OpenZeppelin | security

Scroll Diff Audit Report

September 21, 2023

Table of Contents

Summary

Type	Layer 2
Timeline	From 2023-09-13 To 2023-09-15
Languages	Solidity

Total Issues	3 (3 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	1 (1 resolved)

Scope

We performed a diff audit of the `scroll-tech/scroll` repository for pull request 887 at commit `02bce20`, pull request 912 at commit `10743c2`, pull request 893 at commit `f8b9da0`, and pull request 943 at commit `3d9bfb5`.

In scope were the following contracts:

```
contracts
└── src
    ├── L1/rollup
    │   ├── IL1MessageQueue.sol
    │   ├── L1MessageQueue.sol
    │   └── ScrollChain.sol
    ├── L2/predeploys
    │   └── L2TxFeeVaults.sol
    └── libraries
        └── FeeVault.sol
```

System Overview

Scroll is an EVM-equivalent ZK-rollup designed to be a scaling solution for Ethereum. It achieves this by interpreting EVM bytecode directly at the bytecode level, following a similar path to projects like Polygon's zkEVM and Consensys' Linea.

This audit reviewed the addition of four different pull requests to the Scroll ZK-rollup protocol.

This report presents our findings and recommendations for the new additions to the Scroll ZK-rollup protocol. We urge the Scroll team to consider these findings in their ongoing efforts to provide a secure and efficient Layer 2 solution for Ethereum.

Low Severity

L-01 Insufficient Tests When Using BitMaps

Pull request 893 changes the way of popping and tracking skipped messages, using BitMaps and buckets instead. This represents a sensitive change compared to the previous version. However, only a single test case with a fixed random BitMap, count, and starting index was introduced.

In order to verify that the expected behavior of filling the buckets and skipping messages is the expected one from the rest of the code, consider adding more test cases, especially testing edge cases for filling buckets.

Update: *Resolved in pull request 956 at commit `6749eb7`.*

L-02 Implicit Limitation of Withdrawal

On pull request 912, the `FeeVault` contract introduced the possibility to pass a parameter of the value to withdraw. However, this value is implicitly limited to the contract's current balance by the `sendMessage` function call.

In order to fail early and return a clear error message (which would help when debugging a reverted transaction), consider adding a requirement that asserts that the value passed is equal to or less than the contract's balance.

Update: *Resolved in pull request 954 at commit `08b8bc9`.*

Notes & Additional Information

N-01 Misleading Documentation

Throughout the codebase, there are some instances of incorrect or misleading documentation. In particular:

- Pull request 943 refactored the functionality of the `FeeVault` contract under the `L2TxFeeVault` contract. However, the NatSpec was copied from the `FeeVault` contract, stating: "The L2TxFeeVault contract contains the basic logic for the various different vault contracts used to hold fee revenue generated by the L2 system". Unless the `L2TxFeeVault` contract is going to be used as a base contract for other future contracts, consider adjusting the documentation to reflect the current behavior.
- The comment in line 67 of `L1MessageQueue.sol` should say "for dropped messages" instead of "skipped messages".

Consider resolving these instances of incorrect documentation to improve the clarity and readability of the codebase.

Update: *Resolved in pull request 955 at commit `e1a3f95`.*

Conclusion

Throughout this 3-day audit, we reviewed the mentioned pull requests and identified two low-severity issues, as well as one note to improve the documentation of the codebase.