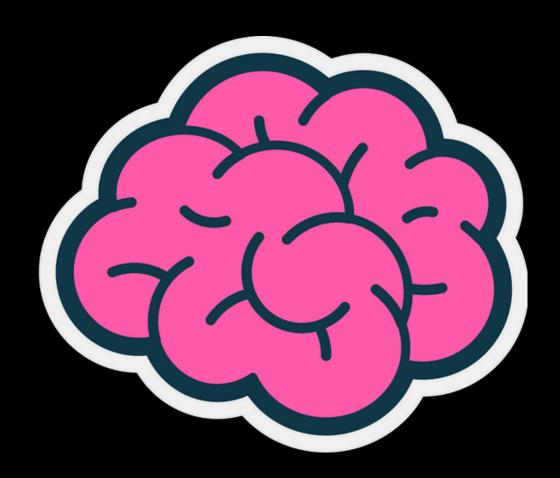


IQ AI Security Review



Lead Auditors



PeterSR



0x539.eth

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- · Risk Classification
- Audit Details
 - Scope
 - Roles
- Executive Summary
 - Issues found
- Findings
 - Medium

Protocol Summary

Tokenized agents for DeFi and beyond

Disclaimer

The Chain Defenders team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of

the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

Likelihood/Impact	High	Medium	Low
High	Н	H/M	M
Medium	H/M	M	M/L
Low	M	M/L	L

Audit Details

Scope

Id	Files in scope	
1	AIToken.sol	
2	Agent.sol	
3	AgentFactory.sol	
4	AgentRouter.sol	
5	BootstrapPool.sol	
6	LiquidityManager.sol	
7	TokenGovernor.sol	

Roles

Id	Roles
1	Owner
2	User

Executive Summary

Issues found

Severity	Count	Description
High	0	Critical vulnerabilities
Medium	1	Significant risks
Low	0	Minor issues with low impact
Informational	0	Best practices or suggestions
	0	Optimization opportunities

Findings

Medium

Mid 01 Unchecked Fraxswap Pair Fee

Finding description and impact

The LiquidityManager:: addLiquidityToFraxswap function does not verify the fee of an existing Fraxswap pair. A malicious actor could front-run the liquidity movement by creating a Fraxswap pair with a higher fee than expected. This would cause the contract to interact with a pair that has an unintended fee structure, potentially leading to incorrect swap calculations, reduced liquidity efficiency, and unexpected fee costs for users.

Proof of Concept

- $\textbf{1. The} \ \texttt{moveLiquidity} \ \textbf{function} \ \textbf{is} \ \textbf{called,} \ \textbf{which} \ \textbf{triggers} \ \texttt{addLiquidity} \\ \textbf{ToFraxswap}$
- 2. The code checks if a Fraxswap pair exists. If not, it creates one with the predefined fee.
- 3. However, if an attacker front-runs the transaction and creates the pair with a higher fee, the contract proceeds to use this existing pair without checking the fee.
- Subsequent swaps and liquidity additions occur in a pool with an unintended fee, leading to miscalculations in getAmountOut and suboptimal liquidity provisioning.

Recommended Mitigation Steps

Add a fee check when using an existing Fraxswap pair. This ensures only pairs with the expected fee are used, preventing fee manipulation via front-running.