# CURRA

## SMART CONTRACTS REVIEW

zokyo

January 4th 2024 | v. 1.0

# Security Audit Score

## PASS

Zokyo Security has concluded that these smart contracts passed a security audit.

# # ZOKYO AUDIT SCORING CURRA

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

# HYPOTHETICAL SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 1 Critical issue: 1 resolved = 0 points deducted
- 0 High issues: 0 points deducted
- 0 Medium issues: 0 points deducted
- 2 Low issues: = 2 resolved = 0 points deducted
- 4 Informational issues: 4 resolved = 0 points deducted
- lack of test coverage = -5 points deducted

Thus, 100- 5 = 95

# TECHNICAL SUMMARY

This document outlines the overall security of the Curra smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Curra smart contracts codebase for quality, security, and correctness.

## Contract Status


There was 1 critical issue found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contracts but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Curra team put in place a bug bounty program to encourage further active analysis of the smart contracts.

# Table of Contents

# AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Curra repository:
Repo: https://github.com/curraprotocol/contracts

Last commit -758f742f742212ad53d722365f94e606d3a7d273

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- AuthorizedSenderRule.sol
- Curra.sol
- ERC1967Factory.sol
- Forwarder.sol
- Presets.sol
- RuleBase.sol
- WhitelistedAddressRule.sol

**During the audit, Zokyo Security ensured that the contract:**

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most resent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Curra smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contracts by industry leaders.

03 Thorough manual review of the codebase line by line.

# Executive Summary

The Zokyo team conducted a thorough evaluation of Curra's codebases. During the assessment, a critical issue was identified, alongside findings of low and informational significance. The Curra team promptly addressed these issues. Detailed descriptions of these findings can be found in the "Complete Analysis" section.

Curra's non-custodial nature is proven by the following features:
1. Rules that deployed and limit the operator are upgradable only by the user, so the operator cannot upgrade rules without the user.
2. User's funds can be forwarded/sweaped/flushed only to the user's wallet.

It shows Curra clients that Curra team has no access to their assets

# STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Curra team and the Curra team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

**Critical**

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

**High**

The issue affects the ability of the contract to compile or operate in a significant way.

**Medium**

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

**Low**

The issue has minimal impact on the contract’s ability to operate.

**Informational**

The issue has no impact on the contract’s ability to operate.

# COMPLETE ANALYSIS

## FINDINGS SUMMARY

| # | Title | Risk | Status |
|---|---|---|---|
| 1 | Anyone can upgrade an implementation | Critical | Resolved |
| 2 | Updated implementation of the ERC1967Factory | Low | Resolved |
| 3 | Unnecessary admin of proxy | Low | Resolved |
| 4 | Unused constant | Informational | Resolved |
| 5 | Unused constant | Informational | Resolved |
| 6 | Wrong solidity version | Informational | Resolved |
| 7 | Missing zero address validation | Informational | Resolved |

## CRITICAL-1 | RESOLVED

**Anyone can upgrade an implementation**

File: src/ERC1967Factory.sol
Details:
While the admin is not checked anymore, and both functions upgrade and upgradeAndCall have become public, anyone could call them and change the implementation.

**Recommendation:**
Make sure the upgradeAndCall function is either private or internal while the upgrade function is internal.

## LOW-1 | RESOLVED

**Updated implementation of the ERC1967Factory**

File: src/ERC1967Factory.sol
Details:
A newer implementation for the ERC1967Factory is available at its author's repository under commit hash: `9833cc764901b603398012b6f3b48930a7418f80`. It saves even more gas.

**Recommendation:**
Consider updating the ERC1967Factory library.

LOW-2 | RESOLVED

**Unnecessary admin of proxy**

File: src/ERC1967Factory.sol
Details:
While the original ERC1967Factory implementation uses admin to control the access to some functions in the contract itself, there is no need for this in your modified implementation. The deploy function is internal now, and access to it is being checked outside of the contract. Removing usage of the admin slot could save even more gas.

**Recommendation:**
Consider removing the admin slot from the contract.

INFORMATIONAL-1 | RESOLVED

**Unused constant**

File: src/ERC1967Factory.sol
Constant: _SALT_DOES_NOT_START_WITH_CALLER_ERROR_SELECTOR
Details:
The constant is declared but never used. You can save gas by removing it.

**Recommendation:**
Remove unsued constant

INFORMATIONAL-2 | RESOLVED

**Unused constant**

File: src/ERC1967Factory.sol
Constant: _ADMIN_CHANGED_EVENT_SIGNATURE
Details:
The constant is declared but never used. You can save gas by removing it.

**Recommendation:**
Remove unsued constant

## INFORMATIONAL-3 | RESOLVED

**Wrong solidity version**

File: src/ERC1967Factory.sol
Details:
The ERC1967Factory declares to use the solidity compiler version "^0.8.4" which allows older versions.

**Recommendation:**
Consider using the same pragma version for all solidity files in the project.

## INFORMATIONAL-4 | RESOLVED

**Missing zero address validation**

File: src/Forwarder.sol
Line: #17
Details:
The constructor accepts an argument "rule" but doesn't validate it to be a non-zero address. There's also no way to change this address later (it's immutable).

**Recommendation:**
Check that the address is not zero.

| | AuthorizedSenderRule.sol<br>Curra.sol<br>ERC1967Factory.sol<br>Forwarder.sol<br>Presets.sol<br>RuleBase.sol<br>WhitelistedAddressRule.sol |
|---|---|
| Re-entrancy | Pass |
| Access Management Hierarchy | Pass |
| Arithmetic Over/Under Flows | Pass |
| Unexpected Ether | Pass |
| Delegatecall | Pass |
| Default Public Visibility | Pass |
| Hidden Malicious Code | Pass |
| Entropy Illusion (Lack of Randomness) | Pass |
| External Contract Referencing | Pass |
| Short Address/ Parameter Attack | Pass |
| Unchecked CALL<br>Return Values | Pass |
| Race Conditions / Front Running | Pass |
| General Denial Of Service (DOS) | Pass |
| Uninitialized Storage Pointers | Pass |
| Floating Points and Precision | Pass |
| Tx.Origin Authentication | Pass |
| Signatures Replay | Pass |
| Pool Asset Security (backdoors in the underlying ERC-20) | Pass |

We are grateful for the opportunity to work with the Curra team.

**The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.**

Zokyo Security recommends the Curra team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo