# Railgun Security Scan Results

## by Pessimistic

This is not a security audit

This report is public

May 28, 2024

Abstract ..... 2
Disclaimer ..... 2
Summary ..... 2
Scan process ..... 3
Workflow ..... 3
Issue categories ..... 3
Scan results ..... 4
Discovered Issues ..... 5
Unchecked return values of ERC20 calls ..... 5
Code quality ..... 5
Missing events ..... 5

# Abstract

This report considers the security of smart contracts of the Railgun protocol. Our task is to find and describe security issues using the static-analysis tools Slither and Slitherin and help resolve them.

The work is financially covered by the Arbitrum Foundation grant.

# Disclaimer

Current work does not give any warranties on the security of the code. It is not an audit or its replacement. Performing this scan, we focused on finding as many crucial issues as possible rather than making sure that the protocol was entirely secure. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

# Summary

In this report, we described issues found in smart contracts of the Railgun protocol.

We scanned the codebase and manually rejected or verified all automated findings, revealing three relevant issues.

The developers commented one issue.

The entire process is described in the section below.

# Scan process

Under the Arbitrum Foundation grant, we researched and developed Arbitrum-specific detectors. They became publicly available with Slitherin v0.6.0 release.

## Workflow

This work consisted of five stages:

1. For the scan, we were provided with the Railgun project on a public GitHub repository, commit df709a58a9af4388c3288afabafe838d40958732. This commit is not present in the current state of the repository.
2. For the analysis of the protocol, we launched Slither v0.10.1 and Slitherin v0.6.0 on the provided codebase.
3. One auditor manually checked (rejected or accepted) all findings reported by the tools. The second auditor verified this work. We shared all relevant issues with the protocol developers and answered their questions.
4. The developers reviewed the findings, updated the code accordingly, and gave comments on issues they do not intend to fix. We reviewed the fixes and found no new issues.
5. We prepared this final report summarizing all the issues and comments from the developers.

## Issue categories

Within the confines of this work, we were looking for:

- Arbitrum-specific problems;
- Standard vulnerabilities like re-entrancy, overflow, arbitrary calls, etc;
- Non-compliance with popular standards like ERC20 and ERC721;
- Some access control problems;
- Integration issues with some popular DeFi protocols;
- A wide range of code quality and gas efficiency improvement opportunities.

This scan does not guarantee that these issues are not present in the codebase.

## Scan results

| Issue category | Number of detectors | Status |
|---|---|---|
| Compilation | 1 | Passed |
| Arbitrum Integration | 3 | Passed |
| AAVE Integration | 1 | Passed |
| Uniswap V2 Integration | 7 | Passed |
| OpenZeppelin | 2 | Passed |
| ERC-20 | 7 | Passed |
| ERC-721 | 2 | Passed |
| Known Bugs | 15 | Passed |
| Access Control | 3 | Passed |
| Arbitrary Call | 5 | Passed |
| Re-entrancy | 6 | Passed |
| Weak PRNG | 2 | Passed |
| Upgradability | 2 | Passed |
| Ether Handling | 3 | Passed |
| Low-level Calls | 2 | Passed |
| Assembly | 2 | Passed |
| Inheritance | 3 | Passed |
| Arithmetic | 2 | Passed |
| Old Solidity Versions Bugs | 10 | Passed |
| Code Quality | 15 | 1 issue found |
| Best Practices | 4 | 1 issue found |
| Gas | 7 | 1 issue found |

# Discovered Issues

## Unchecked return values of ERC20 calls

In **Sweeper.sol** contract in `transferERC20` function, the returned value after interactions with token contracts is not checked.

According to the ERC20 standard:

```
Callers MUST handle false from returns (bool success). Callers MUST
NOT assume that false is never returned.
```

Consider using SafeTransfer function to transfer ERC20 tokens.

> *Comment from the developers: The **Sweeper.sol** contract was previously used during migration and is retired now.*

## Code quality

There are several contracts where functions can be declared as `external` instead of `public`:

- `delegate` function in **VestLock.sol** contract;
- `lock` function in **OnlyAddress.sol** contract;
- `multisend` function in **Multisend.sol** contract.

It helps to improve code readability and optimize gas consumption in the project.

## Missing events

There are several contracts where setter functions do not `emit` an `event`:

- `setVerificationKey` function in **VkeySetter.sol** contract;
- `setIntervalBP` function in **GovernorRewards.sol** contract;
- `setExecutorL2` function in **Sender.sol** contract.

Emitting of events in setter functions allows to notify the contract owner and relevant parties about important state changes within the contract.

This analysis was performed by Pessimistic:

Egor Dergunov, Junior Security Engineer
Evgeny Bokarev, Junior Security Engineer
Daria Korepanova, Senior Security Engineer
Pavel Kondratenkov, Senior Security Engineer
Nikita Kirillov, Product Owner
Evgeny Marchenko, CTO
Konstantin Zherebtsov, Business Development Lead

May 28, 2024