BAIL security

Mira Network

FINAL REPORT

February ‘2025

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the audit, and the audit results may not be complete or identify all possible vulnerabilities or issues. The audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the audit report and should seek additional professional advice if needed. The audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the audit report or the smart contract.

By engaging in a smart contract audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

1. Project Details

Important:
Please ensure that the deployed contract matches the source-code of the last commit hash.

Project	Mira Network
Website	mira.network
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/Aroha-Labs/mira-contracts-v0/tree/c3f5d8cbd5cb84042c08deb637443f6b367e2edf/contracts
Resolution 1	https://github.com/Aroha-Labs/mira-contracts-v0/tree/06b4449d3e4b735ee8d04c399a0b8f73a5b9cd98

2. Detection Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)	Failed Resolution
High	1	1			
Medium	1	1			
Low	5			5	
Informational	6	1		5	
Governance					
Total	13	3		10	

2.1 Detection Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium-level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low-level risk to the project or users. Nevertheless, the issue should be fixed immediately.
Informational	Effects are small and do not pose an immediate danger to the project or users.
Governance	Governance privileges which can directly result in a loss of funds or other potential undesired behavior.

3. Detection

AppRegistryV0

The AppRegistryV0 contract is a simple registry contract which allows for adding apps and updating their status.

Addition can be done via the registerApp function while status updates can be done via the updateAppStatus function.

Furthermore, the admin address can be changed via the transferAdmin function.

Core Invariants:

INV 1: Only existing apps can be changed in their status

INV 2: The registerApp and updateApp functions must only be called by the admin

INV 3: An app cannot be registered twice

INV 4: Only valid apps can be added

INV 5: Only the admin can change the admin address

INV 6: Only existing apps can be changed in their status

Privileged Functions

- registerApp
- updateAppStatus
- transferAdmin

Issue_01	Chance of locking out admin
Severity	**Informational**
Description	The transferAdmin function allows the admin to change the admin address. In the scenario where the wrong address is set, all onlyAdmin functions are essentially rendered unusable which means new apps cannot be added and existing apps cannot be updated in their status.
Recommendations	Consider being careful when the admin address is changed.
Comments / Resolution	Acknowledged.

Issue_02	isAppActive check is unnecessary strict
Severity	**Informational**
Description	The isAppActive function checks if an app is active as follows: *appRegistry[appId].isActive && appRegistry[appId].registrationBlock > 0;* This behavior is unnecessary as an app can only be set to active if registrationBlock > 0
Recommendations	Consider simplifying this check.
Comments / Resolution	Resolved.

Issue_03	No possibility to remove apps from array
Severity	**Informational**
Description	Currently, there is no possibility to remove an app from the array. The only functionality is to update the status. This is not necessarily considered as an issue but rather a side-note to consider in the design.
Recommendations	Consider acknowledging this issue.
Comments / Resolution	Acknowledged.

InferenceListenerV0

The InferenceListenerV0 contract is an event emission contract which allows the admin to emit inference logs which can be retrieved by an off-chain component. Each inferencelog is tied to an appId, a userWallet and a logHash.

Single logs can be emitted via the submitInferenceLog function and batch logs can be emitted via the submitBatchInferenceLogs function.

The contract implements a rate limit functionality which is tied to each appId and by default limits the emission per block per appId to 100 but can be changed by the admin as well.

Core Invariants:

INV 1: Only the admin can call the submitInferenceLog/submitBatchInferenceLog/updateMaxSubmittions and transferOwnership functions

INV 2: A single app cannot have more interferences than maxSubmissionsPerBlock in a single block

Privileged Functions

- submitInferenceLogs
- submitBatchInferenceLogs
- updateMaxSubmissions
- transferAdmin

Issue_04	appId validity is not checked upon submitInferenceLog/submitBatchInferenceLogs
Severity	**Low**
Description	Currently, any appId can be passed as parameter as long as it has a non-zero length. This however does not ensure that the appId is indeed existing + active in the registry.
Recommendations	Consider executing a check which ensures that the provided appId is active in the AppRegistryV0 contract.
Comments / Resolution	Acknowledged.

Issue_05	Rate limit mechanism can be bypassed
Severity	**Low**
Description	As already explained the contract implements a rate limit mechanism. This rate can however be changed by the same address which can emit these events which makes it trivially bypassable.
Recommendations	Consider implementing a more intrusive role system.
Comments / Resolution	Acknowledged.

Issue_06	submissionsCount should be public
Severity	Informational
Description	The submissionsCount mapping tracks submissions per block per app. However the mapping is set to private. Private variables are harder to access compared to public. Additionally solidity generates getters for public variables. Automated tools may also benefit from the submissionsCount variable being set to public.
Recommendations	Consider marking the submissionsCount mapping as public.
Comments / Resolution	Acknowledged.

InferenceStatsV0

The InferenceStatsV0 contract is a storage contract which allows the admin to write stats to the contract storage for each appId. A stat corresponds to:

a) block.number when the stat is written
b) inferenceCount for the app
c) tokenCount for the app

It furthermore expoes various different view-only functions in an effort to fetch stats for a corresponding appId.

Core Invariants:

INV 1: appId must be non-zero length

INV 2: Only the admin can call the writeStats function

Privileged Functions

- writeStats
- transferAdmin

Issue_07	appId validity is not checked upon writeStats
Severity	**Low**
Description	Currently, any appId can be passed as parameter as long as it has a non-zero length. This however does not ensure that the appId is indeed existing + active in the registry.
Recommendations	Consider executing a check which ensures that the provided appId is active in the AppRegistryV0 contract.
Comments / Resolution	Acknowledged.

Issue_08	Missing rate limiting mechanism
Severity	**Low**
Description	Currently, the writeStats function allows multiple calls per block without any rate limiting mechanism. While this is usually not an issue due to the onlyAdmin function, it exposes an inconsistency compared to the InferenceListenerV0 contract.
Recommendations	Consider if it is desired to implement a similar rate limiting mechanism.
Comments / Resolution	Acknowledged.

Issue_09	Potential excessive gas consumption if getStatsHistory is called by another contract
Severity	**Informational**
Description	The getStatsHistory function returns the full appStats array for an appId in memory. In case this array becomes excessively large, that function will revert if another contract calls it because the gas limit of a block is exceeded. This can also be limiting for view-only calls which is however based on the RPC provider.
Recommendations	Consider acknowledging this issue.
Comments / Resolution	Acknowledged.

StakingVault

The StakingVault contract is a simple extension of the ERC4626 vault which ensures that the share inflation attack cannot be executed. This is done by minting an initialAmount of 1e18 shares and requiring an amount of at least 1e18 tokens to be provided by the DEPLOYER during the initialize function.

Core Invariants:

INV 1: The initialize function can only be called once

INV 2: The deposit and mint functions can only be called once the contract is initialized

INV 3: Only the DEPLOYER can initialize the contract

Privileged Functions

- initialize

Issue_10	Initialization of the vault will be incompatible for lower decimal tokens.
Severity	**High**
Description	According to the comments, one full token is needed for the initial deposit. *MIN_DEPOSIT_AMOUNT = 1e18; // Minimum initial deposit (1 full token)* However the MIN_DEPOSIT_AMOUNT does not scale for lower or higher decimal tokens. Therefore if we were to use USDC for this staking vault, we would need $1e12 in order to initialize. This will be simply impossible as it is equal to 1000000000000 USDC. For higher decimal tokens, it will mean less nominal funds will be required.
Recommendations	Consider scaling the MIN_DEPOSIT_AMOUNT based on the asset token decimals.
Comments / Resolution	Resolved.

Issue_11	StakingVaullt is not compatible with tokens that do not return bool on transfer.
Severity	**Medium**
Description	During initialization of the staking vault, the logic requires that a successful transferFrom of the initialAmount is done. *require(IERC20(asset()).transferFrom(msg.sender, address(this), initialAmount), "Transfer failed");* However some tokens such as USDT do not return a bool on transfers and thus will not be able to be used with the vault. Furthermore related to custom tokens is the fact that the transferFrom is being executed before the initialization which can be abused to initialize the contract multiple times in the case of reentrancy and mint a multiple of 1e18 shares. This is however just a note and does not need to be considered because the DEPLOYER is trusted anyways.
Recommendations	Consider using the safeERC20 library to be compatible with tokens that do not return bool on transfers.
Comments / Resolution	Resolved.

Issue_12	Potentially incorrect initial exchange rate in case of transfer-tax tokens or wrong initialAmount
Severity	**Low**
Description	The initialize function mints 1e18 shares and transfers the initialAmount in. This will the determine the exchange rate. If both are 1e18, the ER is simply 1. If however 100e18 assets are distributed in, a user must provide 100e18 assets to receive 1e18 shares. First of all, the DEPLOYER should carefully determine which exchange rate is reasonable and secondly, tokens with a transfer-tax will alter the expected exchange rate, since less tokens are actually received by the vault (besides the fact that ERC4626 vaults are anyways not compatible with TX-tax tokens). Furthermore, if the token is a very cheap token or is denominated with large decimals, the DEPLOYER address can provide a large initialAmount which would then round down any received shares for subsequent depositors. This is however just a note and does not need to be considered because the DEPLOYER is trusted anyways.
Recommendations	Consider being conscious with the initialAmount parameter (we recommend 1e18 to keep it simple) and consider not supporting transfer-tax tokens.
Comments / Resolution	Acknowledged.

Issue_13	initialize does not emit an event
Severity	Informational
Description	The initialize function currently does not emit an event. It is recommended to add events to important functions, given that after calling the initialize function users will be able to stake, it is important to emit an event.
Recommendations	Consider adding an event to the initialize function.
Comments / Resolution	Acknowledged.