Hydration: Intergalactic Aave v3 deploy

Security Review

Cantina Managed review by:

0xWeiss, Security Researcher

Brian McMichael, Security Researcher

January 23, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Hydration is a next-gen DeFi protocol which is designed to bring an ocean of liquidity to Polkadot.

From Nov 18th to Nov 20th the Cantina team conducted a review of intergalactic-aave-v3-deploy on commit hash b4b83090. The team identified a total of **2** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 0
- Informational: 2

3 Findings

3.1 Informational

3.1.1 Protocol is EOA controlled

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: Administration of protocol security is handled by a single EOA account via the ACLManager contract.

> *Update January 21, 2025*
>
> *As of January 20, 2025, the deployer keys have been removed from privileged roles in the system.*
>
> - `poolAdmin` revocation transaction: 0x814d...bf4c.
> - `emergencyAdmin` revocation transaction: 0xa3e5...d957.
> - `riskAdmin` revocation transaction: 0x0312...05bd.

The Access Control List Manager `POOL_ADMIN` is set to address `0x71FeB8b2849101a6E62e3369eaAfDc6154CD0Bc0` on the Hydration network, which resolves to an EOA account. The block explorer sometimes displays unverified contracts as EOA's on this network, but this can be confirmed with foundry:

```
$ cast code 0x71FeB8b2849101a6E62e3369eaAfDc6154CD0Bc0
0x
```

The `POOL_ADMIN` role is effectively a superuser in the protocol, and inherits all methods accessible to `ASSET_LISTING_ADMIN` and `RISK_ADMIN`, as well as the ability to extract tokens from pools and change oracles. See the details at the roles-pool-admin section from the docs.

Note: The `emergencyAdmin` is set to the same address as the current `poolAdmin` EOA, and although it has fewer permissions, the same considerations apply, as this role is able to pause and unpause Pools. See roles-emergency-admin section from the docs.

The use of an EOA for protocol management is not recommended due to the potential of private key mismanagement on a single machine, the probability of a private key to be obtained from an unsecured machine remotely via phishing or network breach, or a physical intrusion against the machine. A single operator controlling this EOA can cause significant damage and chaos in a protocol when in direct control of the contracts.

A multisig or other more secure method of maintaining the protocol is recommended here to maintain operational security of the protocol. Aave uses the `Executor` contract in this role in V3. See:

- Code: Executor.sol.
- Contract: 0xEE56e2B3D491590B5b31738cC34d5232F378a8D5.
- `addPoolAdmin` transaction: 0xc1a825...b9ace3

Impact: Originally marked as high because safe key management is difficult for production systems and end-users will hesitate to deposit funds into an EOA-controlled protocol. Given that the deployer keys have been removed from privileged roles in the system, it is lowered to informational.

Likelihood: Phishing attacks are becoming more complicated, and EOA's are difficult to maintain and safely transmit between intended operators. A disgruntled operator can extract funds directly via intent or coercion. There is a high likelihood that the entire protocol could be compromised by this account and role.

Recommendation: Replace all permissioned EOA addresses in the protocol with multisig wallets or Executor contracts, as necessary. The existing deployment can be maintained and permissions managed by adding a new pool admin with a contract controller and revoking the existing EOA account from the pool admin role. Also update `emergencyAdmin` to a secure address and analyze and mitigate the use of any other permissioned EOA's in the deployment.

3.1.2 Contracts are unverified on the block explorer

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: At the time of publication, most of the contracts are listed as unverified on the block explorer at `https://explorer.evm.hydration.cloud/`.

Description: Several of the main contracts are code verified on the block explorer, however most of the logic and periphery contracts are not. This makes verifying the code difficult for end-users and erodes trust in the protocol.

I was able to match the on-chain bytecode with that in the deployment artifacts for many of the deployed contracts, however, more casual users will want to verify that the code in the explorer matches the code on-chain. This applies to depositors to the protocol, but also to holders of the underlying tokens distributed by the system, which are also unverified.

I checked many, but not all of the deployed contracts for their verification status, currently unverified contracts include `BorrowLogic` and other libraries, `DOT-AToken-Hydration`, `DOT-StableDebtToken-Hydration`, and other tokens, `Pool-Implementation`, `Pool-Proxy-Hydration`, and many other deployed contracts.

Impact: Depositors will feel safe utilizing the protocol if the contract code is visible in block explorers, as the Aave code that this is based on is generally considered safe and battle-tested, however, they will need to be able to verify not only that the code matches what is expected, but also that they are interacting with the appropriate contracts.

Token holders in the system will want to view methods and code available on the tokens that they are holding, and verified ERC20 token contracts will allow block explorers to appropriately display user balances and information regarding their token holdings.

Likelihood: Unverified contracts are not necessarily a heavy security risk, as everything on-chain will still work through the code paths, but the protocol will have friction with integrators and end-users if the contract code is not available to them in an accessible way.

Recommendation: Ensure that all contracts deployed in the `deployments` folder artifacts have been verified in the flagship block explorer.

4 Appendix

4.1 Review summary

Overall the deployment itself looks good.

- Scripts have been cross-referenced with other Aave deployments and adhere to standards.
- Typescript variables were properly interpreted by the compiler and passed as valid params.
- Contracts verified with `solc 0.8.10` using `100000 runs`, which matches the Aave mainnet deployment.
- Bytecode deployed on-chain matches that provided in the Hydration deployment artifacts and compared to Aave mainnet deployments.
- Ensure access control is revoked to existing EOA after multisig handoff.
- Ensure contracts are verified on flagship block explorer.

It would be ideal for the team to prepare a detailed document with the descriptions for every role and the different multi-sigs with different thresholds.

Additionally, it is always good to have timelock contracts as owners in some functions that do not need to be triggered with maximum urgency (such as a `pause` function).

This is also a good guide for what should be considered the basic standard OPSEC practices, which include managing all the roles in the codebase:

Using a monitoring system to track state transitions within the codebase is also advised. This practice could help to detect un-expected changes in state and keep track of most of the parameters of the system.