OpenZeppelin | security

ZKChain Release Candidate Audit

ZKsync

February 10, 2025

Table of Contents

Summary

Type	Layer 2
Timeline	From 2025-01-13 To 2025-02-05
Languages	Solidity

Total Issues	13 (0 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	4 (0 resolved)
Notes & Additional Information	9 (0 resolved)

Scope

We conducted a diff audit of the matter-labs/era-contracts repository, comparing the HEAD commit 91631aa against the BASE commit 79215dc.

In scope were the following files:

```
da-contracts/contracts/
    - CalldataDA.sol
    - DAContractsErrors.sol
    - RollupL1DAValidator.sol
l1-contracts/contracts/
    bridge/
        - BridgeHelper.sol
        - L1ERC20Bridge.sol
        - L1Nullifier.sol
        - L2WrappedBaseToken.sol
        - L2WrappedBaseTokenStore.sol
        asset-router/
            - AssetRouterBase.sol
            - IAssetRouterBase.sol
            - IL1AssetRouter.sol
            - IL2AssetRouter.sol
            - L1AssetRouter.sol
            - L2AssetRouter.sol
        interfaces/
            - AssetHandlerModifiers.sol
            - IAssetHandler.sol
            - IL1SharedBridgeLegacy.sol
            - IL2SharedBridgeLegacy.sol
        ntv/
            - IL1NativeTokenVault.sol
            - INativeTokenVault.sol
            - L1NativeTokenVault.sol
            - L2NativeTokenVault.sol
            - NativeTokenVault.sol
    bridgehub/
        - Bridgehub.sol
        - CTMDeploymentTracker.sol
        - IBridgehub.sol
        - L1BridgehubErrors.sol
        - MessageRoot.sol
    common/
        - L1ContractErrors.sol
        - L2ContractAddresses.sol
        libraries/
            - DataEncoding.sol
            - Merkle.sol
            - UnsafeBytes.sol
            - L2ContractHelper.sol
    governance/
        - IRestriction.sol
        - L2AdminFactory.sol
        - L2ProxyAdminDeployer.sol
        - PermanentRestriction.sol
        - TransitionaryOwner.sol
    upgrades/
        - BytecodesSupplier.sol
        - GatewayUpgrade.sol
        - IGatewayUpgrade.sol
        - IL1GenesisUpgrade.sol
        - L1GatewayBase.sol
        - L1GenesisUpgrade.sol
        - ZkSyncUpgradeErrors.sol
    state-transition/
        - ChainTypeManager.sol
        - IChainTypeManager.sol
        - L1StateTransitionErrors.sol
        chain-deps/
            - DiamondInit.sol
            - ZKChainStorage.sol
            facets/
                - Admin.sol
                - Executor.sol
                - Getters.sol
                - Mailbox.sol
                - ZKChainBase.sol
        chain-interfaces/
            - IAdmin.sol
            - IGetters.sol
        data-availability/
            - CalldataDA.sol
            - CalldataDAGateway.sol
        l2-deps/
            - IL2GenesisUpgrade.sol
        libraries/
            - BatchDecoder.sol
            - PriorityTree.sol
    transactionFilterer/
        - GatewayTransactionFilterer.sol
system-contracts/
    bootloader/
        - bootloader.yul
    contracts/
        - L2GatewayUpgradeHelper.sol
        - L2GenesisUpgrade.sol
        - MsgValueSimulator.sol
        - SystemContext.sol
        - SystemContractErrors.sol
        - L2GatewayUpgrade.sol
        interfaces/
            - IL1Messenger.sol
            - IL2GenesisUpgrade.sol
        libraries/
            - SystemContractHelper.sol
```

Update:

Due to Low and Note severity issues identified during this audit, the Matter Labs team decided to acknowledge all issues in this report and implement recommended changes as part of the next release.

System Overview

The introduced changes primarily include fixes, refactoring, and refinements based on suggestions and findings from previous audits. Notable changes were made to the following areas of the codebase:

Priority Operations

When a chain switches from `priorityQueue` to `priorityTree`, a skipping mechanism is implemented to allow the last batch to be executed entirely as a priority queue and automatically account for the unprocessed index on the priority tree before switching over. When a chain is migrated from L1 to a settlement layer, the settlement layer priority queue will be automatically deactivated to ensure that all priority operations will be on the tree.

Bridging Previously Unregistered Assets

When bridging previously unregistered assets, an automatic registration flow is set at the `AssetRouter` to register the token with the chain's default `AssetHandler`(`NativeTokenVault`) instead of reverting. This enhances the user experience when it comes to bridging previously unregistered assets. Particular care is taken for the L2 `NativeTokenVault` to account for previously bridged assets via legacy shared bridges. To facilitate this, the `bridgeBurnData` is updated with an extra `tokenAddress` for both the registration and validation of the `assetId`.

Proof Validation

The proof metadata uses an extra byte to indicate if it is the `finalProofNode`. This is to distinguish from the case where only one batch has been executed and, thus, an empty proof should be allowed. In order to accommodate this case, the empty proof check has also been removed from the Merkle library.

`BaseToken` and WETH

The L1 and L2 `NativeTokenVault` will not accept any base token value during a `bridgeMint` call despite it being a `payable` function. This restricts any attached `l2value` to zero when bridging via BridgeHub's `requestL2TransactionTwoBridges` function. `msg.value` is further restricted at places where it is not expected to be non-zero, such as `_bridgeBurnBridgedToken` and `bridgeRecoverFailedTransfer`.

In addition, these restrictions make it impossible for a ZKChain to keep the default L1 `NativeTokenVault` while using a custom L2 `AssetHandler` which accepts non-zero `msg.value` in `bridgeMint` and other functions. WETH is only allowed to be registered on L1 `NativeTokenVault` so that it can receive previously bridged-out WETH.

Legacy Handling

The special, backward-compatible handling of `amount` in the `_bridgeBurnNativeToken` function of `NativeTokenVault` has been updated to ensure consistent transaction data hash for successful recovery after failed transfers. Further restrictions have also been placed on the interfaces of legacy functions to only allow access for L1-originated tokens via legacy bridges to Era chain.

Security Model and Trust Assumptions

No notable changes have been made to privileged roles. However, the diamond proxy admin functions have been further refined and are restricted to L1 only. For detailed trust assumptions on privileged roles, we refer to our previous audit reports.

Low Severity

L-01 Empty Proof Can Pass Validation on `proveL2LeafInclusion`

Due to the removal of the `pathLength` check, empty proofs can pass validation from the external Mailbox.proveL2LeafInclusion call with `_leaf` as the locally stored root for that `_batchNumber` and `_proof` being empty.

During withdrawals from `L1Nullifier`, this will not pose a problem as one cannot specify the `_leaf` directly since it is computed via the _proveL2LogInclusion function. However, `proveL2LeafInclusion` is an `external` function and such behavior may affect unaware integrators who may want to validate proofs using this function.

Consider adding extra documentation to warn about such behavior and emphasizing on additional validation on the `_leaf` hash or the `_proof` length.

L-02 `registerLegacyChain` on `BridgeHub` Should Be `OnlyL1`

The `registerLegacyChain` function of the `BridgeHub` contract is intended to be called only on L1. Consider restricting access to `registerLegacyChain` to `onlyL1` for improved clarity of intention.

L-03 Misleading Documentation

Throughout the codebase, multiple instances of misleading documentation were identified:

- The comment for `_getAbiParams` in `MsgValueSimulator.sol` switches the definitions of the second and third ABI parameters that are returned from the function. The second returned parameter is the one that defines whether the `systemCall` flag should be used, and the third one should be the address to call.
- The comment in the `hashFactoryDeps` function of `L2ContractHelper.sol` states that the resulting hashes are stored in the array sequentially "in bytes", whereas it should more accurately say "in words".

- The comment above the `_approveFundsToAssetRouter` function of `L1ERC20Bridge.sol` states that funds are transferred to `native token vault` whereas, in fact, they are transferred to `L1ERC20Bridge`.
- The comment in the `finalizeWithdrawal` function of `L1ERC20Bridge.sol` refers to "shared bridge" whereas it should refer to `L1Nullifier` instead.
- The comment in `Mekle.sol` noting that the `proofLength` is the height of the tree is misleading since this function is also used with proofs of length that are equal to the height of the tree plus 1. This is because a chain's batch root also includes the aggregated root.
- Since the Merkle proof length might be the height of the tree plus one (N + 1), `index` is no longer simply the leaf index in the tree in all cases. In particular, for the recursive L3 -> L1 proof to work, the index for the `settlementLayer` leaf is of the `2^N, ... , 2^(N+1)-1` range to ensure the correct order of hashing the final two elements. Consider documenting this special case for clarity.
- In `DataEncoding.sol`, the comment of the `encodeTxDataHash` function suggests that `_transferData` only includes the deposit amount and the address of the L2 receiver, whereas it also includes the `maybeToken` address.
- The comment for the `bridgeMint` function of the `NativeTokenVault` contract refers to the legacy Shared Bridge.
- The comment for the `ChainTypeManager` contract refers to the legacy "State Transition Manager" naming.

Consider correcting any instances of misleading documentation to improve the clarity and maintainability of the codebase.

L-04 Wrong Error Parameters Order

In the `L1ContractsErrors` contract, the `AssetIdMismatch` custom error is defined with its first parameter being the expected `assetId` value and the second parameter being the actually provided one. However, the `_assetIdCheck` function of the `NativeTokenVault` contract throws the `AssetIdMismatch` error due to the argument being provided in the opposite order.

To avoid misleading errors from being thrown, consider fixing the order of the error parameters in the `_assetIdCheck` function of `NativeTokenVault`.

Notes & Additional Information

N-01 Redundant Cast

In the `_setAssetHandlerAddressThisChain` function of `L1AssetRouterBase.sol`, the `_nativeTokenVault` argument is unnecessarily cast to the `address` type.

Consider removing the redundant cast for improved code clarity.

N-02 Undocumented Constant

The `CREATE_PREFIX` constant value in the `L2ContractHelper` library is a random `bytes32` value.

Consider documenting how this value was generated.

N-03 Typographical Errors

Throughout the codebase, multiple instances of typographical errors were identified:

- `IL2GenesisUpgrade`, line 17: "THe" should be "The".
- `L2AdminFactory.sol`, line 63: "validateRestrctions" should be "validateRestrictions".
- `L2NativeTokenVault.sol`, line 235: there is an extra dot at the end of the sentence.
- `TransitionaryOwner.sol`, line 11: "a" should be "as".
- `IGetters.sol`, line 67: "transaction" should be "transactions".
- `PriorityTree.sol`, line 74: "is ensures" should be "is ensured".

Consider fixing all typographical errors to improve the clarity and readability of the codebase.

N-04 Code Duplication

In the `setNativeTokenVault` function of the `L1AssetRouter` contract, the `assetId` of ETH is calculated. However, the `ETH_TOKEN_ASSET_ID` constant can be used instead to avoid code duplication.

Consider using the `ETH_TOKEN_ASSET_ID` constant instead of recalculating the `assetId` for ETH.

N-05 Inconsistent Code Style

Throughout the codebase, multiple instances of inconsistent coding style were identified:

- In `L2ContractAddresses.sol`, the `SYSTEM_CONTRACTS_OFFSET` constant is used to define the address of some system contracts. However, a number of other system contracts' addresses are defined without using the offset value. Consider using the offset constant for all system contracts' addresses definitions for consistency.
- In `L2ContractAddresses.sol` and `L2ContractHelper.sol`, the `IL2Messenger` interface and the `L2_MESSENGER` constant refer to the `L1Messenger` system contract. Consider renaming to `IL1Messenger` and `L1_MESSENGER` for consistency and clarity.

Consider maintaining a consistent code style throughout the codebase for improved code clarity and readability.

N-06 Unused Return Parameter

In `NativeTokenVault.sol`, the `ensureTokenIsRegistered` function calls `_registerToken` if the token is not already registered. However, the `newAssetId` return variable of the `internal` function is ignored. In addition, in most of the cases where `ensureTokenIsRegistered` is called, the token's `assetId` is either calculated or retrieved from `NativeTokenVault` right afterwards.

Consider making `ensureTokenIsRegistered` return the token's `assetId` by using the return parameter of `_registerToken` when the token is not already registered.

N-07 Duplicate Import

Duplicate imports can negatively impact code clarity. In `BridgeHub.sol`, the `CTMNotRegistered` custom error is imported twice.

Consider removing the duplicate import to improve code clarity and maintainability.

N-08 Unused Custom Errors

Unused code can negatively impact code clarity. Throughout the codebase, multiple instances of unused custom errors were identified:

`L1BridgehubErrors.sol`:

- `AssetIdAlreadyRegistered`: the same error is defined in `L1ContractsErrors.sol`, which is the only one used within the codebase.
- `ChainIdNotRegistered`: a similar error is defined in `L1ContractsErrors.sol`, which is the only one used within the codebase.

`L1ContractsErrors.sol`:

- `InvalidPubDataHash`: the same error is defined in `L1StateTransitionErrors` and `DAContractsErrors`, which are the only ones used within the codebase.
- `InvalidTxType`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `L2UpgradeNonceNotEqualToNewProtocolVersion`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `NewProtocolMajorVersionNotZero`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PatchCantSetUpgradeTxn`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PatchUpgradeCantSetBootloader`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PatchUpgradeCantSetDefaultAccount`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PreviousProtocolMajorVersionNotZero`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PreviousUpgradeNotCleaned`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PreviousUpgradeNotFinalized`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `ProtocolVersionMinorDeltaTooBig`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `ProtocolVersionTooSmall`: the same error is defined in `ZkSyncUpgradeErrors`, which is the only one used within the codebase.
- `PubdataCommitmentsEmpty`: the same error is defined in `DAContractsErrors`, which is the only one used within the codebase.

- NotEnoughGas: never used within the codebase.

L2ContractErrors.sol:

- WithdrawFailed: the same error is defined in L1ContractsErrors, which is the only one used within the codebase.

SystemContractsErrors.sol:

- HashMismatch: the same error is defined in L1ContractsErrors, which is the only one used within the codebase.
- NonIncreasingTimestamp: the same error is defined in L1ContractsErrors, which is the only one used within the codebase.
- NotEnoughGas: never used within the codebase.
- TooMuchGas: the same error is defined in L1ContractsErrors, which is the only one used within the codebase.

ZkSyncUpgradeErrors.sol:

- ZeroAddress: never used within the codebase.

Consider removing or using any currently unused custom errors to improve the clarity and maintainability of the codebase.

N-09 Naming Suggestions

Throughout the codebase, multiple opportunities for improved contract and custom error naming were identified:

- The AddressAlreadyUsed and AddressAlreadySet errors defined in L1ContractErrors.sol are similar and can be merged together.
- The L2GatewayUpgradeHelper library is used as a helper contract during the genesis upgrade for both Gateway and ZKChains. Consider renaming it to L2GenesisUpgradeHelper for improved clarity.
- The name of the L1GatewayBase abstract contract implies that it serves as a base contract for Gateway. However, it is inherited by GatewayUpgrade and L1GenesisUpgrade contracts, providing functionality to retrieve some necessary data during the genesis upgrade process. Consider renaming L1GatewayBase to a name that more accurately reflects its role.
- The TokenNotLegacy and TokenIsNotLegacy errors defined in L1ContractErrors.sol are similar and can be merged.

Consider renaming any contracts or custom errors whose current name does not clearly indicate their functionality or usage.

Conclusion

This diff audit mostly covers the fixes and code improvements that were made in response to the findings and suggestions of previous audits. Several low- and note-severity issues have been reported to improve the clarity of the codebase and facilitate future audits, integrations, and development. The Matter Labs team has been responsive throughout the engagement, providing useful explanations and insights whenever needed.