



Pashov Audit Group

# Aave Security Review

November 29th 2025 - December 3rd 2025



## Contents

1. About Pashov Audit Group .....	3
2. Disclaimer .....	3
3. Risk Classification .....	3
4. About Aave v3.6.0 .....	4
5. Executive Summary .....	4
6. Findings .....	9
<b>Low findings</b> .....	<b>10</b>
[L-01] Missing event emission for change in pending LTV .....	10



## 1. About Pashov Audit Group

Pashov Audit Group consists of 40+ freelance security researchers, who are well proven in the space - most have earned over \$100k in public contest rewards, are multi-time champions or have truly excelled in audits with us. We only work with proven and motivated talent.

With over 300 security audits completed — uncovering and helping patch thousands of vulnerabilities — the group strives to create the absolute very best audit journey possible. While 100% security is never possible to guarantee, we do guarantee you our team's best efforts for your project.

Check out our previous work [here](#) or reach out on Twitter [@pashovkrum](#).

## 2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

## 3. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

### Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users
- **Medium** - leads to a moderate material loss of assets in the protocol or moderately harms a group of users
- **Low** - leads to a minor material loss of assets in the protocol or harms a small group of users

### Likelihood

- **High** - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost
- **Medium** - only a conditionally incentivized attack vector, but still relatively likely
- **Low** - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive



## 4. About Aave v3.6.0

Aave v3.6.0 introduces decoupled eMode configurations, allowing assets to be exclusively borrowable or collateral within specific eModes, while removing automatic collateral enabling for aTokens, isolated collaterals, and transfers to reduce gas costs and simplify code. It also adds functions to renounce unused allowances, aligns event emissions with OpenZeppelin standards, deprecates certain legacy functions, and updates core contracts, libraries, and periphery logic to support these improvements and more granular risk management.

## 5. Executive Summary

A time-boxed security review of the `aave-dao/aave-v3-origin` repository was done by Pashov Audit Group, during which `0x37`, `unforgiven`, `0xl33`, `Shaka`, `ZeroTrust01` engaged to review Aave v3.6.0. A total of 1 issue was uncovered.

### Protocol Summary

Project Name	Aave v3.6.0
Protocol Type	Lending Protocol
Timeline	November 29th 2025 - December 3rd 2025

#### Review commit hash:

- [c4857cf9a78560cc9326c8dc7ee7dbd5b7aed8bc](#)  
(aave-dao/aave-v3-origin)

#### Fixes review commit hash:

- [4df386b1c57381564a603fc78e0ad5663b567dae](#)  
(aave-dao/aave-v3-origin)

## Attack vectors covered

## Collateral Validation, LTV0 Rules & Collateral Manipulation

### eMode collateral timing and LTV0 behavior

**Description:** Users may exploit timing when assets switch from eMode-only LTV0 collateral to non-zero LTV. Also includes adding LTV0 collateral in eMode or global mode.

**Protection:** `validateUseAsCollateral()`, `getUserReserveLtv()`, and `validateSetUserEMode()` prevent enabling LTV0 collateral in target eMode unless allowed.

### Using disabled or LTV0 reserves to manipulate health factor

**Description:** Attempting to manipulate HF using reserves that are disabled or LTV0.

**Protection:** Reserve can only be enabled when balance > 0. Automatically disabled when balance = 0. Collateral cannot be disabled unless user supply = 0.



## Withdrawing/transferring non-LTV0 collateral before LTV0 collateral

**Description:** Trying to withdraw or transfer safer collateral first to escape LTV0 constraints.

**Protection:** `validateHFAndLtvzero()` ensures any movement is restricted when LTV0 collateral is present.

## Exploiting LTV precedence to gain favorable borrowing terms

**Description:** Attempt to exploit eMode/global LTV precedence mismatch.

**Protection:** All paths consistently use a single function for effective LTV selection.

## Borrowability, Debt Creation, Flashloans & eMode Interactions

### Borrowing when asset is not borrowable in target eMode

**Description:** Attacker attempts to borrow disallowed assets in eMode.

**Protection:** `validateBorrow()` and `validateSetUserEMode()` enforce borrowable bitmaps for both modes.

### Flashloan bypass of eMode restrictions

**Description:** Trying to use flashloans to bypass borrowability or collateral rules.

**Protection:** Debt creation path only counts user-enabled collateral; LTV0 checks prevent bypass.

### Changing eMode while borrowing

**Description:** Attacker attempts eMode switch during borrowing to gain unintended behavior.

**Protection:** All borrowed assets must remain borrowable in target eMode; HF recalculated before finalizing switch.

### Reaching unexpected states by combining eMode actions

**Description:** Combining multiple eMode transitions to reach an invalid state.

**Protection:** eMode transition logic always validates target state consistency.

## Health Factor Manipulation, Liquidation Threshold, Borrowable Bitmap

### eMode LTV changes and instant liquidation risk

**Description:** Could users be liquidated instantly when eMode LTV changes ( $LTV > 0 \rightarrow LTV_0$ ).



**Protection:** Existing positions are grandfathered; borrowers receive governance notice for changes.

### User switches eMode leaving unhealthy HF

**Description:** Changing eMode may reduce LT/LTV and push HF below 1.0.

**Protection:** HF is validated after applying target eMode configuration. Switch fails if HF unsafe.

### Borrowable bitmap changes and increased liquidation exposure

**Description:** Disabling the borrowable bitmap for eMode-only collateral can cause existing positions to fall back to the normal-mode liquidation threshold and LTV values. This may increase liquidation risk if the user relied on more favorable eMode parameters.

**Protection:** This behavior is governed by protocol governance. Active borrowers receive advance notice of such changes and are expected to adjust or close their positions before the update to avoid adverse effects.

## Decoupling of eMode vs Global Mode (LT/LTV, isolation logic)

### Decoupled eMode/global LTV & LT configuration inconsistencies

**Description:** Decoupling introduces risk of mismatched effective parameters.

**Protection:** Single source of truth for effective LTV; decoupling prevents cross-mode leakage.

### Decoupling's impact on existing users

**Description:** Concerns regarding backward compatibility.

**Protection:** Positions and calculations remain valid post-upgrade.

### Decoupling eMode 0 and >0 in all actions

**Description:** Operations must respect separate LTV/LT for eMode 0 and non-zero eMode.

**Protection:** `getLiquidationThreshold()` and `getLtv()` use updated decoupled logic.

### Switching to eMode incorrectly enabling global-mode disabled collateral

**Description:** Switching modes must not mistakenly enable assets allowed only in one mode.

**Protection:** eMode switch validates all existing collateral against target configuration.

### General global vs eMode LT/LTV separation

**Description:** Decoupling ensures updates to one do not affect the other.

**Protection:** Independent configuration paths guarantee isolation.



## Isolation Mode & Automatic Collateral Enabling

### Isolation mode collateral bypass risk

**Description:** Removal of `ISOLATED_COLLATERAL_SUPPLIER_ROLE` raises bypass concerns.

**Protection:** `validateAutomaticUseAsCollateral()` blocks automatic enabling for isolation-mode assets.

### Side effects from disabling automatic collateral enabling in transfers/liquidations

**Description:** Whether transfer/liquidation depends on auto-enabling collateral.

**Protection:** Execution paths explicitly set collateral when required; no reliance on automatic behavior.

### DOS or manipulation via token donations

**Description:** Donation may manipulate collateral logic or block eMode switching.

**Protection:** Donations do not trigger automatic collateral enabling; cannot block eMode change.

## eMode Categories, Configuration, and Category-0 Handling

### Using `eModeCategories[0]` incorrectly

**Description:** Risk of unintended behavior if category 0 index used.

**Protection:** Checks prevent indexing or setting category 0; global mode data stored via `_reserves`.

### Impact of `ltvzeroBitmap` on eMode removals

**Description:** Fear that new bitmap may block removal of assets from eMode.

**Protection:** No behavior change; bitmap off by default; removals unaffected.

## Storage Layout, Upgrade Safety & Struct Changes

### Storage layout safety of new eMode changes

**Description:** Adding/removing struct fields may cause storage collision.

**Protection:** Memory-only structs unaffected; `EModeCategory` field appended safely.

### Storage safety when upgrading Aave v3 contracts

**Description:** Typical upgrade concerns (proxy pattern).



Protection: `ltvZeroBitmap` addition does not shift any storage slots.

## Event Emission Correctness

### Correct event behavior on tokens

Description: `transferFrom()` no longer emits approval events.

Protection: Correct event logic retained in `approval()` ; removed from transfer path as intended.

### Scope

`GenericLogic.sol` `LiquidationLogic.sol` `SupplyLogic.sol` `ValidationLogic.sol`  
`DataTypes.sol` `Pool.sol` `PoolConfigurator.sol` `AToken.sol`  
`VariableDebtToken.sol` `DebtTokenBase.sol` `IncentivizedERC20.sol`





## 6. Findings

### Findings count

Severity	Amount
Low	1
Total findings	1

### Summary of findings

ID	Title	Severity	Status
[L-01]	Missing event emission for change in pending LTV	Low	Resolved



## Low findings

### [L-01] Missing event emission for change in pending LTV

`PoolConfigurator` sets the `_pendingLtv` mapping to zero in two places: `_setReserveLtvzero()` and `configureReserveAsCollateral()`. In the first case, the `PendingLtvChanged` event is emitted with the new pending LTV value (zero).

```
function _setReserveLtvzero(
  (...)
  if (ltvZero) {
    if (currentConfig.getLtv() == 0) return;
    newPendingLtv = currentConfig.getLtv();
    _pendingLtv[asset] = newPendingLtv;
    currentConfig.setLtv(0);
  } else {
    // ltvzero can only be removed on non frozen reserves
    require(!currentConfig.getFrozen(), Errors.ReserveFrozen());
    newLtv = _pendingLtv[asset];
    if (newLtv == 0 || currentConfig.getLtv() != 0) return;
    currentConfig.setLtv(newLtv);
    delete _pendingLtv[asset];
  }
  @> emit PendingLtvChanged(asset, newPendingLtv);
```

However, the `configureReserveAsCollateral()` function misses the event emission when clearing the pending LTV:

```
function configureReserveAsCollateral(
  (...)
  if (currentConfig.getFrozen()) {
    _pendingLtv[asset] = ltv;
    newLtv = 0;

    emit PendingLtvChanged(asset, ltv);
  } else {
    @> if (_pendingLtv[asset] != 0) delete _pendingLtv[asset];
    currentConfig.setLtv(ltv);
  }
```

It is recommended to emit the `PendingLtvChanged` event when the pending LTV is cleared in `configureReserveAsCollateral()` to maintain consistency and provide accurate event logs for off-chain monitoring and analysis.