Security Assessment Report

Horizon Aave

May 2025

Prepared for Aave Labs

certora

aave/horizon

Table of contents

certora

aave/horizon

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Horizon Aave	https://github.com/aave/aave-v3-horizon	04419e2	EVM

Project Overview

This document describes the manual code review and findings of **Aave V3.3 Horizon**. The work was undertaken from **May 19, 2025** to **May 26, 2025**

The following contract list is included in our scope:

```
aave-v3-horizon/src/contracts/protocol/tokenization/RwaAToken.sol
aave-v3-horizon/src/contracts/protocol/tokenization/AToken.sol
aave-v3-horizon/src/contracts/instances/RwaATokenInstance.sol
aave-v3-horizon/src/contracts/interfaces/IRwaAToken.sol
aave-v3-horizon/src/contracts/interfaces/IRwaATokenManager.sol
aave-v3-horizon/src/contracts/misc/RwaATokenManager.sol
aave-v3-horizon/src/contracts/protocol/libraries/helpers/Errors.sol
```

The team performed a manual audit of all the Solidity smart contracts. During the manual audit, the Certora team discovered bugs in the Solidity smart contracts code, as listed on the following page.

Protocol Overview

Horizon is a specialized financial infrastructure by Aave Labs designed for institutional Real-World Asset (RWA) integration, operating as a licensed fork of Aave Protocol v3.3 to meet regulatory requirements. Horizon's architecture centers on permissioned RWA assets that can only be used as collateral by verified addresses meeting each token issuer's eligibility criteria. The system allows any user to supply non-RWA tokens, such as stablecoins, to earn yield, while RWA collateral suppliers must be authorized by respective issuers. Since aTokens representing RWA (RwaATokens) deposits are extensions of regulated securities, their transfers are disabled for users, though a protocol-wide authorizedTransfer() function can forcibly move RwaATokens when legally required. Liquidations are restricted to entities already allowed to hold the underlying RWA collateral, ensuring regulatory compliance even during enforcement events. This structure enables Horizon to bridge traditional financial assets with DeFi while maintaining necessary regulatory safeguards for institutional participation.

More generally, Aave enables users to participate as liquidity providers by depositing assets into lending pools or as borrowers who can take out loans using their deposited assets as collateral.
Its architecture revolves around a unified pool containing different assets. Each asset has an associated aToken that represents a user's deposit and aTokens that correspond to borrowable non-RWA tokens automatically accrue interest, while borrowers receive debt tokens that track their loan obligations and accrued interest. Key actors include lenders who earn passive income, borrowers who can make overcollateralized loans or flash loans, liquidators who maintain the system's solvency by liquidating undercollateralized positions, and governance participants who decide on protocol parameters.

Security Considerations

Horizon represents a significant adaptation of the Aave Protocol, introducing specialized functionality for Real-World Assets while maintaining core lending mechanics. The implementation includes critical modifications, primarily centered around the new RWA-specific aToken contract and its transfer restrictions.

Our security assessment focused on both the implementation of these new restrictions and their interaction with unchanged protocol components. We verified that the core economic mechanics remain sound while RWA-specific constraints are properly enforced.

While the protocol retains much of Aave v3.3's battle-tested architecture, the new permissioning layer and administrative capabilities introduce additional attack surfaces that must be carefully managed through robust governance processes and comprehensive security monitoring. These changes, though necessary for institutional RWA integration, require ongoing vigilance to ensure they don't compromise the protocol's security posture.

Audit Goals

1. Verify RWA tokens cannot be supplied on behalf of other users.

2. RwaATokenManager admin can transfer on behalf of users, as long as a healthy health factor is maintained.

3. Ensure all transfer and allowance methods are blocked for RwaATokens, namely:
    a. approve()
    b. decreaseAllowance()
    c. increaseAllowance()
    d. transfer()
    e. transferFrom()
    f. transferOnLiquidation()
    g. transferUnderlyingTo()
    h. permit()

4. Verify that RwaATokens cannot be liquidated; only RWA underlying collateral can be liquidated.

5. RWA collateral cannot be borrowed.

6. RWA debt tokens cannot be created.

7. RWA collateral cannot be flashloaned.

8. If issuers' allowlist is corrupted, there are no major risks to Horizon Aave or its users.

9. Verify that non-RWA tokens maintain their usual properties and adhere to specifications such as:
    a. Debt can only be repaid with the underlying token or aToken corresponding to the variable-debt tokens (vTokens) issued.

    b. Non-RWA tokens can be supplied on behalf of other users.

    c. Non-RWA tokens can be borrowed and flashborrowed.

    d. Non-RWA tokens can be withdrawn to the original supplier or a different user address.

10. In case a RwaAToken holder that has a borrow position open loses their private key, the issuer will be able to successfully repay the user's debt and transfer their RwaATokens to their new wallet via authorizedTransfer().

Coverage and Conclusions

1. RWA tokens cannot be supplied on behalf of other users as expected.

2. Confirmed that RwaATokenManager admin can transfer on behalf of users, as long as a healthy health factor is maintained.

3. All transfer and allowance methods are blocked for RwaATokens, namely:
    a. approve()
    b. decreaseAllowance()
    c. increaseAllowance()
    d. transfer()
    e. transferFrom()
    f. transferOnLiquidation()
    g. transferUnderlyingTo()
    h. permit()

4. RwaATokens cannot be liquidated as the implementation of the transferOnLiquidation() function prevents it. Only RWA underlying collateral can be liquidated.
    a. The system requires liquidationProtocolFeeAmount to be zero. Otherwise, the liquidation will revert due to the transferOnLiquidation() function call.

5. RWA collateral cannot be borrowed as the implementation of the transferUnderlyingTo() function prevents it.

6. Technically RWA debt tokens cannot be created as the implementation of the transferUnderlyingTo() function prevents it, and all paths creating debt tokens must call this function.

7. RWA collateral cannot be flashloaned as the implementation of the transferUnderlyingTo() function prevents it.

8. If issuers allowlist is corrupted, there are no major risks to Horizon Aave or its users, because loans need to be overcollateralized (assuming the reserves have been configured appropriately).

9. It has been concluded that non-RWA tokens maintain their usual properties and adhere to specifications such as:
    a. Debt can only be repaid with the underlying token corresponding to the variable-debt tokens (vTokens) issued, or with the aToken corresponding to the variable-debt tokens (via repayWithATokens()). As debt can only be created for non-RWA tokens, debt can only be repaid with the corresponding non-RWA tokens.

    b. It has been confirmed that non-RWA tokens can be supplied on behalf of other users.

    c. It has been confirmed that non-RWA tokens can be borrowed and flashborrowed.

    d. It has been confirmed that non-RWA tokens can be withdrawn to either the original supplier or to a different address.

10. In case a RwaAToken holder that has a borrow position open loses their private key, the issuer will be able to successfully repay the user's debt and transfer their RwaATokens to their new wallet via the authorizedTransfer() function.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	7	7	5
Total	7	7	5

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
I-01	Non-RWA tokens, such as stablecoins, can be used as collateral if the reserve configuration is inappropriate	Informational	Acknowledged
I-02	Aave does not enforce that RWA collateral tokens cannot be transferred	Informational	Fixed
I-03	Update documentation to reflect code nomenclature	Informational	Fixed
I-04	Centralization risk by RwaATokenManager Admin	Informational	Acknowledged
I-05	Change transferOnLiquidation() visibility to external	Informational	Fixed
I-06	RWA collateral liquidation fails if protocol fee is not zero	Informational	Fixed
I-07	Document rules or boundaries for RwaATokens unbacked mint cap	Informational	Fixed

Informational Issues

I-01. Non-RWA tokens, such as stablecoins, can be used as collateral if the reserve configuration is inappropriate

Description: Non-RWA tokens, such as stablecoins, cannot be used as collateral to borrow RWA tokens, but they could be used to borrow more non-RWA tokens if the LTV is set inappropriately to something other than zero. Despite not making sense economically to perform such an action, it would break the assumption that non-RWA tokens cannot be used as collateral.

Consider implementing a check in the deployment script to ensure the reserve is configured as intended upon deployment.

Customer's response: Acknowledged. This flexibility is intended as a future potential feature.
Fix Review: Accepted customer response, no fix is required.

I-02. Aave does not enforce that RWA collateral tokens cannot be transferred.

Description: Aave itself does nothing to prevent RWA underlying collateral tokens from being transferred to different addresses, either through withdraw() or liquidationCall(), for example. If issuers' allowlist is corrupted, there are no major risks to Horizon Aave or its users, because loans need to be overcollateralized (assuming the reserves have been configured accordingly).

Customer's response: Fixed. Documentation has been updated accordingly to clarify these transfers during a withdraw() or liquidationCall(), which are subject to underlying RWA token permissioning.
Fix Review: Fixed. Clarifying points have been added to the documentation.

I-03. Update documentation to reflect code nomenclature

Description: Document

a. Change aTokens to RwaATokens to increase document clarity.

b. Change aToken Transfer Admin contract to RwaATokenManager to maintain a 1:1 relationship between documentation and code.

Customer's response: Fixed.
Fix Review: Fixed in the documentation markdown file.

I-04. Centralization risk by RwaATokenManager Admin

Description: RwaATokenManager admin, like other Aave permissioned roles, has the ability to take full control of user assets. In this case, specifically RwaATokens. Therefore, it is of utmost importance to establish high standards for operational security processes and comprehensive security monitoring.

Customer's response: Acknowledged. The permissioned role is necessary in this Aave instance in order to comply with regulatory requirements.
Fix Review: The issue has been acknowledged, but will not be fixed as it would go against regulatory requirements.

I-05. Change transferOnLiquidation() visibility to external

Description: transferOnLiquidation() is never called internally, therefore, its visibility can be changed to external.

Customer's response: Fixed.
Fix Review: Fixed.

I-06. RWA collateral liquidation fails if the protocol fee is not zero

Description: It is not explicitly documented this is expected behavior for the Horizon instance of Aave. We recommend adding an explicit note in the documentation to highlight the fact protocol fees on liquidation will always be zero, and if for whatever reason it changes, the revert is expected.

Customer's response: Fixed. Documentation has been updated accordingly to clarify that liquidations will revert if the Liquidation Protocol Fee is not properly configured to 0.
Fix Review: Fixed. The issue has now been properly addressed in the documentation.

I-07. Document rules or boundaries for RwaATokens unbacked mint cap

Description: Considering the sensitive nature of regulatory frameworks, it is important to delineate accepted values for the unbacked mint cap of RwaATokens, by either referencing acceptable values in the documentation of a given RwaAToken or implementing checks in deploy scripts.

Customer's response: Fixed. The documentation has been updated accordingly to clarify that there is no intention to mint unbacked RwaATokens, through proper configuration.
Fix Review: Fixed - documentation has been updated.

Disclaimer

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Its flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare and hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.