Ojo Yield Risk Engine V2

Security Review

Disclaimer

Security Review

Ojo Yield Risk Engine V2

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Security Review

Ojo Yield Risk Engine V2

Table of Contents

Summary

Security Review

Ojo Yield Risk Engine V2

Summary

Three Sigma audited Ojo YieldRiskEngineV2 in a 0.4 person week engagement. The audit was conducted on 12/06/2025.

Protocol Description

YieldRiskEngine is a Chainlink-compatible smart contract factory that helps Morpho risk curators cap the price of yield-bearing tokens based on the time-elapsed compounded yield. By using compounding math to model expected growth over time, it prevents oracle overvaluation, safeguarding markets from positive-price-swing attacks and reducing the risk of bad debt from inflated collateral.

Scope

Security Review

Ojo Yield Risk Engine V2

Scope

Filepath	nSLOC
src/OjoYieldRiskEngineV2.sol	87
Total	**87**

Methodology

Security Review

Ojo Yield Risk Engine V2

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit, we classify findings based on Immunefi's Vulnerability Severity Classification System (v2.3) as a guideline. The final classification considers both the potential impact of an issue, as defined in the referenced system, and its likelihood of being exploited. The following table summarizes the general expected classification according to impact and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Impact / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Security Review

Ojo Yield Risk Engine V2

Project Dashboard

Application Summary

Name	Ojo YieldRiskEngineV2
Repository	https://github.com/ojo-network/yield-risk-engine
Commit	a43e0ef
Language	Solidity
Platform	Ethereum

Engagement Summary

Timeline	12/06/2025
№ of Auditors	2
Review Time	0.4 person weeks

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
None	1	1	0

Category Breakdown

Suggestion	1
Documentation	0
Bug	0
Optimization	0
Good Code Practices	0

Risk Section

Security Review

Ojo Yield Risk Engine V2

Risk Section

No risk has been identified.

Findings

Security Review

Ojo Yield Risk Engine V2

Findings

3S-Ojo-N01

Missing **_disableInitializers** in constructor

Id	3S-Ojo-N01
Classification	None
Category	Suggestion
Status	Addressed in #420fab9.

Description

The **OjoYieldRiskEngineV2** contract is designed to be upgradeable, utilizing the **Initializable** contract from OpenZeppelin. Although the contract does not call **_disableInitializers** in its constructor, which is a best practice to prevent the implementation contract initialization, this omission has no impact in the current context. The contract works as intended, but addressing this would align with best practices for security and maintainability.

Recommendation

To adhere to best practices and prevent the implementation contract from being initialized, it is recommended to call **_disableInitializers** in the constructor of the **OjoYieldRiskEngineV2** contract.

```
+ constructor() {
+     _disableInitializers();
+ }
```