OpenZeppelin | security

ZKChain: Upgrades and Libraries Diff Audit

October 28, 2024

Table of Contents

Summary

Type	Layer 2
Timeline	From 2024-10-14 To 2024-10-23
Languages	Solidity

Total Issues	13 (12 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	1 (1 resolved)
Low Severity Issues	5 (4 resolved)
Notes & Additional Information	7 (7 resolved)
Client Reported Issues	0 (0 resolved)

Scope

We audited pull request #793 of the matter-labs/era-contracts repository at commit 8208402.

From the list below, the files that were newly added were audited fully while the rest was only audited as a diff against commit 9615d90:

```
l1-contracts/contracts
├── common
|   ├── Config.sol
|   ├── L2ContractAddresses.sol
|   ├── L1ContractErrors.sol
|   ├── interfaces
|   |   └── IL2ContractDeployer.sol
|   ├── libraries
|   |   ├── DataEncoding.sol
|   |   ├── DynamicIncrementalMerkle.sol
|   |   ├── L2ContractHelper.sol
|   |   ├── Merkle.sol
|   |   ├── SystemContractsCaller.sol
|   |   └── UnsafeBytes.sol
├── upgrades
|   ├── BaseZkSyncUpgrade.sol
|   ├── BaseZkSyncUpgradeGenesis.sol
|   ├── IL1GenesisUpgrade.sol
|   ├── L1GenesisUpgrade.sol
|   ├── IGatewayUpgrade.sol
|   └── GatewayUpgrade.sol
├── vendor
|   └── AddressAliasHelper.sol
└── transactionFilterer
    └── GatewayTransactionFilterer.sol

l2-contracts/contracts
├── L2ContractHelper.sol
└── errors
    └── L2ContractErrors.sol

system-contracts
├── bootloader
|   └── bootloader.yul
└── contracts
    ├── Constants.sol
    ├── L1Messenger.sol
    ├── L2GenesisUpgrade.sol
    ├── PubdataChunkPublisher.sol
    ├── SystemContractErrors.sol
    ├── interfaces
    |   └── IMessageRoot.sol
```

```
    └── libraries
        └── SystemContractHelper.sol
```

System Overview

The changeset under review mainly consists of updates in service of the newly implemented custom bridging framework and chain migration to the Gateway. Due to the newly set bridging mechanism with `L2NativeTokenVault` and `L2AssetRouter`, some L2 system functionalities are included in the `l1-contracts` libraries to facilitate testing as well as L2 force deployment during the genesis upgrade.

A `GatewayUpgrade` contract is newly developed to upgrade the ZKsync Era chain to be part of the ZKChain ecosystem contracts. Furthermore, a newly added `GatewayTransactionFilterer` contract for the L1 Gateway Mailbox imposes restrictions of bridging to the Gateway to only chain migration purposes. We elaborate below on these aspects.

Genesis Upgrade

When creating a new chain via L1's Bridgehub, the `L1GenesisUpgrade` is delegate called from the newly deployed chain's Diamond Proxy to initiate protocol upgrade with an `L2GenesisUpgrade` transaction to force deploy and initiate the corresponding `L2BridgeHub`, `L2AssetRouter`, `L2NativeTokenVault` as well as `L2MessageRoot`.

ZKsync Era Gateway Upgrade

The `GatewayUpgrade` contract is used to migrate ZKsync Era to be part of the ZKChain ecosystem contracts by initializing its `baseTokenAssetId` and the `priorityTree` on L1. Additionally, it facilitates the force deployment of the L2 bridging contracts, such as `L2AssetRouter`, with Era-specific constructor arguments.

Gateway Transaction Filterer

The `GatewayTransactionFilterer` contract is meant to be deployed on L1 and attached to the Gateway's diamond proxy. It filters out all bridging transactions via `L1AssetRouter` that are not for chain migration purposes. This will not block any relayed bridging transactions for chains that settle on the Gateway.

Security Model and Trust Assumptions

Privileged Roles

In relation to the changeset, the following privileged roles can perform critical functionality:

- The owner of the `GatewayTransactionFilterer` contract can add or remove addresses from the set allowed to send transactions to the Gateway chain.
- The `proposedUpgrade` L2 transaction passed to the `GatewayUpgrade.upgrade` function comes from the usual upgrade process and hence, it is constructed off-chain with force deployment data. The governance is trusted to thoroughly verify the `ProposedUpgrade` data before approving it.

We assume that the accounts in charge of the above actions always act in the intended way. Hence, any attacks or vulnerabilities targeting this part of the system were not considered throughout this audit.

Medium Severity

M-01 `GatewayUpgrade` Caller Validation Fails

The `GatewayUpgrade` contract is designed to facilitate the migration of the ZKsync Era chain to be part of the ZKchain ecosystem contracts. The upgrade transaction will revert when validating the caller, as the `msg.sender` can not be the chain's diamond proxy contract. The upgrade goes through the `AdminFacet` upgrade functions and thus the `msg.sender` is either the `chainAdmin` or the `ChainTypeManager` contract.

Since the `GatewayUpgrade` is just the logic contract being delegate called by the diamond proxy, it is not necessary to have access control on this contract. Consider removing the afore-mentioned check, as the upgrade functions of the `AdminFacet` have already implemented the necessary access control restrictions.

Update: *Resolved in pull request #981.*

Low Severity

L-01 Unnecessary `Initializable` Inheritance in `GatewayUpgrade`

The `GatewayUpgrade` contract contains the logic for migrating the ZKsync Era chain to be part of the ZKchain ecosystem. Since `GatewayUpgrade` is only used as a logic contract called through `delegateCall` and requires no initialization, there is no need to inherit the `Initializable` contract.

To reduce gas costs and remove any potential confusions when reading the contract, consider removing the `Initializable` inheritance.

Update: *Resolved in pull request #983.*

L-02 Unnecessary Legacy Check For Chain Migration

The `GatewayTransactionFilterer` contract is designed to filter transactions directed towards the gateway. Specifically, when the transaction sender is identified as `L1_ASSET_ROUTER`, the contract restricts allowed calls exclusively to those with `finalizeDeposit` signatures. Among these, only transactions related to chain migration are permitted.

However, it is not necessary to check the IL2Bridge.finalizeDeposit decoding for two reasons:

1. The encoding associated with this function is different from the legacy interface utilized by the L2AssetRouter. Consequently, the `finalizeDeposit` function, as referenced, does not actually exist within the current framework.

2. It is not possible to encode a chain migration transaction with `L1AssetRouter` legacy calldata encoding. This is because a `ctmAssetId` will invariably have its `tokenAddress` set to zero on `L1NativeTokenVault`, therefore always leading to the encoding inside the `if` block.

Consider filtering out the legacy encoding by removing the check against IL2Bridge.finalizeDeposit.selector.

Update: *Resolved in pull request #1000.*

L-03 The Name `L2Messenger` Is Misleading

There are many L2 system contract addresses defined inside the `L2ContractAddresses.sol` file, each referring to a system contract force deployed on the zkEVM. In particular, on the `0x8008` address there is contract `L1Messenger`, responsible for passing messages and logs from L2 to L1.

It is confusing when the `L2_MESSENGER` is in fact referring to the `L1Messenger`, whose address is defined earlier as `L2_TO_L1_MESSENGER_SYSTEM_CONTRACT_ADDR`. In the event of further L2<>L2 communication, there could be a future `L2Messenger` as distinguished from `L1Messenger` on the system contract.

In addition, the interface `IL2Messenger` is the same as `IL1Messenger`. The instance where `L2_MESSENGER` is used can be replaced by existing `IL1Messenger` instead.

When referring to the `L1Messenger` system contract, consider using the already existing code such as `IL1Messenger(L2_TO_L1_MESSENGER_SYSTEM_CONTRACT_ADDR)`, instead of `L2_MESSENGER` and `IL2Messenger`.

Update: *Acknowledged, will resolve. The Matter Labs team stated:*

> *Acknowledged. We use different names depending on the context (L2 or L1 contracts). We will add this change to one of the next refactoring releases.*

L-04 Missing or Incomplete Docstrings

Throughout the codebase, multiple instances of missing or incomplete docstrings were identified. For instance,

- In `IL1GenesisUpgrade.sol`, the `_zkChain`, `_l2Transaction`, `_protocolVersion` and `_factoryDeps` parameters of the `GenesisUpgrade` event are not documented.
- In `IL2ContractDeployer.sol`, the `_deployParams` parameter of the `forceDeployOnAddresses` function is not documented.
- In `L1GenesisUpgrade.sol`, the `_l1GenesisUpgrade`, `_chainId`, `_protocolVersion`, `_l1CtmDeployerAddress`, `_forceDeploymentsData` and `_factoryDeps` parameters of the `genesisUpgrade` function are not documented.
- In `L1Messenger.sol`, the `_l2DAValidator` parameter of the `publishPubdataAndClearState` function is not documented.
- In `GatewayUpgrade.sol`, the `THIS_ADDRESS` state variable is not documented.
- In `IGatewayUpgrade.sol`, the `IGatewayUpgrade` interface is not documented.
- In `IGatewayUpgrade.sol`, the `upgradeExternal` function is not documented.
- In `IL1GenesisUpgrade.sol`, the `IL1GenesisUpgrade` interface is not documented.
- In `IL1GenesisUpgrade.sol`, the `genesisUpgrade` function is not documented.
- In `IMessageRoot.sol`, the `IMessageRoot` interface is not documented.
- In `IMessageRoot.sol`, the `getAggregatedRoot` function is not documented.
- In `L2ContractHelper.sol`, the `getNewAddressCreate2` function is not documented.
- In `L2ContractHelper.sol`, the `verifyCompressedStateDiffs` function is not documented.
- In `L2GenesisUpgrade.sol`, the `genesisUpgrade` function is not documented.

Additionally, because some upgrade functions do not have a standard invocation flow and are often a chain of `delegatecall`, it would greatly improve the understanding and clarity of code for both auditors and developers if the invocation paths would be indicated directly in the documentation of each function.

Update: *Resolved in pull request #999.*

L-05 Unused Code

Throughout the codebase there are several parts which are unused:

- The `InsufficientAllowance`, `InvalidInput`, `PubdataIsEmpty`, `UnimplementedMessage` and `UnsupportedPaymasterFlow` errors of the `L1ContractErrors.sol` file.
- The `AddressMismatch`, `AssetIdMismatch`, `DeployFailed`, `EmptyBytes32`, `InvalidCaller`, `NonSequentialVersion`, `UnimplementedMessage` errors of the `L2ContractErrors.sol` file.
- The `readUint128` function.
- The `PubdataChunkPublisher` contract does not use functionality inherited from the `SystemContractBase` contract, and hence does not require this inheritance.

To make the code easier to read and save some gas costs at deployment time, consider removing the above mentioned instances of unused code.

Update: *Resolved in pull request #995.*

Notes & Additional Information

N-01 TODO Comments in the Code

Throughout the codebase, multiple instances of TODO/Fixme comments were found. For instance:

- The `TODO` comment in line 21 of `Config.sol`.
- The `todo` comment in line 128 of `DataEncoding.sol`.

Consider removing all instances of TODO/Fixme comments and instead tracking them in the issues backlog. Alternatively, consider linking each inline TODO/Fixme to the corresponding issues backlog entry.

Update: *Resolved in pull request #994. The Matter Labs team stated:*

> *We removed `todo` from DataEncoding, the `todo` in Config.sol is linked to SMA-184.*

N-02 Legacy Naming

An instance of legacy naming is identified below:

- stmAddress should be "ctmAddress".

Consider updating it for consistency and clarity.

Update: *Resolved in pull request #991.*

N-03 Unnecessary Code

There are several instances where code is duplicated and therefore unnecessary:

- The `DEPLOYER_SYSTEM_CONTRACT` address is the same as `L2_FORCE_DEPLOYER_ADDR`, consider removing one of them.
- The `IL2ContractDeployer` file can be deleted as the same interface but with a different name `IContractDeployer` exists within `L2ContractHelper`.

Update: *Resolved in pull request #990.*

N-04 Unsafe ABI Encoding

It is not an uncommon practice to use `abi.encodeWithSignature` or `abi.encodeWithSelector` to generate calldata for a low-level call. However, the first option is not typo-safe and the second option is not type-safe. The result is that both of these methods are error-prone and should be considered unsafe.

The use of `abi.encodeWithSelector` within `GatewayUpgrade.sol` is unsafe.

Consider replacing all the occurrences of unsafe ABI encodings with `abi.encodeCall` which checks whether the supplied values actually match the types expected by the called function and also avoids errors caused by typos.

Update: *Resolved in pull request #989.*

N-05 Unnecessary Cast

Within the `SystemContractsCaller` contract, the `uint32(Utils.safeCastToU32(data.length))` cast is unnecessary. Consider removing it.

Update: *Resolved in pull request #988.*

N-06 Typographical Errors

Consider correcting the following typographical errors in the codebase:

- `asse3t` should be `asset`.
- `L2_BRIDDGE_HUB` should be `L2_BRIDGE_HUB` and all instances of imported use in the L2GenesisUpgrade.sol contract.
- `Progapatate` should be `Propagate`.

Update: *Resolved in pull request #987.*

N-07 Misleading Documentation

Below are two instances of misleading documentation:

- The comment on top of the `decodeTokenData` function mentions that the encoding of `_tokenData` contains the asset deployment tracker, which is not the case as it is encoded from the name, symbol and decimals with or without a chainId. Consider removing it.
- The use of `upgrade proxy` naming causes confusion as the upgrade functions are meant to be invoked via the `Admin` facet of the chain's diamond proxy. Consider removing references to `upgrade proxy` for clarity.

Update: *Resolved in pull request #986.*

Conclusion

The changeset updates some library files and upgrade contracts in service of the newly implemented custom bridging framework and chain migration to the Gateway. We appreciate the Matter Labs team for their kind support during the audit particularly showing us the off-chain upgrade mechanism.