HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Right To Privacy
**Date**: November 2nd, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Right to Privacy. |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | Privacy System Platform |
| **Platform** | Ethereum / Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Repository** | https://github.com/Railgun-Privacy/contract |
| **Commit** | d2c63577ddd8310c87dced0d549cf9505b372111 |
| **Technical Documentation** | NO |
| **JS tests** | YES |
| **Website** | righttoprivacy.foundation |
| **Timeline** | 25 OCTOBER 2021 - 02 NOVEMBER 2021 |
| **Changelog** | 02 NOVEMBER 2021 - INITIAL AUDIT |

## Table of contents

## Introduction

Hacken OÜ (Consultant) was contracted by Right to Privacy (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between October 25th, 2021 - November 2nd, 2021.

## Scope

The scope of the project is smart contracts in the repository:

**Repository:**
 https://github.com/Railgun-Privacy/contract
**Commit:**
 d2c63577ddd8310c87dced0d549cf9505b372111
**Technical Documentation:** No
**JS tests:** Yes (included: “/test/”)
**Contracts:**
 governance/Delegator.sol
 governance/Deployer.sol
 governance/Staking.sol
 governance/Voting.sol
 logic/Commitments.sol
 logic/Globals.sol
 logic/Poseidon.sol
 logic/RailgunLogic.sol
 logic/Snark.sol
 logic/TokenWhitelist.sol
 logic/Verifier.sol
 proxy/Proxy.sol
 proxy/ProxyAdmin.sol
 teststubs/governance/Getter.sol
 teststubs/governance/GovernanceTarget.sol
 teststubs/governance/StakingStub.sol
 teststubs/logic/CommitmentsStub.sol
 teststubs/logic/TokenWhitelistStub.sol
 teststubs/proxy/ProxyTarget.sol
 teststubs/TokenStubs.sol
 token/Distributor.sol
 token/Multisend.sol
 token/VestLock.sol
 treasury/Treasury.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

| Category | Check Item |
|---|---|
| Code review | Reentrancy |
| | Ownership Takeover |
| | Timestamp Dependence |
| | Gas Limit and Loops |
| | DoS with (Unexpected) Throw |
| | DoS with Block Gas Limit |
| | Transaction-Ordering Dependence |
| | Style guide violation |
| | Costly Loop |
| | ERC20 API violation |
| | Unchecked external call |
| | Unchecked math |
| | Unsafe type inference |
| | Implicit visibility level |
| | Deployment Consistency |
| | Repository Consistency |
| | Data Consistency |
| Functional review | Business Logics Review |
| | Functionality Checks |
| | Access Control & Authorization |
| | Escrow manipulation |
| | Token Supply manipulation |
| | Assets integrity |
| | User Balances manipulation |
| | Data Consistency manipulation |
| | Kill-Switch Mechanism |
| | Operation Trails & Event Generation |

## Executive Summary

According to the assessment, the Customer's smart contracts are secured but some functions could run out of gas.

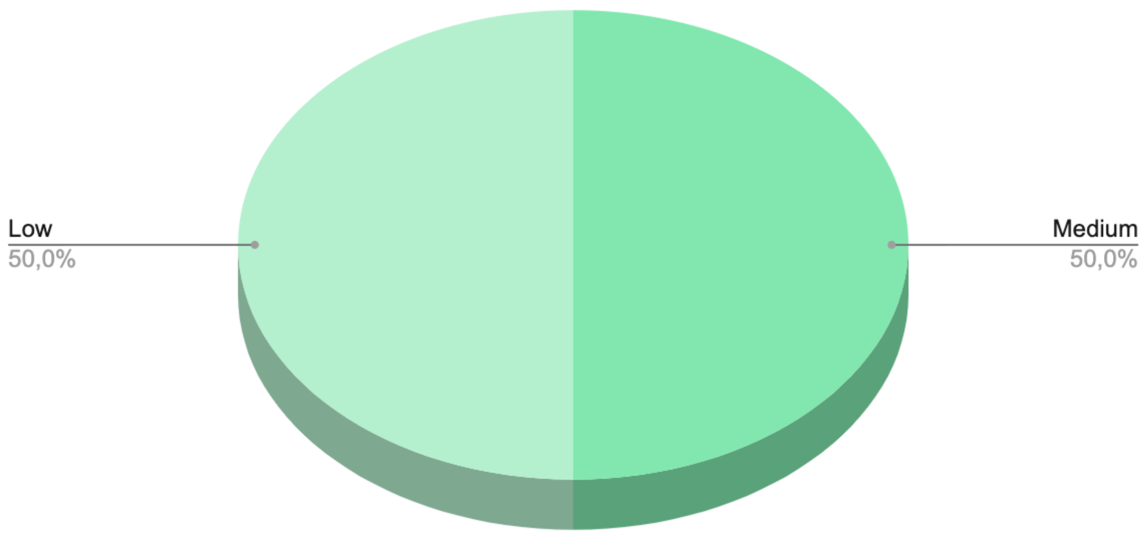
Insecure Poor secured Secured Well-secured

You are here

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **2** medium and **2** low severity issues.

*Graph 1. The distribution of vulnerabilities after the audit.*



## Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution |

## Audit overview

### Critical

No critical issues were found.

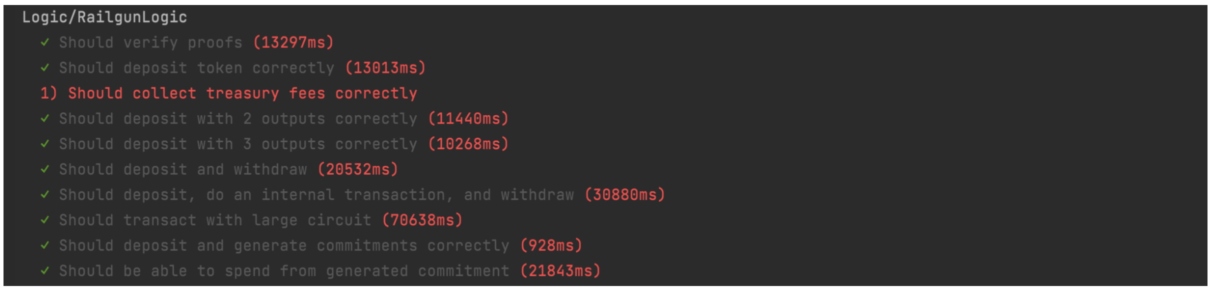
### High

No high severity issues were found.

### Medium

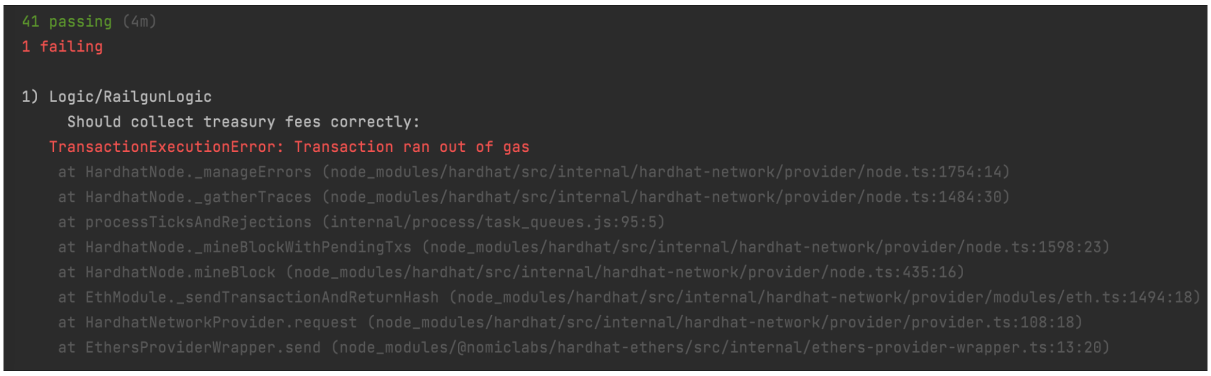
1. One test failed

While 41 tests are passing but 1 is failing. It fails with the “Out of Gas” message which means your logic could be too complicated and overloaded with loops, maths, and external calls.



```
Logic/RailgunLogic
  ✓ Should verify proofs (13297ms)
  ✓ Should deposit token correctly (13013ms)
  1) Should collect treasury fees correctly
  ✓ Should deposit with 2 outputs correctly (11440ms)
  ✓ Should deposit with 3 outputs correctly (10268ms)
  ✓ Should deposit and withdraw (20532ms)
  ✓ Should deposit, do an internal transaction, and withdraw (30880ms)
  ✓ Should transact with large circuit (70638ms)
  ✓ Should deposit and generate commitments correctly (928ms)
  ✓ Should be able to spend from generated commitment (21843ms)
```

....



```
41 passing (4m)
1 failing

1) Logic/RailgunLogic
     Should collect treasury fees correctly:
   TransactionExecutionError: Transaction ran out of gas
    at HardhatNode._manageErrors (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1754:14)
    at HardhatNode._gatherTraces (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1484:30)
    at processTicksAndRejections (internal/process/task_queues.js:95:5)
    at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1598:23)
    at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)
    at EthModule._sendTransactionAndReturnHash (node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1494:18)
    at HardhatNetworkProvider.request (node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:108:18)
    at EthersProviderWrapper.send (node_modules/@nomiclabs/hardhat-ethers/src/internal/ethers-provider-wrapper.ts:13:20)
```

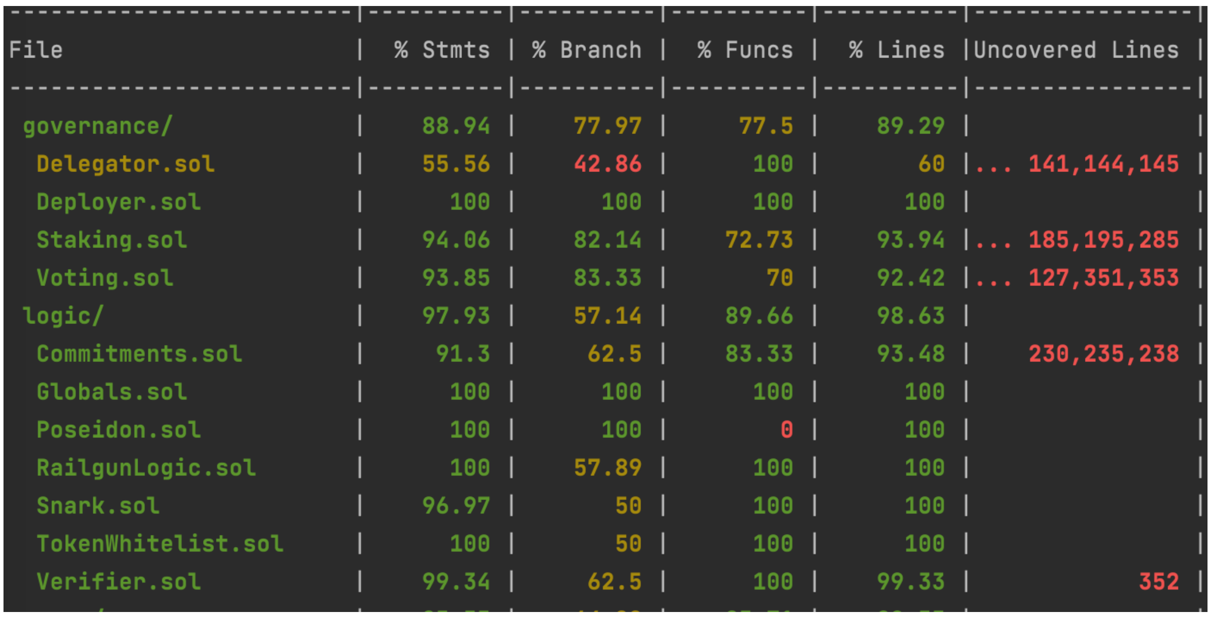
**Contracts**: RailgunLogic.sol

**Recommendation**: Please check the functionality of the RailgunLogic and make sure you’re not running out of gas and all tests are passing.

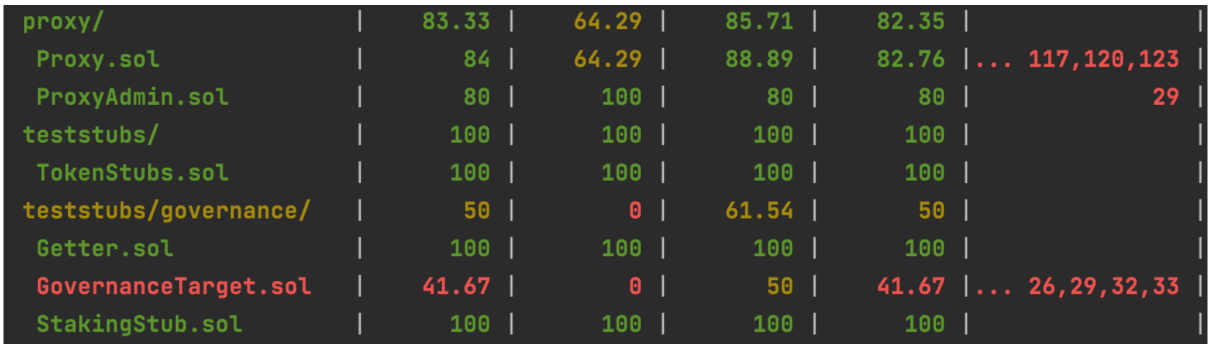
2. Too low test coverage

Global test coverage is about 68% for code branches, while the main RailgunLogic contract is covered only for 57.89% of logic branches.

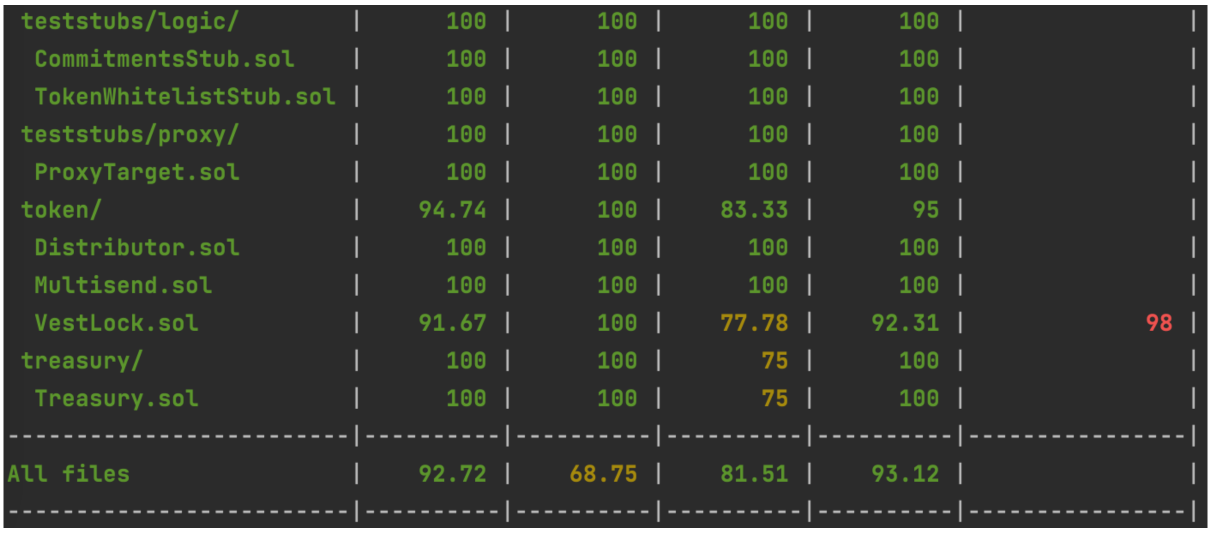
The recommended coverage is minimum 95% for branches, while it should be definitely 100% for the main logic contracts.



```
-------------------------|----------|----------|----------|----------|----------------|
File                     |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-------------------------|----------|----------|----------|----------|----------------|
 governance/             |    88.94 |    77.97 |     77.5 |    89.29 |                |
  Delegator.sol          |    55.56 |    42.86 |      100 |       60 |... 141,144,145 |
  Deployer.sol           |      100 |      100 |      100 |      100 |                |
  Staking.sol            |    94.06 |    82.14 |    72.73 |    93.94 |... 185,195,285 |
  Voting.sol             |    93.85 |    83.33 |       70 |    92.42 |... 127,351,353 |
 logic/                  |    97.93 |    57.14 |    89.66 |    98.63 |                |
  Commitments.sol        |     91.3 |     62.5 |    83.33 |    93.48 |    230,235,238 |
  Globals.sol            |      100 |      100 |      100 |      100 |                |
  Poseidon.sol           |      100 |      100 |        0 |      100 |                |
  RailgunLogic.sol       |      100 |    57.89 |      100 |      100 |                |
  Snark.sol              |    96.97 |       50 |      100 |      100 |                |
  TokenWhitelist.sol     |      100 |       50 |      100 |      100 |                |
  Verifier.sol           |    99.34 |     62.5 |      100 |    99.33 |            352 |
```



```
 proxy/                  |    83.33 |    64.29 |    85.71 |    82.35 |                |
  Proxy.sol              |       84 |    64.29 |    88.89 |    82.76 |... 117,120,123 |
  ProxyAdmin.sol         |       80 |      100 |       80 |       80 |             29 |
 teststubs/              |      100 |      100 |      100 |      100 |                |
  TokenStubs.sol         |      100 |      100 |      100 |      100 |                |
 teststubs/governance/   |       50 |        0 |    61.54 |       50 |                |
  Getter.sol             |      100 |      100 |      100 |      100 |                |
  GovernanceTarget.sol   |    41.67 |        0 |       50 |    41.67 |... 26,29,32,33 |
  StakingStub.sol        |      100 |      100 |      100 |      100 |                |
```



```
 teststubs/logic/        |      100 |      100 |      100 |      100 |                |
  CommitmentsStub.sol    |      100 |      100 |      100 |      100 |                |
  TokenWhitelistStub.sol |      100 |      100 |      100 |      100 |                |
 teststubs/proxy/        |      100 |      100 |      100 |      100 |                |
  ProxyTarget.sol        |      100 |      100 |      100 |      100 |                |
 token/                  |    94.74 |      100 |    83.33 |       95 |                |
  Distributor.sol        |      100 |      100 |      100 |      100 |                |
  Multisend.sol          |      100 |      100 |      100 |      100 |                |
  VestLock.sol           |    91.67 |      100 |    77.78 |    92.31 |             98 |
 treasury/               |      100 |      100 |       75 |      100 |                |
  Treasury.sol           |      100 |      100 |       75 |      100 |                |
-------------------------|----------|----------|----------|----------|----------------|
All files                |    92.72 |    68.75 |    81.51 |    93.12 |                |
-------------------------|----------|----------|----------|----------|----------------|
```

**Contracts**: Commitments.sol, RailgunLogic.sol, Snark.sol, TokenWhitelist.sol, Verifier.sol, Delegator.sol

**Recommendation**: Please make sure you have at least 95% of overall code branches covered by tests and to have 100% branches coverage for the main business logic code.

■ **Low**

1. Missing zero address validation

Accidentally setting "_vestLockImplementation" to zero-address could lead to contract out of work because it doesn't have the ability to update it in any way.

**Contracts**: Distributor.sol

**Function**: constructor

**Recommendation**: Please check "_vestLockImplementation" for being zero address.

2. A public function that could be declared external

**public** functions that are never called by the contract should be declared **external** to save gas.

**Contracts**: Multisend.sol

**Function**: multisend

**Recommendation**: Use the external attribute for functions never called from the contract.

## Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **2** medium and **2** low severity issues.

## Disclaimers

### Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

### Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.