RAILGUN_

RAILGUN_

SMART CONTRACT AUDIT

zokyo

April 21th, 2022 | v. 1.0

Security Audit Score

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

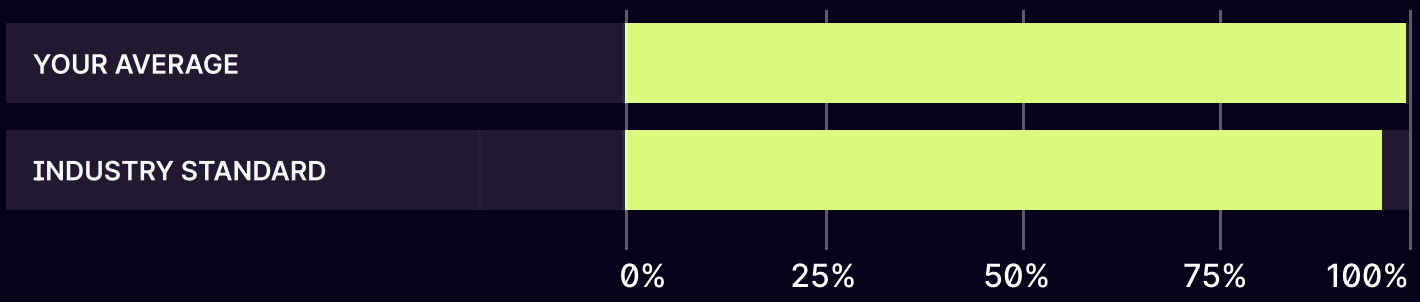
TECHNICAL SUMMARY

This document outlines the overall security of the RAIL Token smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

Testable Code



The testable code is significant for the ensurance of contracts security.

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

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from Etherscan.

Repository: https://etherscan.io/address/0xe76c6c83af64e4c60245d8c7de953df673a7a33d#c ode#F1#L1

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- Rail.sol

Additional requirements to check:

- Behavior of `transfer` method is equivalent to standard ERC20 behaviour in ALL scenarios after ANTI_BOT_LOCKTIME has passed
  Check result – confirmed.
- - ANTI_BOT_LOCKTIME has passed and there is no way for the `transfer` method to revert back to a non-standard ERC20 transfer behavior
  Check result – confirmed.
- - governanceMint method is only callable by the governance contract and can mint up to a max supply of cap
  Check result – confirmed.
- - _mint() can mint up to a max supply of cap and can be called by method governanceMint and during deploy of smart contract
  Check result – confirmed.

Throughout the review process, care was taken to ensure that the contract:

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of RAIL Token smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite. 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	Testing contract logic against common and uncommon attack vectors.
04	Thorough, manual review of the codebase, line-by-line.

Executive Summary

There were 0 critical issues found during the audit.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. 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

INFORMATIONAL | UNRESOLVED

Solidity version should be updated.
Best practices for Solidity development and auditors standard checklist requires strict and explicit usage of the latest stable version of Solidity.

Recommendation:
Consider updating to “pragma solidity 0.8.11;”.

	Rail.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

We are grateful to have been given the opportunity to work with RAIL Token.

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

Zokyo's Security Team recommends that RAIL Token put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo