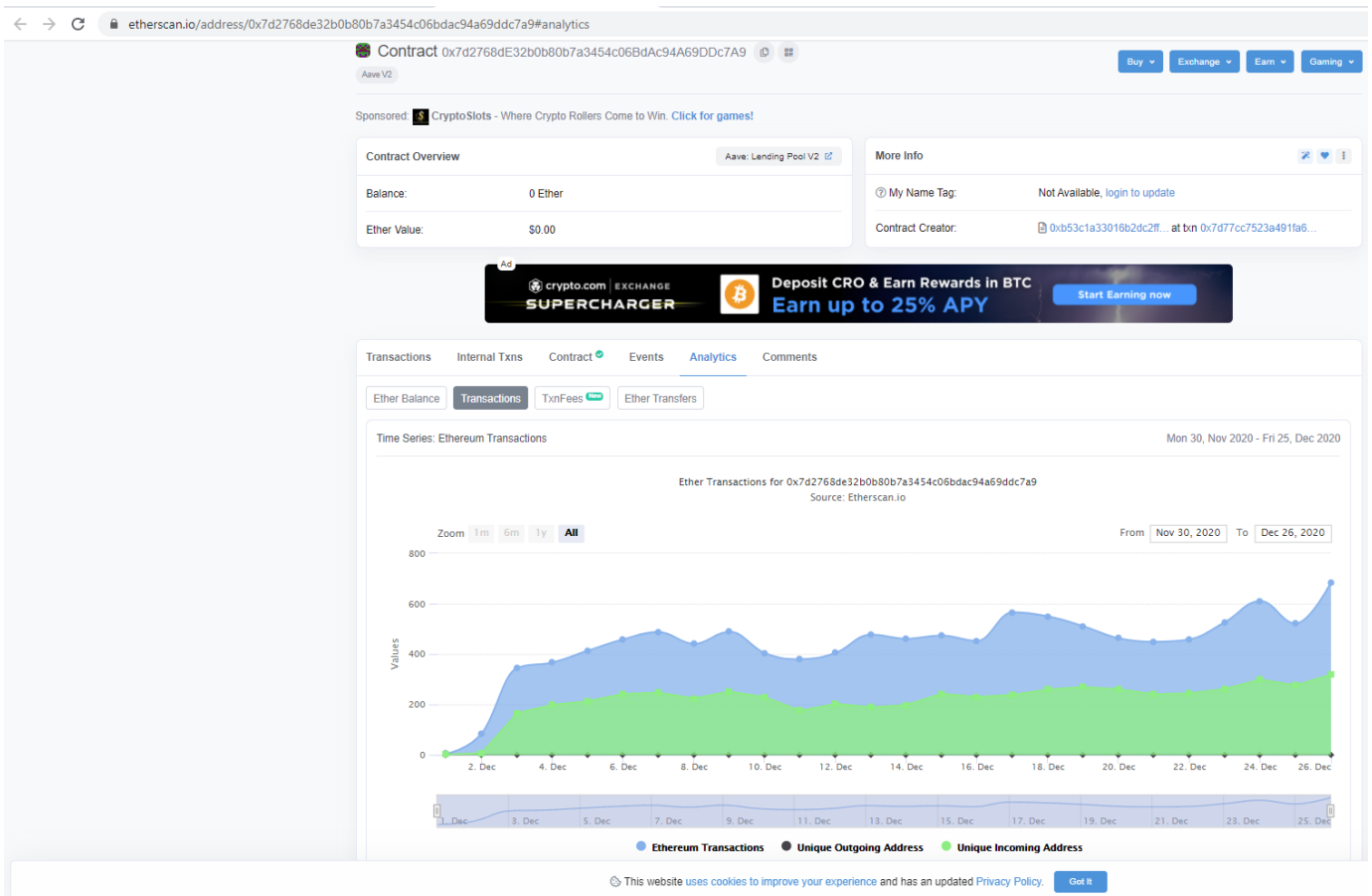
The Aave protocol is an ecosystem of multiple markets, with the first market being the main Aave Market.

Mainnet

Kovan

Contracts	Code	Address
LendingPoolAddressesProvider	Github	0xB53C1a33016B2DC
LendingPoolAddressesProviderRegistry	Github	0x52D306e36E3B6B0
LendingPool	Github	0x7d2768dE32b0b80
LendingPoolCollateralManager	Github	0xbd4765210d4167C
LendingPoolConfigurator	Github	0x311Bb771e4F8952
LendingRateOracle	-	0x8A32f49FFbA88ab
Price Oracle	-	0xA50ba011c48153D
Pool Admin	-	0xB9062896ec3A615
Emergency Admin	-	0xB9062896ec3A615
ProtocolDataProvider	Github	0x057835Ad21a177d

Code Used Appendix



Example Code Appendix

```

1  /**
2   * @dev Function is invoked by the proxy contract when the LendingPool co
3   * LendingPoolAddressesProvider of the market.
4   * - Caching the address of the LendingPoolAddressesProvider in order to
5   * on subsequent operations
6   * @param provider The address of the LendingPoolAddressesProvider
7   */
8  function initialize(ILendingPoolAddressesProvider provider) public initial
9     _addressesProvider = provider;
10 }
11
12 /**
13 * @dev Deposits an `amount` of underlying asset into the reserve, receiv
14 * - E.g. User deposits 100 USDC and gets in return 100 aUSDC
15 * @param asset The address of the underlying asset to deposit
16 * @param amount The amount to be deposited
17 * @param onBehalfOf The address that will receive the aTokens, same as m
18 * wants to receive them on his own wallet, or a different address if th
19 * is a different wallet
20 * @param referralCode Code used to register the integrator originating th
21 * 0 if the action is executed directly by the user, without any middle-
22 */
23 function deposit(
24     address asset,
25     uint256 amount,

```

```

26     address onBehalfOf,
27     uint16 referralCode
28 ) external override whenNotPaused {
29     DataTypes.ReserveData storage reserve = _reserves[asset];
30
31     ValidationLogic.validateDeposit(reserve, amount);
32
33     address aToken = reserve.aTokenAddress;
34
35     reserve.updateState();
36     reserve.updateInterestRates(asset, aToken, amount, 0);
37
38     IERC20(asset).safeTransferFrom(msg.sender, aToken, amount);
39
40     bool isFirstDeposit = IAToken(aToken).mint(onBehalfOf, amount, reserve.i
41
42     if (isFirstDeposit) {
43         _usersConfig[onBehalfOf].setUsingAsCollateral(reserve.id, true);
44         emit ReserveUsedAsCollateralEnabled(asset, onBehalfOf);
45     }
46
47     emit Deposit(asset, msg.sender, onBehalfOf, amount, referralCode);
48 }
49

```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	12	3489	491	897	2101	167

Comments to Code 897 / 2101 = 43%

Typescript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
TypeScript	30	7498	1416	741	5341	342

Tests to Code 5341/ 2101= 254%

