

Monad: Distribution Escrow

Security Review

Cantina Managed review by:
0xRajeev, Lead Security Researcher
Rscodes, Associate Security Researcher

December 1, 2025

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Informational . . . 4
  - 3.1.1 `DEFAULT_ADMIN_ROLE` should have the highest possible security . . . 4
  - 3.1.2 Verification of added wallets and amounts should have sufficient rigor and redundancy 4
  - 3.1.3 `receive()` may be restricted to `SENDING` stage to prevent premature crediting of funds . . . 4
  - 3.1.4 `WalletRemoved` event is missing the amount parameter for consistency . . . 4
  - 3.1.5 `WalletNotAdded` can be named better to account for other revert scenarios . . . 5
  - 3.1.6 Allowing `removeWallets(...)` in any stage may prevent accidentally removed wallets from being added again . . . 5

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 level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

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 are 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

The Monad protocol delivers full EVM compatibility with breakthrough performance, true decentralization, production-grade security, and exceptional throughput.

From Oct 21st to Oct 22nd the Cantina team conducted a review of distribution-escrow on commit hash c36783bc. The team identified a total of **6** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	0	0	0
Gas Optimizations	0	0	0
Informational	6	3	3
Total	**6**	**3**	**3**

2.1 Scope

The security review had the following components in scope for distribution-escrow on commit hash c36783bc:

```
src/Escrow.sol
```

3 Findings

3.1 Informational

3.1.1 `DEFAULT_ADMIN_ROLE` should have the highest possible security

Severity: Informational

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

Description: `DEFAULT_ADMIN_ROLE` is used to access control `finalizeCurrentStage()` and `rescueTokens(...)`. It can also grant/revoke `ADD_ROLE`, `REMOVE_ROLE` and `SENDER_ROLE`, which are used to add/remove eligible wallets and send them their due amounts. Given the criticality of this admin role, it should have the highest possible security. This is currently planned to be a 4/7 multisig wallet.

Recommendation: Ensure that `DEFAULT_ADMIN_ROLE` is a 4/7 or higher threshold multisig wallet with diverse signers and platforms using dedicated and isolated devices along with offline verification of calldata and clear signing to minimize compromise.

Monad Foundation: Acknowledged.

Cantina Managed: Acknowledged.

3.1.2 Verification of added wallets and amounts should have sufficient rigor and redundancy

Severity: Informational

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

Description: The `ADD_ROLE` and `SENDER_ROLE` are expected to be temporarily granted to wallets that may be assumed to be compromised in the threat model. The planned flow therefore includes a verification step of added wallets and amounts, which includes running scripts and manual checking to top+random accounts as indicated. Any mistake/miss here may lead to ineligible/incorrect wallets/amounts being sent.

Recommendation: Ensure that this verification step of added wallets and amounts has sufficient rigor and redundancy by comparing the added wallets/amounts against multiple redundant offline lists several times in different ways.

Monad Foundation: Acknowledged.

Cantina Managed: Acknowledged.

3.1.3 `receive()` may be restricted to `SENDING` stage to prevent premature crediting of funds

Severity: Informational

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

Description: The planned flow expects MON to be sent to the Distribution Escrow contract only in the `SENDING` stage after all the eligible wallets have been added and manually verified as being correct.

However, this is not enforced in the `receive()` function. If an incorrect amount (e.g. because of ineligible wallets accidentally included but removed thereafter) is accidentally sent to the contract in the earlier `CLAIM_DETAILS` stage then the differential amount will have to be rescued thereafter.

Recommendation: Consider restricting `receive()` to `SENDING` stage to prevent premature crediting of funds.

Monad Foundation: Fixes merged after commit e425429c.

Cantina Managed: Fix verified.

3.1.4 `WalletRemoved` event is missing the amount parameter for consistency

Severity: Informational

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

Description: Unlike `WalletAdded(wallet, amount)` and `TokensSent(wallet, amountDue)` events, `WalletRemoved(wallet)` does not include the `amount = balanceOf[wallet]` as a event emit parameter, which is inconsistent.

Recommendation: Consider including the amount parameter for consistency.

Monad Foundation: Fixes merged after commit e425429c.

Cantina Managed: Fix verified.

3.1.5 `WalletNotAdded` can be named better to account for other revert scenarios

Severity: Informational

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

Description: `WalletNotAdded` error in `removeWallets(...)` can be triggered not only when operating on a wallet not added, but also on wallets that have been deleted because they have been already removed or already sent their due amounts. The name `WalletNotAdded` therefore accounts for one of the three possible revert scenarios.

Recommendation: Consider if this can be named `WalletNotAddedOrAlreadySentOrAlreadyRemoved` or a shorter but more generic name.

Monad Foundation: Fixes merged after commit e425429c.

Cantina Managed: Fix verified.

3.1.6 Allowing `removeWallets(...)` in any stage may prevent accidentally removed wallets from being added again

Severity: Informational

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

Description: `removeWallets(...)` does not have the `onlyInStage` modifier, which means this can be called by `REMOVE_ROLE` in any of the stages to remove unauthorized wallets that have been added accidentally. However, if `removeWallets(...)` is called after `CLAIM_DETAILS` is finalized, then any wallets that may be incorrectly removed (i.e. eligible recipients of the due amounts) cannot be added again.

Recommendation: Consider if `onlyInStage(EscrowStage.CLAIM_DETAILS)` should be enforced on `removeWallets(...)` so that any eligible wallets incorrectly removed can be added back again. The trade-off is that if ineligible wallets added in this stage are not removed even after offchain manual checks then they cannot be removed in the `SENDING` stage. Such wallets (if identified) can however be skipped for sends and their amounts accordingly accounted for in `owed` while sending MON to the contract or while rescuing.

Monad Foundation: Acknowledged.

Cantina Managed: Acknowledged.