

A CONSENSYS DILIGENCE AUDIT REPORT

Aave Token

Date	July 2020
Lead Auditor	Daniel Luca
Co-auditors	John Mardlin

1 Executive Summary

This report presents the results of our engagement with Aave to review Aave Token.

The review was conducted over 3 days, from July 6 to July 8 2020, by Daniel Luca and John Mardlin as part of an ongoing engagement between Aave and ConsenSys Diligence. A total of 4 person-days were spent.

During the first day, we became familiar with the source code, as well as ran tests and coverage without issues. We also reviewed EIP-2612 and EIP-1967, which describe a significant part of the token's functionality.

During the second day, we continued to manually review the code and tried to find inconsistencies between the EIP-2612 standard and the implementation. We found a discrepancy in implementation of EIP-2612, and checked with the EIP-2612 creator to ensure the specs are correct.

During the third day, we finalized the manual review, scanned the contracts with our common tools suite, and created a few constraints that were validated with the Mythx platform and we then put the report together.

On July 22nd we reviewed changes made to the system and updated this report accordingly.

2 Scope

Our review focused on the commit hash `86d4ef225a8e702b5ec8315eda3add186ff31f33`. The list of files in scope can be found in the [Appendix](#).

Following our initial review, the system was modified and reviewed again at commit hash `b5d7e540d0ce16c7f8ec6e7d0d59f09d5f32f056`.

2.1 Objectives

Together with the Aave team, we identified the following priorities for our review:

1. Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our [Smart Contract Best Practices](#), and the [Smart Contract Weakness Classification Registry](#).
3. Identify any replay attacks or inconsistencies in the EIP-2612 implementation.
4. Check the upgradeability pattern.

3 Document Change Log

Version	Date	Description
1.0	2020-07-08	Initial report
1.1	2020-07-23	Updated report with changes to code

4 Recommendations

The issues are presented in approximate order of priority from highest to lowest.

4.1 Pin the Solidity version to the latest stable 0.6.x Closed

Resolution

This has been addressed in the latest reviewed version of the code.

Description

Some of the contract files are not very specific about the Solidity version, which can cause a bit of a problem in the compilation step for some tools.

`VersionedInitializable` is only compilable with `Solidity 0.6.x` versions.

`code/contracts/utils/VersionedInitializable.sol:L1`

```
pragma solidity >=0.4.24 <0.7.0;
```

This is because of the `abstract` keyword which is not compatible with [lower versions of Solidity](#).

The new keyword `abstract` can be used to mark contracts as **abstract**. It has to be used if a contract does not implement all its functions. **Abstract** contracts cannot be created using the `new` operator, and it is not possible to generate bytecode for them during compilation.

Recommendation

Specify a fixed Solidity version for at least this contract.

4.2 Permit expiration can be set indefinite by using MAX_UINT Closed

Resolution

Addressed per the recommendation.

Description

In the `permit()` function, a deadline of `0` is treated as non-expiring.

code/contracts/token/AaveToken.sol:L114

```
require(expiration == 0 || block.timestamp <= expiration, "INVALID_EXPIRATIC
```

Recommendation

This extra check is unnecessary, the same can be achieved by setting the deadline to `MAX_UINT`. This would also be more consistent with the [Uniswap-V2 implementation](#) referenced in EIP-2612.

5 Issues

The issues are presented in approximate order of priority from highest to lowest.

5.1 Remove `nonce` argument from `permit` functions

Closed

Resolution

This has been addressed in latest reviewed version of the code.

Description

The [EIP-2612](#) specifies a way for a token owner to `approve` tokens for a spender without any gas costs for themselves. This is also a good way to allow a 3rd party to enable `approve` before a `transferFrom`, in the same transaction.

The standard specifies a new `permit` function that looks like this:

```
function permit(
    address owner,
    address spender,
    uint256 value,
    uint256 deadline,
    uint8 v,
    bytes32 r,
    bytes32 s
)
```

The function in the standard does not have a `nonce` argument and as [clarified by the standard creator](#), the nonce does not need to be specified, as it can be used from the contract storage.

However, the current `permit` implementation does contain that `nonce`

code/contracts/token/AaveToken.sol:L92-L101

```
function permit(
    address owner,
    address spender,
    uint256 nonce,
    uint256 expiration,
    uint256 amount,
    uint8 v,
    bytes32 r,
    bytes32 s
) external {
```

In order to match the EIP-2612 standard, the `permit` function needs to be changed in the following manner:

- remove the `nonce` argument in the function definition
- remove the `require` which checks if the provided **nonce** matches the **nonce** in the contract storage
- to generate the digest, use the **nonce** currently available in the contract storage
- if the signature is valid, increment the **nonce** in the contract storage.

Recommendation

Remove the `nonce` argument and make the necessary changes in the code and the matching tests to match the EIP-2612 spec.

Appendix 1 - Files in Scope

This audit covered the following files)

File	git hash-object
contracts/token/AaveToken.sol	4c5dc5478a50da52d5b530f8876368560ba8511c
contracts/token/LendToAaveMigrator.sol	e316261f318659c6af36bed651ae52bb026f0c49
contracts/utils/VersionedInitializable.sol	e5a8b87b8f89b6c5f28b25be9f7499d14b5b6ff3

Appendix 2 - Artifacts

This section contains some of the artifacts generated during our review by automated tools, the test suite, etc. **If any issues or recommendations were identified by the output presented here, they have been addressed in the appropriate section above.**

A.2.1 MythX

MythX is a security analysis API for Ethereum smart contracts. It performs multiple types of analysis, including fuzzing and symbolic execution, to detect many common vulnerability types. The tool was used for automated vulnerability discovery for all audited contracts and libraries. More details on MythX can be found at mythx.io.

Below is the raw output of the MythX vulnerability scan for each contract:

AaveToken

Report for token/AaveToken.sol

Line	SWC Title	Severity	Short Description
110	Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
140	Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
47	DoS With Block Gas Limit	Low	Implicit loop over unbounded data structure.

Report for open-zeppelin/ERC20.sol

Line	SWC Title	Severity	Short Description
306	Presence of unused variables	Low	Unused function parameter "from".
306	Presence of unused variables	Low	Unused function parameter "to".
306	Presence of unused variables	Low	Unused function parameter "amount".

Report for utils/VersionedInitializable.sol

Title	Severity	Short Description	Line	SWC
Presence of unused variables	Low	Unused state variable “__gap”.	43	

Report for utils/VersionedInitializable.sol

Title	Severity	Short Description	Line	SWC
Presence of unused variables	Low	Unused state variable “__gap”.	43	

VersionedInitializable

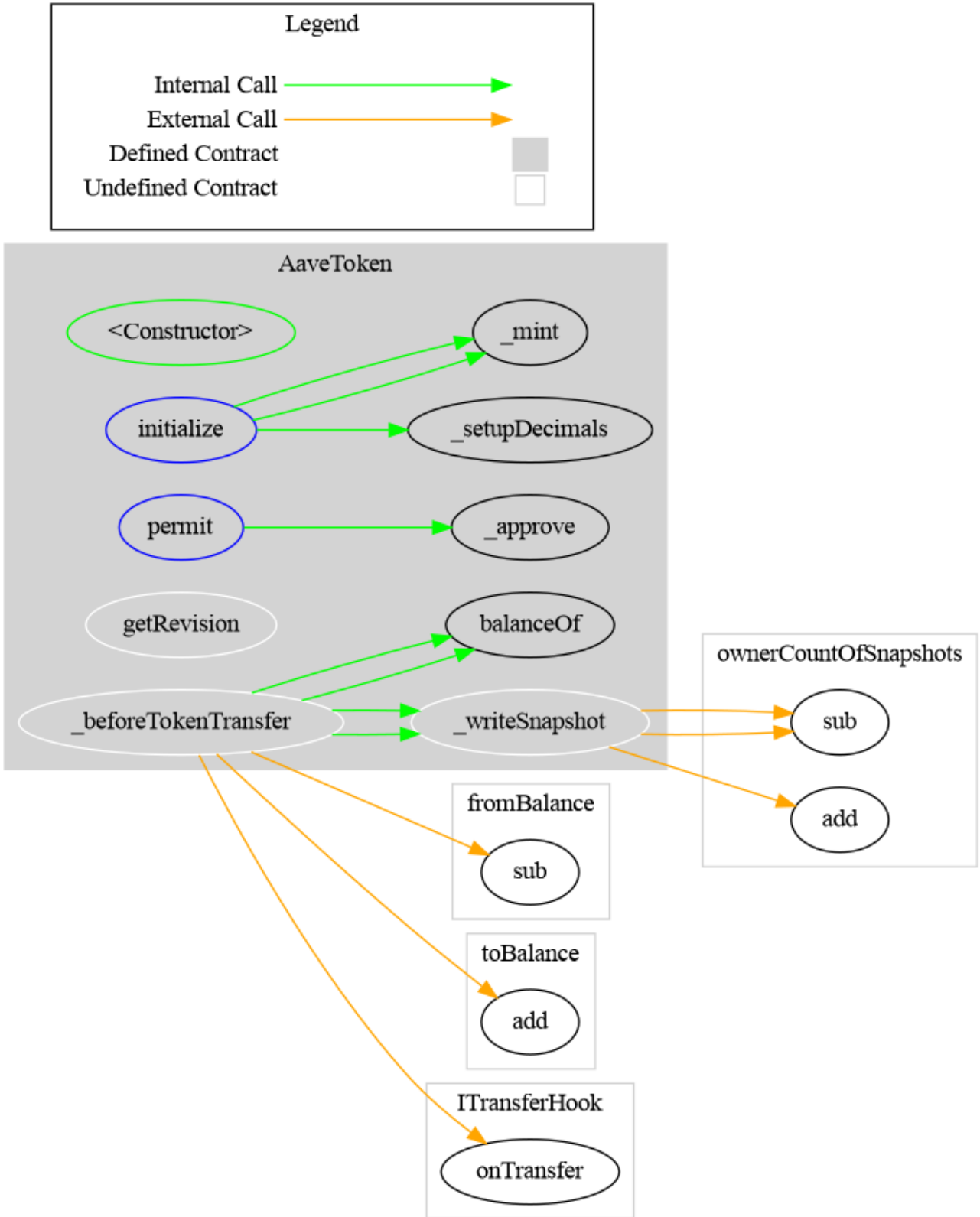
It is an abstract contract; the implementation is used by `AaveToken`.

A.2.2 Surya

Surya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.













Below is a complete list of functions with their visibility and modifiers:

AaveToken





Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers

Contract	Type	Bases		
AaveToken	Implementation	ERC20, VersionedInitializable		
L	<Constructor>	Public 		ERC20
L	initialize	External 		initializer
L	permit	External 		NO 
L	getRevision	Internal 		override
L	_writeSnapshot	Internal 		
L	_beforeTokenTransfer	Internal 		override

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

A.2.3 Tests Suite

The tests are comprehensive and cover all of the execution branches.

Below is the output generated by running the test suite:

```
$ npm test

> aave-token@1.0.0 test /home/daniel/Development/github.com/ConsenSys/aave-t
> buidler test

Compiling...

contracts/interfaces/IERC20.sol: Warning: SPDX license identifier not provic
```

contracts/interfaces/ITransferHook.sol: Warning: SPDX license identifier not

contracts/open-zeppelin/Address.sol: Warning: SPDX license identifier not pr

contracts/open-zeppelin/BaseAdminUpgradeabilityProxy.sol: Warning: SPDX lice

contracts/open-zeppelin/BaseUpgradeabilityProxy.sol: Warning: SPDX license i

contracts/open-zeppelin/Proxy.sol: Warning: SPDX license identifier not prov

contracts/open-zeppelin/SafeMath.sol: Warning: SPDX license identifier not p

contracts/open-zeppelin/UpgradeabilityProxy.sol: Warning: SPDX license ident

contracts/utils/DoubleTransferHelper.sol: Warning: SPDX license identifier r

contracts/utils/MockTransferHook.sol: Warning: SPDX license identifier not p

contracts/utils/VersionedInitializable.sol: Warning: SPDX license identifier

contracts/open-zeppelin/BaseAdminUpgradeabilityProxy.sol:13:1: Warning: This contract BaseAdminUpgradeabilityProxy is BaseUpgradeabilityProxy {
^ (Relevant source part starts here and spans across multiple lines).

contracts/open-zeppelin/Proxy.sol:15:3: The payable fallback function is def
fallback () payable external {
^ (Relevant source part starts here and spans across multiple lines).

contracts/open-zeppelin/InitializableUpgradeabilityProxy.sol:11:1: Warning:
contract InitializableUpgradeabilityProxy is BaseUpgradeabilityProxy {
^ (Relevant source part starts here and spans across multiple lines)

```

^ (Relevant source part starts here and spans across multiple lines).
contracts/open-zeppelin/Proxy.sol:15:3: The payable fallback function is def
  fallback () payable external {
    ^ (Relevant source part starts here and spans across multiple lines).

contracts/open-zeppelin/InitializableAdminUpgradeabilityProxy.sol:12:1: Warr
contract InitializableAdminUpgradeabilityProxy is BaseAdminUpgradeabilityPro
^ (Relevant source part starts here and spans across multiple lines).
contracts/open-zeppelin/Proxy.sol:15:3: The payable fallback function is def
  fallback () payable external {
    ^ (Relevant source part starts here and spans across multiple lines).

contracts/utils/MockTransferHook.sol:8:25: Warning: Unused function paramete
  function onTransfer(address from, address to, uint256 amount) external c
          ^-----^

contracts/utils/MockTransferHook.sol:8:39: Warning: Unused function paramete
  function onTransfer(address from, address to, uint256 amount) external c
          ^-----^

contracts/utils/MockTransferHook.sol:8:51: Warning: Unused function paramete
  function onTransfer(address from, address to, uint256 amount) external c
          ^-----^

Compiled 19 contracts successfully

-> Deploying test environment...
WARNING: Multiple definitions for initialize
WARNING: Multiple definitions for initialize
setup: 359.179ms

*****
Setup and snapshot finished
*****

AAVE token
  ✓ Checks initial configuration
  ✓ Checks the domain separator
  ✓ Checks the revision
  ✓ Checks the allocation of the initial AAVE supply
WARNING: Multiple definitions for initialize
  ✓ Starts the migration
  ✓ Checks the snapshots emitted after the initial allocation
  ✓ Record correctly snapshot on migration (66ms)

```

- ✓ Record correctly snapshot on transfer (48ms)
- ✓ Submits a permit with 0 expiration (38ms)
- ✓ Cancels the previous permit
- ✓ Tries to submit a permit with invalid nonce
- ✓ Tries to submit a permit with invalid expiration (previous to the current)
- ✓ Tries to submit a permit with invalid signature
- ✓ Tries to submit a permit with invalid owner
- ✓ Correct snapshotting on double action in the same block (100ms)
- ✓ Emits correctly mock event of the `_beforeTokenTransfer` hook

LEND migrator

- ✓ Check the constructor is executed properly
- ✓ Check migration isn't started

WARNING: Multiple definitions for initialize

- ✓ Starts the migration
- ✓ Migrates 1000 LEND (70ms)

20 passing (1s)

Even though it seems like there isn't 100% coverage, the unexplored branch in the tests is actually not reachable in that specific case.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered
interfaces/	100	100	100	100	
IERC20.sol	100	100	100	100	
IERC20Detailed.sol	100	100	100	100	
ITransferHook.sol	100	100	100	100	
token/	100	88.89	100	100	
AaveToken.sol	100	87.5	100	100	
LendToAaveMigrator.sol	100	100	100	100	
utils/	100	50	100	100	
DoubleTransferHelper.sol	100	100	100	100	
MintableErc20.sol	100	100	100	100	
MockTransferHook.sol	100	100	100	100	
VersionedInitializable.sol	100	50	100	100	
All files	100	85	100	100	

Appendix 3 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your

reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.