Aave

# Aave v3.0.1 Contract Review

*Version: 2.2*

**December, 2022**

# Contents


**Introduction** **2**
- Disclaimer . . . 2
- Document Structure . . . 2
- Overview . . . 2

**Security Assessment Summary** **3**
- Findings Summary . . . 3

**Detailed Findings** **4**

**Summary of Findings** **5**
- `Mint` Event Is Emitted When Zero Value Is Minted . . . 6
- Miscellaneous General Comments . . . 8

**A Test Suite** **10**

**B Vulnerability Severity Classification** **13**

# Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Aave smart contracts. The review focused solely on the security aspects of the Solidity implementation of the contract, though general recommendations and informational comments are also provided.

## Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the smart contract. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

## Document Structure

The first section provides an overview of the functionality of the Aave smart contracts contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

Outputs of automated testing that were developed during this assessment are also included for reference (in the Appendix: Test Suite).

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Aave smart contracts.

## Overview

Aave `v3.0.1` is an update to the aave-v3-core repository. The following are a list of changes implemented in `v3.0.1`:

- ATokens transfer events have been updated to be ERC20 compliant.
- Patch to prevent reentrancy on withrdrawals and liquidations, only applies to reentrant tokens (e.g. ERC777).
- Allows borrows and repays to occur in the same block.
- Adds a return value for `backUnbacked()`.
- Adds a configuration parameter to a reserve to disable flash loans for a collateral.
- Fix edge case for liquidations if protocol fee is greater than user balance.
- Patches a bug when reserve factor is 100%.
- Optimises state updates if the timestamp is unchanged.
- AToken function `handleRepayment()` adds the `onBehlafOf` parameter.
- Unbacked tokens are always accounted for when calculating interest rates.

# Security Assessment Summary

This review was conducted on the files hosted on the aave-v3-core repository and assessed updates from v3.0.0 to v3.0.1 as seen in PR #701 @ 428e258.

*Note: the OpenZeppelin libraries and dependencies were excluded from the scope of this assessment.*

The manual code review section of the report is focused on identifying any and all issues/vulnerabilities associated with the business logic implementation of the contracts. This includes their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout). Additionally, the manual review process focused on all known Solidity anti-patterns and attack vectors. These include, but are not limited to, the following vectors: re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers. For a more thorough, but non-exhaustive list of examined vectors, see [1, 2].

To support this review, the testing team used the following automated testing tools:

- Mythril: `https://github.com/ConsenSys/mythril`
- Slither: `https://github.com/trailofbits/slither`
- Surya: `https://github.com/ConsenSys/surya`

Output for these automated tools is available upon request.

## Findings Summary

The testing team identified a total of 2 issues during this assessment. Categorised by their severity:

- Informational: 2 issues.

# Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Aave smart contracts. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the contracts, including gas optimisations, are also described in this section and are labelled as “informational”.

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected contract(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

# Summary of Findings

| ID | Description | Severity | Status |
|---|---|---|---|
| AV301-01 | `Mint` Event Is Emitted When Zero Value Is Minted | Informational | **Resolved** |
| AV301-02 | Miscellaneous General Comments | Informational | **Resolved** |

| AV301-01 | `Mint` Event Is Emitted When Zero Value Is Minted |
|---|---|
| Asset | `ScaledBalanceTokenBase.sol` |
| Status | **Resolved:** See Resolution |
| Rating | Informational |

## Description

Scaled balance tokens such as `aTokens` accrue interest and increase in value. When these tokens are transferred both the sender's and receiver's balances are checked for a balance increase. `Mint` events are always emitted for the sender and only for the receiver if there has been a balance increase.

```
138 function _transfer(
    address sender,
140 address recipient,
    uint256 amount,
142 uint256 index
    ) internal {
144 uint256 senderScaledBalance = super.balanceOf(sender);
    uint256 senderBalanceIncrease = senderScaledBalance.rayMul(index) -
146     senderScaledBalance.rayMul(_userState[sender].additionalData);
    uint256 recipientScaledBalance = super.balanceOf(recipient);
148 uint256 recipientBalanceIncrease = recipientScaledBalance.rayMul(index) -
        recipientScaledBalance.rayMul(_userState[recipient].additionalData);
150 _userState[sender].additionalData = index.toUint128();
    _userState[recipient].additionalData = index.toUint128();
152
    super._transfer(sender, recipient, amount.rayDiv(index).toUint128());
154
    emit Transfer(address(0), sender, senderBalanceIncrease);
156 emit Mint(_msgSender(), sender, senderBalanceIncrease, senderBalanceIncrease, index);

158 if (recipientBalanceIncrease > 0) {
        emit Transfer(address(0), recipient, recipientBalanceIncrease);
160     emit Mint(_msgSender(), recipient, recipientBalanceIncrease, recipientBalanceIncrease, index);
    }
```

It is expected for the case where `senderBalanceIncrease = 0` to occur if a sender makes multiple actions in the same block. One example is making two transfers as the first transfer will update the `_userState[sender].additionalData` to the current `index`. The second transfer will then have `_userState[sender].additionalData = index` and hence `senderBalanceIncrease` will be zero.

The impact of the issue is informational. Emitting zero value events will increase gas costs and increase the load of processing events off-chain. There are no funds at risk and the events will always display the correct values.

## Recommendations

The issue may be resolved by only emitting the `Mint` event if the `senderBalanceIncrease` is non-zero.

### Resolution

The issue is resolved in PR #745 by adding the following check.

```
if (senderBalanceIncrease > 0) {
    emit Transfer(address(0), sender, senderBalanceIncrease);
    emit Mint(_msgSender(), sender, senderBalanceIncrease, senderBalanceIncrease, index);
}
```

| AV301-02 | Miscellaneous General Comments |
|---|---|
| Asset | `contracts/*` |
| Status | **Resolved:** See Resolution |
| Rating | Informational |

## Description

This section details miscellaneous findings discovered by the testing team that do not have direct security implications:

1. **`ReserveCache.currLiquidityIndex` is no longer necessary.**
   The variable `currLiquidityIndex` is cached in `ReserveCache` to help minimise storage load operations. This variable is only ever used in `_updateIndexes()`. It is safe to remove this variable and instead use `nextLiquidityIndex` during `_updateIndexes()`.
2. **Camel Case inconsistencies in the use of `flashLoan` and `flashloan`.**
   Consider updating the the following types to have an upper-case "L":
   - `ValidationLogic.validateFlashloanSimple()`
   - `DataTypes.FlashloanSimpleParams`
   - `ValidationLogic.validateFlashloan()`
   - `DataTypes.FlashloanParams`
   - `Pool.updateFlashloanPremiums()` and `IPool.updateFlashloanPremiumTotal()`
   - `IPoolConfigurator.FlashloanPremiumTotalUpdated` event and parameters `oldFlashloanPremiumTotal` and `newFlashloanPremiumTotal`
   - `IPoolConfigurator.FlashloanPremiumToProtocolUpdated` event and parameters `oldFlashloanPremiumTotal` and `newFlashloanPremiumTotal`
   - `IPoolConfigurator` and `PoolConfigurator` function `updateFlashloanPremiumTotal()` function (also parameters and local variables for this function)
   - `IPoolConfigurator` and `PoolConfigurator` function `updateFlashloanPremiumToProtocol()` function (also parameters and local variables for this function)

## Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

## Resolution

The development team have opted not to remove `currLiquidityIndex`. The reason is that the current logic matches `currVariableBorrowIndex` and `nextVariableBorrowIndex`, modifying `currLiquidityIndex` will cause inconsistencies between these two variables and complicate the code base.

The discrepancies between `flashloan` and `flashLoan` require changes to interfaces which impact users of the smart contracts. Changes will not be made in a minor patch and will be implemented in a future update.

# Appendix A Test Suite

A non-exhaustive list of tests were constructed to aid this security review and are provided alongside this document. The `brownie` framework was used to perform these tests and the output is given below.

```
test_borrow                                          PASSED  [0%]
test_borrow_isolation_mode                           PASSED  [1%]
test_repay                                           PASSED  [2%]
test_repay_isolation_mode                            PASSED  [2%]
test_rebalance_stable_borrow_rate                    PASSED  [3%]
test_swap_borrow_rate_mode_stable                    PASSED  [4%]
test_swap_borrow_rate_mode_variable                  PASSED  [4%]
test_mint_unbacked                                   PASSED  [5%]
test_mint_unbacked_supply_and_borrows                PASSED  [6%]
test_mint_unbacked_cap                               PASSED  [6%]
test_mint_unbacked_zero                              PASSED  [7%]
test_only_bridge_modifier                            PASSED  [8%]
test_back_unbacked                                   PASSED  [8%]
test_back_unbacked_erc20_fails                       PASSED  [9%]
test_set_user_emode                                  PASSED  [10%]
test_flashloan_repay                                 PASSED  [11%]
test_flashloan_borrow                                PASSED  [11%]
test_flashloan_simple                                PASSED  [12%]
test_liquidation_call                                PASSED  [13%]
test_linear_interest                                 PASSED  [13%]
test_linear_interest_one_year                        PASSED  [14%]
test_linear_interest_max_values                      PASSED  [15%]
test_linear_interest_zero                            PASSED  [15%]
test_compound_interest                               PASSED  [16%]
test_compound_interest_overflows                     PASSED  [17%]
test_compound_interest_zero                          PASSED  [17%]
test_percent_mul                                     PASSED  [18%]
test_percent_mul_zero                                PASSED  [19%]
test_percent_mul_overflow                            PASSED  [20%]
test_percent_div                                     PASSED  [20%]
test_percent_div_zero                                PASSED  [21%]
test_percent_div_overflow                            PASSED  [22%]
test_init_reserves                                   PASSED  [22%]
test_drop_reserve                                    PASSED  [23%]
test_update_atoken                                   PASSED  [24%]
test_update_stable_debt_token                        PASSED  [24%]
test_update_variable_debt_token                      PASSED  [25%]
test_set_reserve_borrowing                           PASSED  [26%]
test_configure_reserve_as_collateral                 PASSED  [26%]
test_configure_reserve_as_collateral_large_ltv       PASSED  [27%]
test_configure_reserve_as_collateral_non_zero_bonus  PASSED  [28%]
test_configure_reserve_as_collateral_excessive_bonus PASSED  [28%]
test_configure_reserve_as_collateral_small_bonus     PASSED  [29%]
test_configure_reserve_as_collateral_supply          PASSED  [30%]
test_set_reserve_stable_rate_borrowing               PASSED  [31%]
test_set_reserve_active                              PASSED  [31%]
test_set_reserve_active_atokens                      PASSED  [32%]
test_set_reserve_freeze                              PASSED  [33%]
test_set_borrowable_in_isolation                     PASSED  [33%]
test_set_reserve_pause                               PASSED  [34%]
test_set_reserve_factor                              PASSED  [35%]
test_set_debt_ceiling                                PASSED  [35%]
test_set_borrow_cap                                  PASSED  [36%]
test_set_supply_cap                                  PASSED  [37%]
test_set_liquidation_protocol_fee                    PASSED  [37%]
test_set_emode_category                              PASSED  [38%]
test_set_emode_category_invalid_cases                PASSED  [39%]
test_set_emode_category_below_asset                  PASSED  [40%]
test_set_asset_emode_category                        PASSED  [40%]
test_set_asset_emode_category_invalid_threshold      PASSED  [41%]
test_set_unbacked_mint_cap                           PASSED  [42%]
```

```
test_set_pool_paused                                    PASSED  [42%]
test_update_bridge_protocol_fee                         PASSED  [43%]
test_update_flash_loan_premium_total                    PASSED  [44%]
test_update_flash_loan_premium_to_protocol              PASSED  [44%]
test_only_pool_admin                                    PASSED  [45%]
test_only_emergency_admin                               PASSED  [46%]
test_only_emergency_or_pool_admin                       PASSED  [46%]
test_only_asset_listing_or_pool_admins                  PASSED  [47%]
test_only_risk_or_pool_admins                           PASSED  [48%]
test_supply                                             PASSED  [48%]
test_withdraw                                           PASSED  [49%]
test_withdraw_bad_hf                                    PASSED  [50%]
test_finalize_transfer                                  PASSED  [51%]
test_finalize_transfer_bad_hf                           PASSED  [51%]
test_set_user_use_reserve_as_collateral                 PASSED  [52%]
test_set_user_use_reserve_as_collateral_bad_hf          PASSED  [53%]
test_set_user_use_reserve_as_collateral_twice           PASSED  [53%]
test_set_user_use_reserve_as_collateral_isolation_mode  PASSED  [54%]
test_validate_supply_paused                             PASSED  [55%]
test_validate_supply_frozen                             PASSED  [55%]
test_validate_supply_active                             PASSED  [56%]
test_validate_supply_amount_zero                        PASSED  [57%]
test_validate_supply_cap                                PASSED  [57%]
test_validate_withdraw_paused                           PASSED  [58%]
test_validate_withdraw_frozen                           PASSED  [59%]
test_validate_withdraw_amount_zero                      PASSED  [60%]
test_validate_withdraw_insufficient_balance             PASSED  [60%]
test_validate_borrow_paused                             PASSED  [61%]
test_validate_borrow_frozen                             PASSED  [62%]
test_validate_borrow_amount_zero                        PASSED  [62%]
test_validate_borrow_borrowing_disabled                 PASSED  [63%]
test_validate_borrow_price_oracle_sentinel_disallowed   PASSED  [64%]
test_validate_borrow_interest_rate_mode                 PASSED  [64%]
test_validate_borrow_borrow_cap                         PASSED  [65%]
test_validate_borrow_isolation_not_borrowable           PASSED  [66%]
test_validate_borrow_isolation_debt_ceiling             PASSED  [66%]
test_validate_borrow_emode_category                     PASSED  [67%]
test_validate_borrow_zero_collateral                    PASSED  [68%]
test_validate_borrow_bad_hf                             PASSED  [68%]
test_validate_borrow_insufficient_collateral            PASSED  [69%]
test_validate_borrow_stable_disabled                    PASSED  [70%]
test_validate_borrow_collateral_currency                PASSED  [71%]
test_validate_borrow_stable_max_size                    PASSED  [71%]
test_validate_repay_paused                              PASSED  [72%]
test_validate_repay_amount_zero                         PASSED  [73%]
test_validate_repay_same_block_stable                   PASSED  [73%]
test_validate_repay_same_block_variable                 PASSED  [74%]
test_validate_repay_no_debt                             PASSED  [75%]
test_validate_repay_max_on_behalf_of                    PASSED  [75%]
test_validate_swap_rate_mode_paused                     PASSED  [76%]
test_validate_swap_rate_mode_frozen                     PASSED  [77%]
test_validate_swap_rate_mode_no_stable_debt             PASSED  [77%]
test_validate_swap_rate_mode_no_variable_debt           PASSED  [78%]
test_validate_swap_rate_mode_stable_disabled            PASSED  [79%]
test_validate_swap_rate_mode_stable_collateral          PASSED  [80%]
test_validate_rebalance_stable_borrow_rate_paused       PASSED  [80%]
test_validate_rebalance_stable_borrow_rate_criteria     PASSED  [81%]
test_validate_set_use_reserve_as_collateral_paused      PASSED  [82%]
test_validate_set_use_reserve_as_collateral_no_balance  PASSED  [82%]
test_validate_flash_loan_paused                         PASSED  [83%]
test_validate_flash_loan_simple_paused                  PASSED  [84%]
test_validate_flash_loan_disabled                       PASSED  [84%]
test_validate_liquidation_call_paused                   PASSED  [85%]
test_validate_liquidation_call_sentinel                 PASSED  [86%]
test_validate_liquidation_call_health_factor            PASSED  [86%]
test_validate_liquidation_call_collateral_not_enabled   PASSED  [87%]
test_validate_liquidation_call_no_debt                  PASSED  [88%]
test_validate_health_factor                             PASSED  [88%]
test_validate_hf_and_ltv                                PASSED  [89%]
test_validate_transfer                                  PASSED  [90%]
```

```
test_validate_drop_reserve_zero_address                 PASSED  [91%]
test_validate_drop_reserve_invalid_asset                PASSED  [91%]
test_validate_drop_reserve_atoken                       PASSED  [92%]
test_validate_drop_reserve_stable                       PASSED  [93%]
test_validate_drop_reserve_variable                     PASSED  [93%]
test_validate_set_user_emode_invalid_category           PASSED  [94%]
test_validate_set_user_emode_category_match_borrowings  PASSED  [95%]
test_getters                                            PASSED  [95%]
test_wad_mul                                            PASSED  [96%]
test_wad_div                                            PASSED  [97%]
test_ray_mul                                            PASSED  [97%]
test_ray_div                                            PASSED  [98%]
test_ray_to_wad                                         PASSED  [99%]
test_ray_mul_div_rounding                               PASSED  [100%]
```

# Appendix B Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

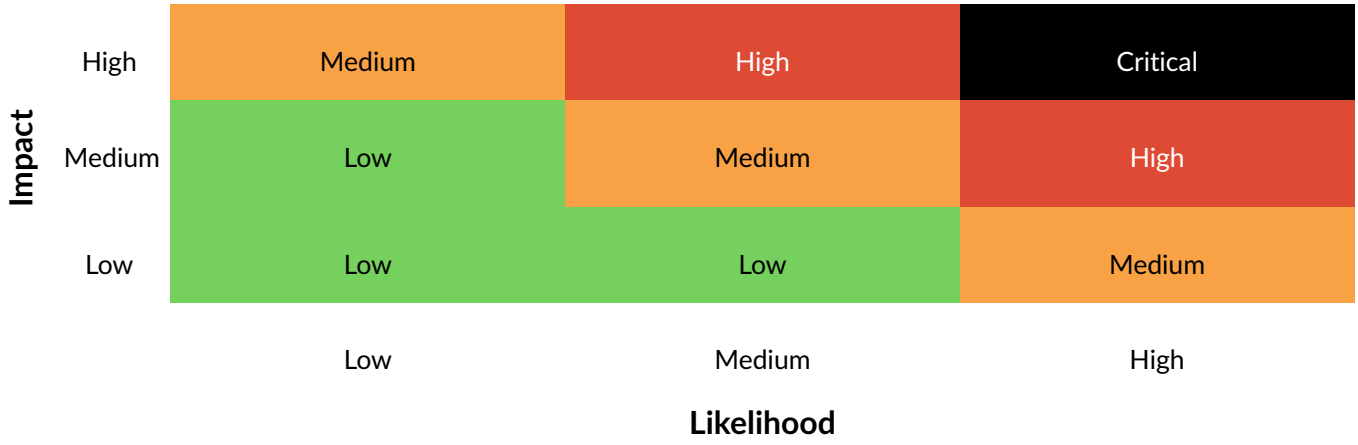


Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

# References

[1] Sigma Prime. Solidity Security. Blog, 2018, Available: https://blog.sigmaprime.io/solidity-security.html. [Accessed 2018].

[2] NCC Group. DASP - Top 10. Website, 2018, Available: http://www.dasp.co/. [Accessed 2018].