Aave Safety Module

Date	September 2020
Lead Auditor	Bernhard Mueller
Co-auditors	Sergii Kravchenko

1 Executive Summary

This report presents the results of our engagement with Aave to review the Aave incentives smart contracts. The review was conducted over one week, from September 7th to September 11th by Sergii Kravchenko and Bernhard Mueller. A total of 10 person-days were spent.

2 Scope

Our review focused on the commit hash b125e181762a0318ce7dc1c9eed1b75d0520e343. The list of files in scope can be found in the Appendix.

The hash of the final commit with all fixes merged was a058e0e4443b775f403ee49062e304e7d857e07e.

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 ways of gaming or abusing the system, e.g. to drain funds from the rewards contract.

We specifically focused on identifying possible ways of gaming the staking, rewards distribution and cooldown mechanisms, e.g.:

- Ways of claiming rewards without any time passing between stake and unstake;
- Incorrect updates of internal accounting (e.g. user can claim the same reward twice by transferring stakedTokens, etc.)l
- Abusing or bypassing the cooldown.

3 Results Overview

We did not identify any critical issues that would have allowed to game the system.

It was noted that the cooldown mechanism for unstaking can be manipulated to a certain degree: A user could shorten the coolodown by splitting their stake into chunks and starting the cooldown at different points in time. However, while the mechanism isn’t perfect, it still achieves its goal of preventing users from unstaking if they anticipate being slashed. The effect of the mechanism can be improved by choosing appropriate values for the cooldowbn and unstaking window (see “issues” section).

Besides the above, only minor code quality issues and best practive violations were found.

3.1 Recommendations

Review the issues and code quality recommendations documented in this report.

4 Issues

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

4.1 Unhandled return values of transfer and transferFrom

Medium ✓ Fixed

Resolution

The issue was fixed by using OpenZeppelin's `safeTransfer` and `safeTransferFrom` wrappers.

ERC20 implementations are not always consistent. Some implementations of `transfer` and `transferFrom` could return 'false' on failure instead of reverting. It is safer to wrap such calls into `require()` statements to these failures.

code/contracts/stake/StakedToken.sol:L92

```
IERC20(STAKED_TOKEN).transferFrom(msg.sender, address(this), amount);
```

code/contracts/stake/StakedToken.sol:L156

```
REWARD_TOKEN.transferFrom(REWARDS_VAULT, to, amountToWithdraw);
```

code/contracts/stake/StakedToken.sol:L125

```
IERC20(STAKED_TOKEN).transfer(to, amount);
```

4.2 Staking cooldown can be avoided for a part of the funds Minor ✓ Fixed

Resolution

The cooldown window will be set to much higher value (to the order of days) in production. The mechanism is sufficient to prevent stakers from withdrawing if the cooldown window is long enough while also being larger than the withdrawal window.

Aave is planning to introduce a slashing mechanism for the staking system in the future. In order to prevent stakers from withdrawing their stake immediately, the team has added a “cooldown” mechanism. The idea is that whenever stakers want to redeem the stake, they should call the `cooldown` function and wait for `COOLDOWN_SECONDS`. After that, a time period called `UNSTAKE_WINDOW` starts during which the stake can be withdrawn.

However, depending on the settings (“COOLDOWN_SECONDS” and “UNSTAKE_WINDOW” values), various algorithms exist that would allow users to optimize their withdrawal tactics. By using such tactics, stakers may be able to withdraw at least a part of the stake immediately.

Let’s assume that the values are the same as in tests: `COOLDOWN_SECONDS == 1 hour` and `UNSTAKE_WINDOW == 30 minutes`. Stakers can split their stake into 3 parts and call `cooldown` for one of them every 30 minutes. That would ensure that at least ⅓ of the stake can be withdrawn immediately at any time. And on average, more than ½ of the stake can be withdrawn immediately.

Remediation:

Make sure that the `COOLDOWN_SECONDS` value is much larger than the `UNSTAKE_WINDOW`. This will make any cooldown optimization techniques less effective.

4.3 Minor code quality issues Minor ✓ Fixed

Resolution

all issues have been fixed in production.

We recommend the following improvements:

Fix todos

Clean up all TODOs before going into production:

code/contracts/stake/AaveDistributionManager.sol:L44-L46

```
function configureAssets(DistributionTypes.AssetConfigInput[] calldata asset
  external
//    override TODO: create interface
```

Fix incorrect NatSpec comments

Clean up NatSpec comments to improve readability.

The function `claimRewards()` in `StakedToken` has the same description as the `stake()` function:

code/contracts/stake/StakedToken.sol:L141-L145

```
 * @dev Stakes tokens to start earning rewards
 * @param to Address to stake for
 * @param amount Amount to stake
 **/
function claimRewards(address to, uint256 amount) external override {
```

One function argument is missing from the docstrings for `claimRewards()` in `AaveIncentivesController`:

code/contracts/stake/AaveIncentivesController.sol:L97-L107

```
/**
 * @dev Claims reward for an user, on all the assets of the lending pool, accu
 * @param amount Amount of rewards to claim
 * @param to Address that will be receiving the rewards
 * @return Rewards claimed
 **/
function claimRewards(
  uint256 amount,
  address to,
  bool stake
) external override returns (uint256) {
```

Appendix 1 - Files in Scope

This audit covered the following files:

File Name	SHA-1 Hash
code/contracts/stake/AaveDistributionManager.sol	d4538fad03eb23e1cb1c8d2ec5678066adb85182
code/contracts/stake/AaveIncentivesController.sol	ae4e31b00899767366ffaf8245733f8b1e78a9cc
code/contracts/stake/StakedAave.sol	f92b21fb160280e01f0a011e56e85a511da67b5c
code/contracts/stake/StakedToken.sol	58efb6d8aaee835bbd2f0879acd1d089d4b6b02b

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 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:

Contract	**Type**	**Bases**		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
AaveDistributionManager	Implementation			
L		Public ❗	🛑	NO❗
L	configureAssets	External ❗	🛑	NO❗
L	_updateAssetStateInternal	Internal 🔒	🛑	
L	_updateUserAssetInternal	Internal 🔒	🛑	
L	_claimRewards	Internal 🔒	🛑	
L	_getUnclaimedRewards	Internal 🔒		
L	_getRewards	Internal 🔒		
L	_getAssetIndex	Internal 🔒		
L	getUserAssetData	Public ❗		NO❗

Contract	Type	Bases		
AaveIncentivesController	Implementation	IAaveIncentivesController, VersionedInitializable, AaveDistributionManager		
L		Public ❗	🛑	AaveDistributionManager
L	initialize	External ❗	🛑	initializer
L	handleAction	External ❗	🛑	NO❗
L	getRewardsBalance	External ❗		NO❗
L	claimRewards	External ❗	🛑	NO❗
L	getUserUnclaimedRewards	External ❗		NO❗
L	getRevision	Internal 🔒		
StakedAave	Implementation	StakedToken		
L		Public ❗	🛑	StakedToken

Contract	Type	Bases		
StakedToken	Implementation	IStakedAave, ERC20WithSnapshot, VersionedInitializable, AaveDistributionManager		
L		Public ❗	🛑	ERC20WithSnapshot AaveDistributionManager
L	initialize	External ❗	🛑	initializer
L	stake	External ❗	🛑	NO❗
L	redeem	External ❗	🛑	NO❗
L	cooldown	External ❗	🛑	NO❗
L	claimRewards	External ❗	🛑	NO❗
L	_transfer	Internal 🔒	🛑	
L	_updateCurrentUnclaimedRewards	Internal 🔒	🛑	
L	_getNextCooldownTimestamp	Internal 🔒	🛑	
L	getTotalRewardsBalance	External ❗		NO❗
L	getRevision	Internal 🔒		

A.2.2 Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

A.2.3 Tests Suite

Below is the output generated by running the test suite:

```
AaveIncentivesController claimRewards tests
   ✓ Accrued rewards are 0, claim 0 (181ms)
   ✓ Accrued rewards are 0, claim not 0 (148ms)
   ✓ Accrued rewards are not 0 (156ms)
   ✓ Should allow -1 (160ms)
   ✓ Should add extra premium on withdrawal to stake (179ms)
   ✓ Should withdraw everything if amountToClaim more then rewards balance
   ✓ Should withdraw to another user (116ms)
   ✓ Should withdraw to another user and stake (144ms)

 AaveIncentivesController configureAssets
   ✓ Tries to submit config updates not from emission manager
   ✓ Submit initial config for the assets (41ms)
   ✓ Submit updated config for the assets (57ms)
   ✓ Indexes should change if emission are set not to 0, and pool has depo
   ✓ Indexes should cumulate rewards if next emission is 0 (43ms)
   ✓ Indexes should not change if no emission (39ms)
   ✓ Should go to the limit if distribution ended (40ms)
   ✓ Should not accrue any rewards after end or distribution (41ms)

 AaveIncentivesController constructor tests
   ✓ should assign correct params (127ms)

 AaveIncentivesController getRewardsBalance tests
   ✓ Accrued rewards are 0 (138ms)
   ✓ Accrued rewards are not 0 (110ms)
   ✓ Accrued rewards are not 0 (102ms)

 AaveIncentivesController handleAction tests
   ✓ should revert if it's not lending pool
   ✓ All 0 (75ms)
   ✓ Accrued rewards are 0, 0 emission (65ms)
   ✓ Accrued rewards are 0, 0 user balance (68ms)
   ✓ 1. Accrued rewards are not 0 (59ms)
   ✓ 2. Accrued rewards are not 0 (73ms)
```

```
  AaveIncentivesController initialize
    ✓ Tries to call initialize second time, should be reverted
    ✓ allowance on aave token should be granted to psm contract for pei

  StakedAave. Basics
    ✓ Initial configuration after initialize() is correct (47ms)
    ✓ Reverts trying to stake 0 amount
    ✓ User 1 stakes 50 AAVE: receives 50 SAAVE, StakedAave balance of AAVE i
    ✓ User 1 stakes 20 AAVE more: his total SAAVE balance increases, StakedA
    ✓ User 1 claim half rewards
    ✓ User 1 tries to claim higher reward than current rewards balance
    ✓ User 1 claim all rewards (46ms)
    ✓ User 6 stakes 50 AAVE, with the rewards not enabled (86ms)
    ✓ User 6 stakes 30 AAVE more, with the rewards not enabled (80ms)
    ✓ Validates staker cooldown with stake() while being on valid unstake wi

  StakedAave. Redeem
    ✓ Reverts trying to redeem 0 amount
    ✓ User 1 stakes 50 AAVE
    ✓ User 1 tries to redeem without activating the cooldown first
    ✓ User 1 activates the cooldown, but is not able to redeem before the CO
    ✓ User 1 activates the cooldown again, and tries to redeem a bigger amou
    ✓ User 1 activates the cooldown again, and redeems within the unstake pe
    ✓ User 4 stakes 50 AAVE, activates the cooldown and redeems half of the
    ✓ User 5 stakes 50 AAVE, activates the cooldown and redeems with rewards

  StakedAave. Transfers
    ✓ User 1 stakes 50 AAVE (96ms)
    ✓ User 1 transfers 50 SAAVE to User 5 (183ms)
    ✓ User 5 transfers 50 SAAVE to himself (148ms)
    ✓ User 5 transfers 50 SAAVE to user 2, with rewards not enabled (213ms)
    ✓ User 4 stakes and transfers 50 SAAVE to user 2, with rewards not enabl
    ✓ Activate cooldown of User2, transfer entire amount from User2 to User3
    ✓ Transfer balance from User 3 to user 2 cooldown  of User 2 should be r
    ✓ Transfer balance from User 3 to user 2, cooldown of User 2 should be t


  54 passing (7s)
```

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.