OpenZeppelin | security

Scroll - Bridge Gas Optimizations Audit

February 6, 2024

Table of Contents

Summary

Type	Layer 2
Timeline	From 2024-01-08 To 2024-01-19
Languages	Solidity

Total Issues	11 (4 resolved, 2 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	1 (1 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (1 resolved)
Notes & Additional Information	8 (2 resolved, 2 partially resolved)

Scope

We audited pull request #1011 from the scroll-tech/scroll repository at commit 45e1305.

In scope were the following files:

```
contracts/src
├── L1
│   ├── gateways
│   │   ├── L1CustomERC20Gateway.sol
│   │   ├── L1ERC1155Gateway.sol
│   │   ├── L1ERC721Gateway.sol
│   │   ├── L1ETHGateway.sol
│   │   ├── L1GatewayRouter.sol
│   │   ├── L1StandardERC20Gateway.sol
│   │   ├── L1WETHGateway.sol
│   │   └── usdc
│   │       └── L1USDCGateway.sol
│   ├── L1ScrollMessenger.sol
│   └── rollup
│       ├── IL1MessageQueue.sol
│       ├── IL1MessageQueueWithGasPriceOracle.sol
│       ├── IL2GasPriceOracle.sol
│       ├── IScrollChain.sol
│       ├── L1MessageQueue.sol
│       ├── L1MessageQueueWithGasPriceOracle.sol
│       ├── L2GasPriceOracle.sol
│       └── ScrollChain.sol
├── L2
│   ├── gateways
│   │   ├── L2CustomERC20Gateway.sol
│   │   ├── L2ERC1155Gateway.sol
│   │   ├── L2ERC721Gateway.sol
│   │   ├── L2ETHGateway.sol
│   │   ├── L2GatewayRouter.sol
│   │   ├── L2StandardERC20Gateway.sol
│   │   ├── L2WETHGateway.sol
│   │   └── usdc
│   │       └── L2USDCGateway.sol
│   └── L2ScrollMessenger.sol
├── libraries
│   ├── gateway
│   │   ├── IScrollGateway.sol
│   │   └── ScrollGatewayBase.sol
│   ├── IScrollMessenger.sol
│   └── ScrollMessengerBase.sol
└── misc
    └──EmptyContract.sol
```

Dependencies, tests, scripts, and changes outside the pull request were left out of the scope.

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 path similar to that taken by projects like Polygon's zkEVM and Consensys' Linea.

This audit reviewed the changes made to multiple Scroll contracts as part of pull request #1011. These changes had the single purpose of reducing the gas cost of the operation. The most notable changes consist of:

- Adapting parameters stored in variables as immutable values and assigning those values during the deployment
- Moving validations from the initialization stage to the implementation deployment stage
- Creation of a new contract that inherits from the `MessageQueue` contract and adds functionalities from the `L2GasPriceOracle` contract to reduce the dependency on external calls
- Simplified message gas limit calculation by setting a single multiplying factor
- Initial deprecation of the ETH gateways in favor of connecting the routers to the messengers directly
- Moving forward with the usage of custom errors instead of `require` statements

This report presents our findings and recommendations regarding the additions made 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.

High Severity

H-01 ETH Deposits Can Get Stuck if They Are Not Successfully Bridged

Pull request #1011 introduced the change of redirecting the calls to deposit ETH from the `L1GatewayRouter` contract to the `L1ScrollMessenger` contract without going through the `L1ETHGateway` contract. This was done with the intention of reducing the gas cost associated with such an action.

However, this causes a problem. Namely, in situations in which a message could not be correctly sent through the bridge, the dropping and asset-return mechanism implemented in the `L1ScrollMessenger` contract will get stuck and the assets will not be able to be paid back. This is due to the lack of the `onDropMessage` hook implementation in the `L1GatewayRouter` contract, which serves as a handler to repay the respective origin of the message.

For instance, if a user wants to simply deposit ETH, in both versions they should call the `depositETH` function from the `L1GatewayRouter` contract. The difference lies in who calls and passes the message to the `L1ScrollMessenger` contract. In the former implementation, the message would come from the `L1ETHGateway` contract, whereas in the current implementation, the `L1GatewayRouter` will be the caller. This is relevant as the `_msgSender` call will then be part of the `_xDomainCalldata` data that will be used to keep track of the message (with its hash) but will also be used in case the message needs to be dropped.

In such a dropping scenario, as the address that sent the message to the `L1ScrollMessenger` contract is the one that will be called to execute the `onDropMessage` hook, if such hook is not implemented, the dropping mechanism will fail and the original user will not get their ETH back. This is the same as how it used to happen when routing the call through the `L1ETHGateway` contract. As this does not depend on the data added to the `depositETH` call, those funds will get stuck in case they are not bridged successfully.

To showcase this, one can get inspired by the following gist; however, caution should be made when fixing this issue, since the gist's proposed scenario in which the issue resolves is merely an example and it is not meant to represent a fully valid resolution.

Consider implementing the `onDropMessage` hook in the `L1GatewayRouter` contract to handle the back payment when dropping messages.

Update: *Resolved in pull request #1093 at commit `888c3d2`. The respective contracts have been rolled back to a previous state in which the bypass previously done over the respective ETH gateways is no longer there, and in which the gateway routers have to go through the ETH gateways when depositing/withdrawing ETH.*

Low Severity

L-01 Implementation Keeps Functionalities for Deprecated Variables

At line 82 of `L1GatewayRouter.sol`, it is explained that the `ethGateway` parameter is no longer in use. However, the logic that makes use of/changes this variable, such as the check in the initialize function and the `setETHGateway` function, is maintained.

If a variable is no longer in use, consider removing the logic that uses it.

Update: *Resolved in pull request #1094 at commit `223538d`.*

L-02 Solidity Version Is Not Fixed and Its Use Is Inconsistent

In the codebase, there are some contracts whose pragma statement does not use a fixed version, whereas others are correctly using a fixed one.

Consider reviewing all the contracts and always using the same fixed Solidity pragma version in all of them. This will help improve consistency and avoid compiling contracts with unexpected compiler versions.

Update: *Acknowledged, not resolved. The Scroll team stated:*

> *We prefer to leave base contracts and interfaces using `^0.8.0` such that the third party can inherit it more easily.*

Notes & Additional Information

N-01 Different Frameworks Are Used Concurrently in the Protocol

When it comes to checking the correct execution of unit tests, coverage, and scripts, the project offers integrations with both Hardhat and Foundry. However, we identified some issues that are worth analyzing:

- The coverage does not work if run through Hardhat. This is because the instrumentation step fails with the following error:

```
Error in plugin solidity-coverage: Error: Could not instrument: L2/predeploys/L1BlockContainer.sol. (Please verify solc can compile this file without errors.) extraneous input ',' expecting {'from', '{', '}', '(', 'error', 'for', 'function', 'address', 'calldata', 'if', 'assembly', 'return', 'revert', 'byte', 'let', '=:', 'switch', 'callback', DecimalNumber, HexNumber, HexLiteralFragment, 'break', 'continue', 'leave', 'payable', 'constructor', 'receive', Identifier, StringLiteralFragment} (251:23)
```

- Even though the coverage with Foundry executes well, it shows an empty coverage for all the contracts inside the `src/libraries/verifier` sub-directory.
- There is no script defined in the `package.json` to run the Foundry coverage. Consider adding one as done for the tests and for Hardhat's coverage.

Scripts exist for both Foundry and Hardhat, but it seems that those used in the latter are outdated and deprecated. This is error-prone and a concern since production environment variables can be used in the wrong scripts and thereby run unneeded/erroneous transactions. As such, consider whether is worth maintaining both frameworks, unifying the testing and coverage. In addition, consider having a unique way of running production scripts to avoid unexpected executions.

Update: *Acknowledged, will resolve. In pull request #1095 at commit `26fa7a1` a specific script has been defined to run coverage with Foundry. Verifiers contracts are being skipped by the coverage in the `.solcover.js` file. The Scroll team stated:*

> *We also noted this issue. That's why we added it to `skipFiles` in `.solcover.js`.*

N-02 Renaming Opportunities

The INTRINSIC_GAS_NONZERO_BYTE factor that multiplies the length of the message-to-be-sent in order to calculate the amount of gas needed for the L2 execution, has a name that originates from a previous version that differentiated between zero and non-zero bytes. However, now, there is no such distinction and its name suggests that the whole message does not have a zero byte.

Consider changing its name to something more appropriate that does not refer to old code's behavior.

Update: *Resolved in pull request #1096 at commit `71d8c78`.*

N-03 Mismatch Between Interface and Implementation

Throughout the codebase, there are some instances in which the interface differs from the actual implementation:

- The parameter `_calldata` from the `L1MessageQueue.calculateIntrinsicGasFee` function is defined as `memory` in the interface but as `calldata` in the implementation.
- The `L2GasPriceOracle.intrinsicParams` getter from the implementation is not reflected in the interface.
- The `IL1MessageQueueWithGasPriceOracle` interface does not reflect the existence of the `l2BaseFee` and `whitelistChecker` getters from the implementation.
- The `gasOracle` public variable of the `L1MessageQueue` contract is not defined in the corresponding interface. The same happens for all the immutable and public variables except for `pendingQueueIndex` which has a specific getter defined in the interface.
- The parameter `_calldata` from the `IL1GatewayRouter.setERC20Gateway` function is defined as `memory` in the interface but as `calldata` in the implementation.

Consider reviewing the entire codebase and making all the interfaces consistent with their implementations.

Update: *Partially resolved in pull request #1097 at commit `747f354`. Only the `memory` input in the `IL1MessageQueue` interface and the inconsistency between the `IL1MessageQueueWithGasPriceOracle` interface and its implementation has been resolved. However, in the latter ones, the variables have been marked as `override`. The Scroll team stated:*

> *Two are fixed. The others are intended to not be public in the interface or will be fixed at a later time.*

N-04 Inconsistent Use of the `__gap` Variable

Throughout the codebase, there are contracts that make use of the `__gap` variable whereas others do not. As the majority of the contracts are upgradeable, consider consistently defining a `__gap` variable for each one of such upgradeable contracts, with the corresponding size being according to the defined storage slots. Furthermore, consider adding comments mentioning the slots that were already used to keep track of the deprecated slots when upgrading the contracts.

Moreover, the `L2ScrollMessenger` contract uses a variable called `__used` to reflect the slots that were used prior to changing them into immutable parameters or into parameters are no longer in use. However, the rest of the codebase has adopted the approach of replacing those slots with deprecated private variables.

In order to be consistent and prevent possible mistakes when upgrading future versions of the contract, consider keeping the same style of deprecating previously used slots while also addressing the lack of the `__gap` variable in most of the contracts.

Update: *Acknowledged, not resolved. The Scroll team stated:*

> *We are comfortable with the current implementation, no change is needed.*

N-05 Code Style Inconsistencies

Throughout the codebase, there are places at which the code style adopted is not consistent across all the contracts:

- In some instances, the `onlyInitializing` modifier is used, but in others it is not.
- A few events are defined in the interface, whereas in other cases they are defined in the implementation contract.

- The `L2GasPriceOracle.IntrinsicParams` struct should be defined in the interface instead of in the implementation to be consistent with the rest of the codebase.
- The `ErrorZeroAddress` error defined in `L1GatewayRouter` should be defined in the `IL1GatewayRouter` interface as well to be consistent with the other implementations like the `IScrollChain` interface. The same applies to the `L2GatewayRouter` contract.

Consider fixing such inconsistencies to improve the overall readability and clarity of the codebase.

Update: *Acknowledged, not resolved. The Scroll team stated:*

> *We are comfortable with the current implementation, no change is needed.*

N-06 Potential Gas Improvements

Throughout the codebase, there are some instances in which the code can be refactored to be more gas-efficient:

- Many getters are duplicated due to there being `public` variable declarations along with specific getter definitions as well. Consider using only one of the two and checking the code for other such instances.
- Some functions might be defined as `external` instead of `public`. Consider reviewing the entire codebase for other similar occurrences like the one in the `L2ETHGateway` contract.
- The use of `require` statements instead of custom errors has been proven to consume more gas. Even though there are attempts at porting the existing `require` statements to custom errors, several cases still remain that have not been so ported. Consider porting these across the entire codebase.

Consider whether it is worth refactoring the code to accommodate such changes so that less gas is consumed.

Update: *Acknowledged, not resolved. The Scroll team stated:*

> *We are comfortable with the current implementation, no change is needed.*

N-07 Missing or Inconsistent Documentation

Throughout the codebase, there are inconsistencies in the documentation. Particularly when checking the NatSpec docstrings from the other analogous set of contracts:

- At lines 47 and 57 of the `L1ERC721Gateway` contract, "in L1" is missing at the end. The same happens at lines 47 and 54 of the `L1ERC1155Gateway` contract, at lines 47, 48, 63 and 64 of the `L1CustomERC20Gateway` contract, and at lines 57, 58, 81 and 82 of the `L1USDCGateway` contract.
- The `L1WETHGateway` contract misses the same statement present in the `L2WETHGateway` contract about parameters not being used.
- The `L2ETHGateway` contract is missing documentation in the `_withdraw` function analogous to the one in `L1ETHGateway._deposit` but with the respective parameters.
- The `L2GatewayRouter` is missing the "@dev This variable is no longer used" comment in the `ethGateway` variable definition.
- In the `IL2GasPriceOracle` interface, the two added getter functions do not have any documentation besides a single "@notice" comment.

Consider fixing the reported examples to improve the overall readability of the codebase.

Update: *Partially resolved in pull request #1098 at commit `a8addd8`. The comment on the `L2GatewayRouter` contract has not been added to the variable definition.*

N-08 Deprecated Variables Are Still Being Assigned Values

In the `L1StandardERC20Gateway` contract, the `__l2TokenImplementation` and the `__l2TokenFactory` variables have been deprecated as suggested by the docstrings. However, they are still being assigned values. This not only consumes more gas but is also inconsistent with other places across the codebase, such as the `ScrollGatewayBase` contract where variables are being directly omitted. The same happens in the `L2StandardERC20Gateway` contract with the `__tokenFactory` variable.

Consider removing such assignments to save gas. In addition, consider improving the consistency of the codebase by not assigning values to deprecated variables.

Update: *Resolved in pull request #1099 at commit `fbb7862`.*

Conclusion

Systematic changes have been made across the codebase to reduce the gas consumption of the protocol. The audit yielded one high-severity issue while recommendations to improve the overall quality and health of the codebase have also been made.

The codebase is well-written and has proper documentation. However, it could benefit from a more descriptive reasoning about some of the decisions taken to reduce the gas costs. Furthermore, there are opportunities to improve the deployment and initialization scripts, and to increase the test coverage.

The Scroll team was very responsive throughout the audit period and provided us with information regarding the various aspects of the changes introduced.