AAVE V2 Process Quality Review

Score: 96%

Overview

This is a AAVE Process Quality Review completed on July 19th 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 96%, 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 Smart Chain
Polygon
Avalanche

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.aave.com/developers/deployed-contracts/deployed-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 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 800 transactions a day on contract *InitializableImmutableAdminUpgradeabilityProxy*.sol, 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/aave/protocol-v2.

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 1630 commits and 40 branches in their protocol-v2, this is a robust software 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

Public team info found at 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 (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.aave.com/developers/.

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

Answer: Yes

The AAVE software functions (code) are all well-documented in "The Core Protocol" section of their documentation.

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

Answer: 100%

All the AAVE core protocols have their software functions documented here, as well as governance functions here, and API documentation here, and additional NPM documentation here.

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: 41%

Code examples are in the Appendix. As per the SLOC, there is 41% commenting to

code (CtC).

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

Note: The CtC of AAVE was calculated using the protocol-v2/contracts/protocol repository directory as well as *AaveOracle.sol,* adapter contracts, and other core contracts authored by the AAVE developers. All interface, library, dependencies, and mocks were not implemented within this calculation as they are not the protocol's executing contracts, and most of them come from third-party sources.

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 and explicit traceability between the documented software functions and their implementation within the AAVE source code. A good example of this can be seen here.

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 349% 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: 97%

There is evidence of AAVE code coverage in their SigmaPrime audit report, however they do not explain skips or misses. In addition, they also have 99% codecov in their Governance repository.

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

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

> Answer: Yes

Scrips/Instructions location: Instructions to run tests can be found in the README.

14) Report of the results (%)

> Answer: 100%

As well as their SigmaPrime coverage report, AAVE has their own report here, and

governance codecov report here.

Guidance:

100%	Detailed test report as described below
70%	GitHub Code coverage report visible
0%	No test report evident

15) Formal Verification test done (%)

> Answer: 100%

AAVE has had a Formal Verification test done by Certora.

16) Stress Testing environment (%)

> Answer: 100%

There is evidence of AAVE Kovan test-net smart contract usage at https://docs.aave.com/developers/deployed-contracts/deployed-contracts.

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%

Multiple high-quality AAVE audit reports were published before and after V1 and V2 deployment. The results were also implemented. These reports can be found at https://docs.aave.com/risk/audits/smartcontract-audits.

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, question

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: 60%

AAVE's Bug Bounty program offers up to 250k in rewards.

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%

AAVE admin access control information can easily be found at https://docs.aave.com/developers/protocol-governance/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: 90%

a) Some protocols are clearly labelled as immutable (i.e LendingPoolAddressesProvider), and others are clearly labelled as upgradeable.

b) Defined voting roles and structure are clearly outlined

c) Capabilities for change in contract can be found here.

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%

21) Is the information in non-technical terms that pertain to the investments (%)

> Answer: 90%

They have a technical and non-technical set of documentation.

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

22) Is there Pause Control documentation including records of tests (%)

> Answer: 100%

The AAVE Pause Control function is called Pause Guardian and is documented in the governance subgraph and tests from May 2021 can be found at https://github.com/aave/governance-v2/blob/f16655ae3d91d6043c5e345f59c0111d8207771b/test/governance-admin.spec.ts.

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

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

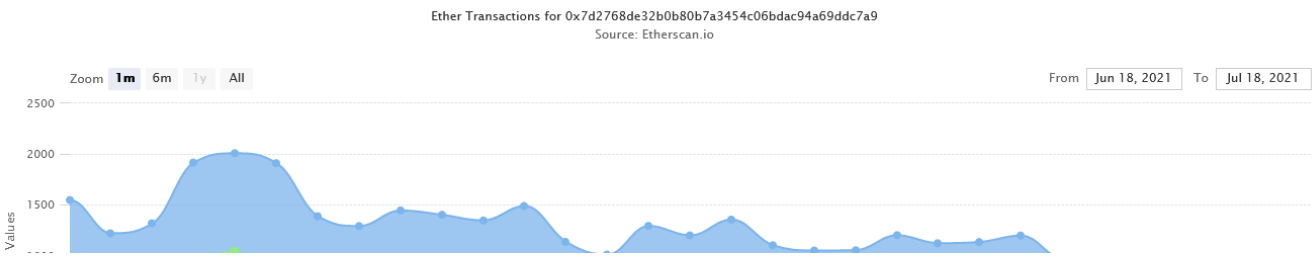
PQ Audit Scoring Matrix (v0.7)	Total Points	AAVE v2 Answer	AAVE v2 Points
Total	260		250
Code and Team			**96%**
2) Is the code actively being used? (%)	5	100%	5
3) Is there a public software repository? (Y/N)	5	Y	5
4) Is there a development history visible? (%)	5	100%	5
5) Is the team public (not anonymous)? (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	41%	2.05
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	99%	4.95
13) Scripts and instructions to run the tests? (Y/N)	5	Y	5
14) Report of the results (%)	10	100%	10
15) Formal Verification test done (%)	5	100%	5
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	60%	6
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	5	80%	4
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	100%	10

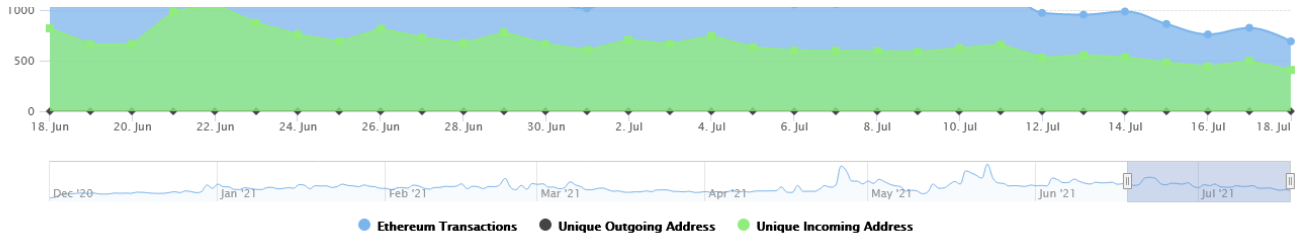
Section Scoring			
Code and Team	50	100%	
Documentation	45	93%	
Testing	50	100%	
Security	80	95%	
Access Controls	35	91%	

Executing Code Appendix

Contracts	Code	Address
LendingPoolAddressesProvider	Github	0xB53C1a33016B2DC2fF3653530bfF1848a51
LendingPoolAddressesProviderRegistry	Github	0x52D306e36E3B6B02c153d0266ff0f85d18BC
LendingPool	Github	0x7d2768dE32b0b80b7a3454c06BdAc94A69D
LendingPoolCollateralManager	Github	0xbd4765210d4167CE2A5b87280D9E8Ee316D
LendingPoolConfigurator	Github	0x311Bb771e4F8952E6Da169b425E7e92d6Ac
LendingRateOracle	-	0x8A32f49FFbA88aba6EFF96F45D8BD1D4b3f
Price Oracle	-	0xA50ba011c48153De246E5192C8f9258A2ba
Pool Admin	-	0xB9062896ec3A615a4e4444DF183F0531a77
Emergency Admin	-	0xB9062896ec3A615a4e4444DF183F0531a77
ProtocolDataProvider	Github	0x057835Ad21a177dbdd3090bB1CAE03EaCF
WETHGateway	Github	0xcc9a0B7c43DC2a5F023Bb9b738E45B0Ef6B
AaveCollector		0x464C71f6c2F760DdA6093dCB91C24c39e5d
IncentivesController	Github	0xd784927Ff2f95ba542BfC824c8a8a98F3495

Code Used Appendix





Example Code Appendix

```
contract LendingPool is VersionedInitializable, ILendingPool, LendingPool
  using SafeMath for uint256;
  using WadRayMath for uint256;
  using PercentageMath for uint256;
  using SafeERC20 for IERC20;

  uint256 public constant LENDINGPOOL_REVISION = 0x2;

  modifier whenNotPaused() {
    _whenNotPaused();
    _;
  }

  modifier onlyLendingPoolConfigurator() {
    _onlyLendingPoolConfigurator();
    _;
  }

  function _whenNotPaused() internal view {
    require(!_paused, Errors.LP_IS_PAUSED);
  }

  function _onlyLendingPoolConfigurator() internal view {
    require(
      _addressesProvider.getLendingPoolConfigurator() == msg.sender,
      Errors.LP_CALLER_NOT_LENDING_POOL_CONFIGURATOR
    );
  }

  function getRevision() internal pure override returns (uint256) {
    return LENDINGPOOL_REVISION;
  }

  /**
   * @dev Function is invoked by the proxy contract when the LendingPool
   * LendingPoolAddressesProvider of the market.
   * - Caching the address of the LendingPoolAddressesProvider in order t
   *   on subsequent operations
   * @param provider The address of the LendingPoolAddressesProvider
   **/
```

```
function initialize(ILendingPoolAddressesProvider provider) public init
  _addressesProvider = provider;
  _maxStableRateBorrowSizePercent = 2500;
  _flashLoanPremiumTotal = 9;
  _maxNumberOfReserves = 128;
}

/**
 * @dev Deposits an `amount` of underlying asset into the reserve, rece
 * - E.g. User deposits 100 USDC and gets in return 100 aUSDC
 * @param asset The address of the underlying asset to deposit
 * @param amount The amount to be deposited
 * @param onBehalfOf The address that will receive the aTokens, same as
 *   wants to receive them on his own wallet, or a different address if
 *   is a different wallet
 * @param referralCode Code used to register the integrator originating
 *   0 if the action is executed directly by the user, without any midd
 **/
function deposit(
  address asset,
  uint256 amount,
  address onBehalfOf,
  uint16 referralCode
) external override whenNotPaused {
  DataTypes.ReserveData storage reserve = _reserves[asset];

  ValidationLogic.validateDeposit(reserve, amount);

  address aToken = reserve.aTokenAddress;

  reserve.updateState();
  reserve.updateInterestRates(asset, aToken, amount, 0);

  IERC20(asset).safeTransferFrom(msg.sender, aToken, amount);

  bool isFirstDeposit = IAToken(aToken).mint(onBehalfOf, amount, reserv

  if (isFirstDeposit) {
    _usersConfig[onBehalfOf].setUsingAsCollateral(reserve.id, true);
    emit ReserveUsedAsCollateralEnabled(asset, onBehalfOf);
  }

  emit Deposit(asset, msg.sender, onBehalfOf, amount, referralCode);
}

/**
 * @dev Withdraws an `amount` of underlying asset from the reserve, bur
 * E.g. User has 100 aUSDC, calls withdraw() and receives 100 USDC, bur
 * @param asset The address of the underlying asset to withdraw
 * @param amount The underlying amount to be withdrawn
 *   - Send the value type(uint256).max in order to withdraw the whole
 * @param to Address that will receive the underlying, same as msg.send
```

```
 *   wants to receive it on his own wallet, or a different address if th
 *   different wallet
 * @return The final amount withdrawn
 **/
function withdraw(
  address asset,
  uint256 amount,
  address to
) external override whenNotPaused returns (uint256) {
  DataTypes.ReserveData storage reserve = _reserves[asset];

  address aToken = reserve.aTokenAddress;

  uint256 userBalance = IAToken(aToken).balanceOf(msg.sender);

  uint256 amountToWithdraw = amount;

  if (amount == type(uint256).max) {
    amountToWithdraw = userBalance;
  }

  ValidationLogic.validateWithdraw(
    asset,
    amountToWithdraw,
    userBalance,
    _reserves,
    _usersConfig[msg.sender],
    _reservesList,
    _reservesCount,
    _addressesProvider.getPriceOracle()
  );

  reserve.updateState();

  reserve.updateInterestRates(asset, aToken, 0, amountToWithdraw);

  if (amountToWithdraw == userBalance) {
    _usersConfig[msg.sender].setUsingAsCollateral(reserve.id, false);
    emit ReserveUsedAsCollateralDisabled(asset, msg.sender);
  }

  IAToken(aToken).burn(msg.sender, to, amountToWithdraw, reserve.liquid

  emit Withdraw(asset, msg.sender, to, amountToWithdraw);

  return amountToWithdraw;
}

/**
 * @dev Allows users to borrow a specific `amount` of the reserve underl
 * already deposited enough collateral, or he was given enough allowanc
 * corresponding debt token (StableDebtToken or VariableDebtToken)
```

```
   * - E.g. User borrows 100 USDC passing as `onBehalfOf` his own address
   *   and 100 stable/variable debt tokens, depending on the `interestRat
   * @param asset The address of the underlying asset to borrow
   * @param amount The amount to be borrowed
   * @param interestRateMode The interest rate mode at which the user wan
   * @param referralCode Code used to register the integrator originating
   *   0 if the action is executed directly by the user, without any midd
   * @param onBehalfOf Address of the user who will receive the debt. Sho
   * calling the function if he wants to borrow against his own collatera
   * if he has been given credit delegation allowance
   **/
  function borrow(
    address asset,
    uint256 amount,
    uint256 interestRateMode,
    uint16 referralCode,
    address onBehalfOf
  ) external override whenNotPaused {
    DataTypes.ReserveData storage reserve = _reserves[asset];

    _executeBorrow(
      ExecuteBorrowParams(
        asset,
        msg.sender,
        onBehalfOf,
        amount,
        interestRateMode,
        reserve.aTokenAddress,
        referralCode,
        true
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	24	5885	766	1501	3618	281

Comments to Code 1501/3618 = 41%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity

TypeScript	34	12208	2073	718	9417	391
JSON	10	3221	6	0	3215	0

Tests to Code 12632/3618 = 349%