RAILGUN_

SMART CONTRACT AUDIT

zokyo

Nov 3th, 2021 | 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

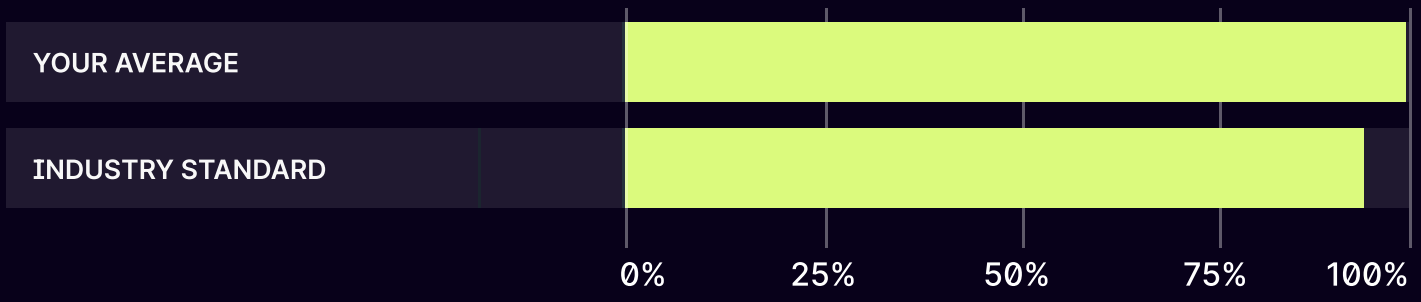
This document outlines the overall security of the RAILGUN smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

There were no critical issues found during the audit.

Testable Code



The testable code is 99.26%, 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, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the RAILGUN 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 the RAILGUN repository.

Repository: https://github.com/Railgun-Privacy/contract/commit/d2c63577ddd8310c87dced0d549cf9505b372111

Contracts under the scope:

- Delegator;
- Deployer;
- Staking;
- Voting;
- Commitments;
- Globals;
- Poseidon;
- RailgunLogic;
- Snark;
- TokenWhitelist;
- Verifier;
- Proxy;
- ProxyAdmin;
- Distributor;
- Multisend;
- VestLock;
- Treasury.

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

- Implements and adheres to existing Token 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 gas, 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 RAILGUN 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 using the Truffle 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 contracts by industry leaders.
03	Testing contract logic against common and uncommon attack vectors	04	Thorough, manual review of the codebase, line-by-line.

Summary

The Zokyo team has conducted a security audit of the given codebase. The contracts provided for an audit are well written and structured. All the findings spotted within the auditing process are presented in this document.

There were no critical issues found during the auditing process. Just a couple of informational issues and one issue with a low severity level were found. All the issues are left unresolved as the contracts are already deployed on the mainnet. However, it is worth mentioning that all of the mentioned issues have no major security or operational risk.

Based on the audit result we can give a score of 99% to the provided codebase.

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

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

Token receiver address defined as payable

LOW | UNRESOLVED

In the treasury contract function transferERC20() you declared the address of the tokens receiver as payable. Addresses declared payable can receive ether.

Recommendation:
Remove payable for a token receiver.

Misleading comment at Treasury.sol

INFORMATIONAL | UNRESOLVED

In the treasury contract function transferERC20() is used for transferring tokens.

Recommendation:
Change comment in the notice from "Transfers ETH to specified address" to "Transfers ERC20 token to specified address".

Misleading comment at VestLock.sol

INFORMATIONAL | UNRESOLVED

In the VestLock contract function transferERC20() is used for transferring tokens.

Recommendation:
Change comment in the notice from "Transfers ETH to specified address" to "Transfers ERC20 token to specified address".

Typo at TokenWhitelist.sol

INFORMATIONAL | UNRESOLVED

In the TokenWhitelist event, TokeDelisting is misspelled.

Recommendation:
Change event to TokenDelisting at line 22 and line 78.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests are written by the RAILGUN team

Code Coverage

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

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	Uncovered Lines
contracts\	88.94	77.97	77.50	89.29	
Delegator.sol	55.56	42.86	100.00	60.00	
Deployer.sol	100.00	100.00	100.00	100.00	
Staking.sol	94.06	82.14	72.73	93.94	...185, 195, 285
Voting.sol	93.85	83.33	70.00	92.42	...127, 351, 353
logic\	97.93	57.14	89.66	98.63	
Commitments.sol	91.30	62.50	83.33	93.48	230, 235, 238
Globals.sol	100.00	100.00	100.00	100.00	
Poseidon.sol	100.00	100.00	0.00	100.00	
RailgunLogic.sol	100.00	57.89	100.00	100.00	
Snark.sol	96.97	50.00	100.00	100.00	
TokenWhitelist.sol	100.00	50.00	100.00	100.00	
Verifier.sol	99.34	62.50	100.00	99.33	352
proxy\	83.33	64.29	85.71	82.35	
Proxy.sol	84.00	64.29	82.76	82.76	...117, 120, 123
ProxyAdmin.sol	80.00	100.00	80.00	80.00	29
token\	94.74	100.00	83.33	95.00	
Distributor.sol	100.00	100.00	100.00	100.00	
Multisend.sol	100.00	100.00	100.00	100.00	
VestLock.sol	91.67	100.00	77.78	92.31	98
treasury	100.00	100.00	75.00	100.00	
Treasury.sol	100.00	100.00	75.00	100.00	
All files	**93.72**	**70.10**	**82.83**	**94.13**	

Test Results

Logic/Whitelist

- ✓ Should initialize whitelist with passed values (132ms)
- ✓ Should add an address to whitelist (129ms)
- ✓ Should remove address from whitelist (117ms)

Logic/Commitments

- ✓ Should initialize the tree correctly (57ms)
- ✓ Should update the tree correctly (2060ms)
- ✓ Should generate commitment correctly (835ms)

Logic/RailgunLogic

- ✓ Should verify proofs (20691ms)
- ✓ Should deposit token correctly (20771ms)
- ✓ Should collect treasury fees correctly (47217ms)
- ✓ Should deposit with 2 outputs correctly (24052ms)
- ✓ Should deposit with 3 outputs correctly (24024ms)
- ✓ Should deposit and withdraw (46906ms)
- ✓ Should deposit, do an internal transaction, and withdraw (69847ms)
- ✓ Should transact with large circuit (130299ms)
- ✓ Should deposit and generate commitments correctly (1051ms)
- ✓ Should be able to spend from generated commitment (24743ms)

Proxy/Proxy

- ✓ Should deploy as paused (126ms)
- ✓ Shouldn't allow unpausing unless the implementation is set
- ✓ Should upgrade and unpause (210ms)
- ✓ Should unpause and pause again (229ms)
- ✓ Should upgrade unpause and upgrade again (319ms)

Proxy/Proxy

- ✓ Should upgrade and unpause (135ms)
- ✓ Should unpause and pause again (208ms)
- ✓ Should upgrade unpause and upgrade again (325ms)

Treasury/Treasury

- ✓ Should transfer ETH (63ms)
- ✓ Should transfer ERC20 (130ms)

Token/Vesting

- ✓ Should set up vesting (475ms)

Token/Multisend

- ✓ Should multisend (1599ms)

Governance/Delegator

- ✓ Should set permissions (57ms)
- ✓ Should be able to call a function with permission (80ms)
- ✓ Should be able to call the function with wildcard contract permission (64ms)
- ✓ Should be able to call the function with wildcard function permission (94ms)
- ✓ Should intercept calls to self correctly (70ms)

Governance/Deployer

- ✓ Should deploy contracts at expected address (104ms)

Governance/Staking

- ✓ Should count intervals properly (38ms)
- ✓ Should return correct snapshot regardless of hint (14317ms)
- ✓ Should go through stake lifecycle correctly (330ms)
- ✓ Should delegate and snapshot correctly (379ms)

Governance/Voting

- ✓ Should go through vote lifecycle correctly (754ms)
- ✓ Should not be able to sponsor after the sponsor window (54ms)
- ✓ Should not execute failed proposal (177ms)
- ✓ Should execute proposals correctly (193ms)

42 passing (8m)

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security team

As part of our work assisting RAILGUN in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the RAILGUN contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

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

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	Uncovered Lines
contracts\	100.00	96.61	100.00	100.00	
Delegator.sol	100.00	100.00	100.00	100.00	
Deployer.sol	100.00	100.00	100.00	100.00	
Staking.sol	100.00	92.86	100.00	100.00	
Voting.sol	100.00	100.00	100.00	100.00	
logic\	98.97	77.14	93.10	100.00	
Commitments.sol	95.65	75.00	100.00	100.00	
Globals.sol	100.00	100.00	100.00	100.00	
Poseidon.sol	100.00	100.00	0.00	100.00	
RailgunLogic.sol	100.00	84.21	100.00	100.00	
Snark.sol	96.97	50.00	100.00	100.00	
TokenWhitelist.sol	100.00	100.00	100.00	100.00	
Verifier.sol	100.00	75.00	100.00	100.00	
proxy\	96.97	100.00	92.86	94.12	
Proxy.sol	96.00	100.00	88.89	93.10	101, 108
ProxyAdmin.sol	100.00	100.00	100.00	100.00	
token\	100.00	100.00	91.67	100.00	
Distributor.sol	100.00	100.00	100.00	100.00	
Multisend.sol	100.00	100.00	100.00	100.00	

VestLock.sol	100.00	100.00	88.89	100.00
treasury	100.00	100.00	75.00	100.00
Treasury.sol	100.00	100.00	75.00	100.00
All files	**99.26**	**90.20**	**94.95**	**99.63**

Test Results

Delegator contract

- ✓ Should initialize owner correct
- ✓ Should set permission correct (70ms)
- ✓ Should revoke permission correct (135ms)
- ✓ Should call the function with permission (78ms)
- ✓ Should revert call function if not has permission (127ms)
- ✓ Should set permission by call contract function correct (89ms)
- ✓ Should transfer ownership by call contract function correct (50ms)
- ✓ Should renounce ownership by call contract function correct (59ms)
- ✓ Should intercept calls to self correctly

Deployer contract

- ✓ Should initialize owner correct
- ✓ Should deploy correctly

Staking contract

- ✓ Should return initial total voting power in system correct
- ✓ Should stake correct (92ms)
- ✓ Shouldn't stake zero amount
- ✓ Should unlock correct (186ms)
- ✓ Shouldn't unlock twice (286ms)

- ✓ Should claim correct (172ms)
- ✓ Shouldn't claim twice (178ms)
- ✓ Should delegate correct (259ms)
- ✓ Shouldn't delegate after unlock (158ms)
- ✓ Shouldn't delegate to zero address (112ms)
- ✓ Should undelegated correct (237ms)
- ✓ Should return stakesLength correct (371ms)

snapshot

- ✓ Should fail intervalAtTime when the requested time is below the deploy time value
- ✓ Should return latestGlobalsSnapshotInterval correct
- ✓ Should return latestAccountSnapshotInterval correct
- ✓ Should return accountSnapshotLength correct
- ✓ Should return globalsSnapshotLength correct
- ✓ Should return accountSnapshot correct
- ✓ Should return globalsSnapshot correct
- ✓ Should return globalsSnapshotAt correct (4022ms)
- ✓ Should return accountSnapshotAt correct (4312ms)

Voting contract

- ✓ Should initial proposalsLength be correct
- ✓ Should getActions correct (60ms)
- ✓ Should getSponsored correct (136ms)
- ✓ Should createProposal correct (85ms)
- ✓ Should revert createProposal with no actions
- ✓ Should sponsorProposal correct (123ms)
- ✓ Should revert sponsorProposal when the proposal went to vote (157ms)
- ✓ Should revert sponsorProposal when sponsor window passed (46ms)
- ✓ Should revert sponsorProposal when not enough voting power (112ms)
- ✓ Should unsponsorProposal correct (134ms)
- ✓ Should revert unsponsorProposal when the proposal went to vote (201ms)
- ✓ Should revert unsponsorProposal when sponsor window passed (90ms)
- ✓ Should revert unsponsorProposal when request more than sponsored (93ms)
- ✓ Should callVote correct (161ms)
- ✓ Should revert callVote when the proposal went to vote (217ms)
- ✓ Should revert callVote when sponsor window passed (45ms)
- ✓ Should revert callVote when not met sponsorship threshold (90ms)
- ✓ Should vote for correct (233ms)
- ✓ Should vote against correct (206ms)

- ✓ Should revert vote if the voting window was not opened (161ms)
- ✓ Should revert vote if call vote was not called for the proposal (69ms)
- ✓ Should revert vote against if yay window was closed (212ms)
- ✓ Should revert vote for if nay window was closed (181ms)
- ✓ Should revert vote if there is not enough voting power (166ms)
- ✓ Should executeProposal correct (291ms)
- ✓ Should revert execute proposal if call vote was not called for the proposal (145ms)
- ✓ Should revert executeProposal if a quorum hasn't been reached (219ms)
- ✓ Should revert executeProposal if the proposal hasn't been passed (220ms)
- ✓ Should revert executeProposal if execution window hasn't been started (222ms)
- ✓ Should revert executeProposal if execution window has been closed (267ms)
- ✓ Should revert executeProposal if a proposal has already been executed (297ms)
- ✓ Should revert executeProposal if proposal action is reverted (287ms)

Commitments contract

- ✓ should initialize merkle tree correct
- ✓ should update the tree correctly
- ✓ should revert adding a new commitment if the leaf is invalid
- ✓ should add new generated commitment correctly
- ✓ should create a new tree correct

Commitments contract

- ✓ Should initialize railgun logic correct (151ms)
- ✓ Should change treasury correct (217ms)
- ✓ Should revert change treasury by not owner
- ✓ Should change fee correct (125ms)
- ✓ Should revert change fee by not owner
- ✓ Should verify proofs (21943ms)
- ✓ Should deposit whitelisted token correct (21641ms)
- ✓ Should deposit and withdraw whitelisted token correct (49991ms)
- ✓ Should revert deposit when adaptID address is incorrect token correct (22057ms)
- ✓ Should revert deposit when the merkle root is incorrect (21974ms)
- ✓ Should revert deposit not whitelisted token (22395ms)
- ✓ Should collect treasury fees correctly (49711ms)
- ✓ Should generateDeposit and withdraw token from the generated commitments (25316ms)
- ✓ Should revert generateDeposit if fee not paid (81ms)
- ✓ Should revert generateDeposit if deposited amount is zero (54ms)
- ✓ Should revert generateDeposit if deposit not whitelisted token (91ms)

TokenWhitelist contract

- ✓ should initialize correctly
- ✓ should add whitelisted token correct (159ms)
- ✓ should revert add whitelisted token if caller, not the owner
- ✓ should remove whitelisted token correct (179ms)
- ✓ should revert remove whitelisted token if caller, not owner

TokenWhitelist contract

- ✓ should verifyProof correct (22972ms)
- ✓ should verifyProof with 10 nullifiers correct (41842ms)
- ✓ should revert verifyProof with incorrect nullifier

TokenWhitelist contract

- ✓ Should upgrade and emit event correct (186ms)
- ✓ Should not set new implementation as non-contract address (45ms)
- ✓ Should pause and emit event correct (221ms)
- ✓ Should unpause and emit event correct (234ms)
- ✓ Should revert unpause before implementation is set
- ✓ Should transfer ownership correct
- ✓ Should revert transfer ownership by not the owner (237ms)

ProxyAdmin contract

- ✓ Should upgrade correct (81ms)
- ✓ Should pause and emit event correct (157ms)
- ✓ Should unpause and emit event correct (187ms)
- ✓ Should transfer ownership correct

Distributor contract

- ✓ Should initialize vesting implementation correct
- ✓ Should create vest lock correct (79ms)
- ✓ Should not create vest lock by not the owner (62ms)

Multisend contract

- ✓ Should multisend correct to one address (76ms)
- ✓ Should multisend correct to multiple addresses (106ms)

VestLock contract

- ✓ Should initialize owner correct
- ✓ Should initialize release time correct
- ✓ Should delegate correct (204ms)
- ✓ Should stake correct (134ms)
- ✓ Should unlock correct (115ms)
- ✓ Should claim correct (248ms)
- ✓ Should receive ETH correct (45ms)
- ✓ Should transferETH correct (78ms)
- ✓ Shouldn't transferETH while not achieved release time
- ✓ Should transferERC20 correct (45ms)
- ✓ Shouldn't transferERC20 while not achieved release time

Treasury contract

- ✓ Should initialize owner correct
- ✓ Should transfer ETH (58ms)
- ✓ Should revert transfer ETH if caller, not owner
- ✓ Should transfer ERC20 (173ms)
- ✓ Should revert transfer ERC20 if caller, not owner (137ms)

125 passing (7m)

We are grateful to have been given the opportunity to work with the RAILGUN team.

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

zokyo