# SMART CONTRACTS REVIEW

zokyo

May 13th 2024 | v. 1.0

# Security Audit Score

## PASS

Zokyo Security has concluded that this smart contract passed a security audit.

# # ZOKYO AUDIT SCORING RAIINMAKER

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.

# 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:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 1 Medium issue: 1 resolved = 0 points deducted
- 2 Low issues: 1 resolved and 1 acknowledged = -2 points deducted
- 1 Informational issue: 1 acknowledged = 0 points deducted

Thus, 100 - 2 = 98

# TECHNICAL SUMMARY

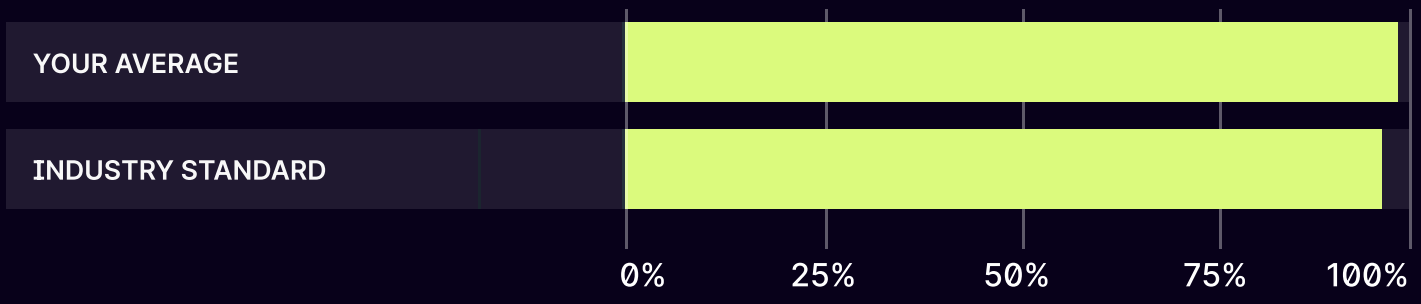
This document outlines the overall security of the Raiinmaker smart contract/s evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Raiinmaker smart contract/s codebase for quality, security, and correctness.

## Contract Status


There were 0 critical issues found during the review. (See Complete Analysis)

## Testable Code




97,70% of the code is testable, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract/s 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 Raiinmaker team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

# Table of Contents

# AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Raiinmaker repository:
Repo: https://github.com/Coiin-Blockchain/coiin-smart-contracts/blob/43d44be19eb7079adcf5408e32dfca764f8151f5/contracts/Coiin.sol

Last commit -a42db18a0de7e49b3dfbaec5a770d5bf56fc2a15

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

- Coiin.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 recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Raiinmaker smart contract/s. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Foundry testing framework. 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 contract/s by industry leaders. |
| 03 | Testing contract/s logic against common and uncommon attack vectors. |
| 04 | Thorough manual review of the codebase line by line. |

# Executive Summary

The Zokyo team has performed a security audit of the provided codebase. The contract submitted for auditing is well-crafted and organized. Detailed findings from the audit process are outlined in the "Complete Analysis" section.
The Coiin contract is a BEP20 standard token with advanced features. To mint Coiin tokens, users must interact with the 'withdraw' function, providing a valid signature and nonce. Signature validation is implemented using the EIP 712 standard through OpenZeppelin's ECDSA contract, ensuring mitigation of signature malleability issues. Signatures are time-bound, expiring after a specified timeframe to enhance security. Withdrawal limits are enforced by the 'checkWithdrawal' function, which verifies the amount withdrawn against the account, cluster, and overall withdrawal limits. Mint requests are queued within the 'withdraw' function and dequeued in 'checkWithdrawal' based on a pop count incremented when the mint request exceeds the withdrawal maximum period. An emergency pause function, exclusively set up by the owner, halts withdrawals during emergencies. Critical parameters, such as withdrawal limits and signer addresses, can only be updated by the contract owner, preventing unauthorized modifications. Furthermore, the contract is upgradeable using the UUPS proxy implementation, with proper overrides to prevent unauthorized upgrades.

# 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 Raiinmaker team and the Raiinmaker 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 | Withdraw Message Would Be Replayable On A Different Chain | Medium | Resolved |
| 2 | Unsafe Transfer | Low | Resolved |
| 3 | Centralization Risks | Low | Acknowledged |
| 4 | Paused Interval Does Not Account For The withdrawMaxPeriod | Informational | Acknowledged |

## MEDIUM-1 | RESOLVED

### Withdraw Message Would Be Replayable On A Different Chain

The message digest calculated for the withdraw includes msgSender, amount, expires, nonce, and address(this), and then the withdrawSigner is recovered. But this message does not include a chainId due to which if the contract gets deployed on another chain then the same message would be valid there with a different chainId and the user would essentially withdraw twice.

**Recommendation:**
Add a chainId value to the message digest calculation.

## LOW-1 | RESOLVED

### Unsafe Transfer

The rescue() functionality is used to rescue mistakenly transferred funds to the contract and this function transfers the token to the owner, but the transfer used here is an unsafe one, i.e. the transfer returns a bool for ERC20 implementations and that value is not checked here, therefore the transfer might fail silently and the tx would still be successful.

**Recommendation:**
Use safeTransfer instead.

## LOW-2 | ACKNOWLEDGED

**Centralization Risks**

All privileged and critical state changing functions are protected with a onlyOwner modifier, it should be made sure that the owner is a multisig with a timelock. Handling all the power to a single address will make the system centralised and prone to failure.

**Recommendation:**
Ensure the owner is a multisig with a timelock with each key residing on a different server.

The client mitigates centralization risks by using multi-signature (multisig) accounts, ensuring that no single user has complete control.

## INFORMATIONAL-1 | ACKNOWLEDGED

**Paused Interval Does Not Account For The withdrawMaxPeriod**

Withdrawals can be paused by the owner using the pauseWithrawals() function, therefore it is possible that during a paused state all user's withdrawal timeline might go beyond the withdrawMaxPeriod and all the user's mint would be dequeued making the withdrawMintHistory empty when the state gets unpaused.

**Recommendation:**
Account for the paused state appropriately.

| | Coiin.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 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 |

# CODE COVERAGE AND TEST RESULTS FOR ALL FILES

## Tests written by Zokyo Security

As a part of our work assisting Raiinmaker in verifying the correctness of their contract/s code, our team was responsible for writing integration tests using the Foundry testing framework.

The tests were based on the functionality of the code, as well as a review of the Raiinmaker contract/s requirements for details about issuance amounts and how the system handles these.

```
Ran 13 tests for test/Coiin.t.sol:CoiinTest
[PASS] testFuzz_setWithdrawAccountLimits(uint256,uint256,address) (runs: 256, μ: 31954, ~: 32311)
[PASS] testFuzz_setWithdrawAccountLimits(uint256,uint256,uint256,address) (runs: 256, μ: 36833, ~: 37313)
[PASS] testFuzz_setWithdrawLimits(uint256,uint256,uint256,uint256,uint256,uint256,uint256,address) (runs: 256, μ: 56649, ~: 57924)
[PASS] testFuzz_setWithdrawMaxLimits(uint256,uint256,address) (runs: 256, μ: 31859, ~: 32216)
[PASS] test_InitialBalances() (gas: 17788)
[PASS] test_Initializations() (gas: 59155)
[PASS] test_deposit() (gas: 38489)
[PASS] test_getWithdrawLimits() (gas: 54410)
[PASS] test_isInCluster() (gas: 187)
[PASS] test_pauseWithdrawals() (gas: 52605)
[PASS] test_rescue() (gas: 79759)
[PASS] test_setWithdrawSigner() (gas: 38158)
[PASS] test_withdraw() (gas: 445487)

Suite result: ok. 13 passed; 0 failed; 0 skipped; finished in 29.65ms (100.38ms CPU time)

Ran 1 test suite in 258.94ms (29.65ms CPU time): 13 tests passed, 0 failed, 0 skipped (13 total tests)
```

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

| FILE | % STMTS | % BRANCH | % FUNCS | % LINES | %UNCOVERED LINES |
|---|---|---|---|---|---|
| Coiin.sol | 97.70 | 92.86 | 94.12 | 98.63 | |
| **All Files** | **97.70** | **92.86** | **94.12** | **98.63** | |

We are grateful for the opportunity to work with the Raiinmaker 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 Raiinmaker team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo