OpenZeppelin | security

FFLONK and EVM Equivalence Diff Audit

March 4, 2025

Table of Contents

Summary

Type	Layer 2
Timeline	From 2025-01-30 To 2025-02-11
Languages	Solidity Yul

Total Issues	6 (5 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	1 (1 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	3 (2 resolved)

Scope

We conducted a diff audit of the matter-labs/era-contracts repository by comparing the HEAD commit 5731979 against the base commits listed in the scope tree for each file. We also conducted a full audit of files marked with the `full` label in the scope tree.

In scope were the following files:

```
.
├── l1-contracts
│   ├── contracts
│   │   ├── common
│   │   │   ├── Config.sol (`fe838fc`)
│   │   │   ├── L1ContractErrors.sol (`d7fabf85`)
│   │   │   └── interfaces
│   │   │       └── IL2ContractDeployer.sol (`fe838fc`)
│   │   ├── governance
│   │   │   ├── ChainAdmin.sol (`21ac083`)
│   │   │   └── IChainAdmin.sol (`21ac083`)
│   │   ├── upgrades
│   │   │   ├── ZkSyncUpgradeErrors.sol (`8217496`)
│   │   │   ├── L1GenesisUpgrade.sol (`8217496`)
│   │   │   └── BaseZkSyncUpgrade.sol (`0778e18`)
│   │   └── state-transition
│   │       ├── chain-deps
│   │       │   ├── DiamondInit.sol (`79215dc`)
│   │       │   ├── GatewayCTMDeployer.sol (`21ac083`)
│   │       │   ├── ZKChainStorage.sol (`79215dc`)
│   │       │   └── facets
│   │       │       ├── Admin.sol (`79215dc`)
│   │       │       ├── Executor.sol (`79215dc`)
│   │       │       ├── Getters.sol (`79215dc`)
│   │       │       ├── Mailbox.sol (`79215dc`)
│   │       │       └── ZKChainBase.sol (`79215dc`)
│   │       ├── chain-interfaces
│   │       │   ├── IAdmin.sol (`79215dc`)
│   │       │   ├── IDiamondInit.sol (`7827bf3`)
│   │       │   ├── IGetters.sol (`0778e18`)
│   │       │   ├── IMailbox.sol (`fe838fc`)
│   │       │   ├── IVerifier.sol (`37719a9`)
│   │       │   └── IVerifierV2.sol (`37719a9`)
│   │       └── verifiers
│   │           ├── DualVerifier.sol (`858da06`)
│   │           ├── VerifierFflonk.sol (`703acc1`)
│   │           └── VerifierPlonk.sol (`49868af`)
├── l2-contracts
│   └── contracts
│       ├── chain-interfaces
```

```
│       │   ├── IVerifier.sol (`37719a9`)
│       │   └── IVerifierV2.sol (`37719a9`)
│       ├── errors
│       │   └── L2ContractErrors.sol (`37719a9`)
│       ├── interfaces
│       │   └── IConsensusRegistry.sol (`00ddc06`)
│       └── verifiers
│           ├── DualVerifier.sol (`37719a9`)
│           ├── VerifierFflonk.sol (`37719a9`)
│           └── VerifierPlonk.sol (`37719a9`)
└── system-contracts
    ├── bootloader
    │   └── bootloader.yul (`0778e18`)
    ├── contracts
    │   ├── AccountCodeStorage.sol (`0778e18`)
    │   ├── BootloaderUtilities.sol (`0778e18`)
    │   ├── Constants.sol (`0778e18`)
    │   ├── ContractDeployer.sol (`0778e18`)
    │   ├── DefaultAccount.sol (`0778e18`)
    │   ├── EvmEmulator.yul (`bbfd8e7`)
    │   ├── EvmEmulator.yul.llvm.options (`0778e18`)
    │   ├── EvmGasManager.yul (`66c8e51`)
    │   ├── EvmPredeploysManager.sol (full)
    │   ├── KnownCodesStorage.sol (`7328b8c`)
    │   ├── L2BaseToken.sol (`0778e18`)
    │   ├── SystemContext.sol (`79215dc`)
    │   ├── SystemContractErrors.sol (`79215dc`)
    │   ├── abstract
    │   │   └── SystemContractBase.sol (`0778e18`)
    │   ├── interfaces
    │   │   ├── IAccountCodeStorage.sol (`213f990`)
    │   │   ├── IBaseToken.sol (`0778e18`)
    │   │   ├── IContractDeployer.sol (`0778e18`)
    │   │   └── IKnownCodesStorage.sol (`0778e18`)
    │   ├── libraries
    │   │   ├── SystemContractHelper.sol (`79215dc`)
    │   │   ├── TransactionHelper.sol (`0778e18`)
    │   │   └── Utils.sol (`816c358`)
    │   └── precompiles
    │       ├── CodeOracle.yul (`875d0a5`)
    │       └── Identity.yul (full)
    └── evm-emulator
        ├── EvmEmulator.template.yul (`bbfd8e79`)
        ├── EvmEmulatorFunctions.template.yul (`a9eb273`)
        ├── EvmEmulatorLoop.template.yul (`16b76a2`)
        ├── EvmEmulatorLoopUnusedOpcodes.template.yul (`df1c554`)
        └── calldata-opcodes
            ├── ConstructorScope.template.yul (`684c072`)
            └── RuntimeScope.template.yul (`16b76a2`)
```

Update: *All resolutions and the final state of the audited codebase mentioned in this report are contained at commit [f1cae64](), including two the new contracts added as part of the fix review* `EvmHashesStorage.sol` *and* `IEvmHashesStorage.sol`*. This commit also covers the* `ChainRegistrar.sol` *contract, which was out of scope for this audit. The Matter Labs team*

noted that this contract is a draft of the technical contract that is planned to be used to collect data in the future for new chains registrations. As such, this contract does not present any security risks.

System Overview

The changes include code adjustments to support the EVM Emulator and Dual Verifier, as well as various fixes, refactoring, and refinements. Key updates were made in the following areas of the codebase:

EVM Equivalence

- **`Identity` Precompile**: An `Identity` precompile has been added to ZKsync to maintain EVM equivalence. Some EVM contracts rely on this precompile to efficiently copy large chunks of memory.
- **`EvmPredeploysManager`**: The `EvmPredeploysManager` was introduced to streamline the deployment of frequently used EVM contracts, providing a better developer experience on ZKsync.
- **Core Contract Updates**: Several core contracts were modified to safely enable EVM emulation within the system, ensuring clear and reliable operations.

Verifiers

Minor changes were made to verifier logic, including updates to verification key hashes and gas optimizations.

Security Model and Trust Assumptions

No notable changes were made to privileged roles. However, the diamond proxy admin can now enable the EVM Emulator—restricted from Layer 1. For detailed information on trust assumptions and privileged roles, please refer to previous audit reports.

Medium Severity

M-01 Bytecode Hash Restoration Failure with `forcedSload` for EVM Contracts

The `forcedSload` function of the `SystemContractHelper` library is intended to let system contracts read the storage slot of any account. This is achieved by forcibly deploying a special `SLOAD` contract at the target address, invoking its `sload` function, and then restoring the previous contract hash with another forced deployment. These deployments are not meant to invoke a constructor, as the `callConstructor` flag is set to `false`.

When `forcedSload` is called for an EVM contract, it does not restore the correct bytecode hash for that account, disrupting the contract's operability. This occurs because during the EVM force deployment logic, the `_constructEVMContract` function initially sets a placeholder bytecode hash indicating an EVM constructing contract, which contradicts the intended behavior of avoiding constructor calls. Since the `_deployment.input` parameter in `forcedSload` is empty, the EVM emulator interprets the constructor logic for an empty constructor code, returning an empty actual EVM bytecode, causing a mismatch in both the versioned bytecode hash and the final EVM bytecode hash. Consequently, the bytecode hash after the forced call differs from the original, breaking the functionality of any deployed EVM contract.

Consider disallowing the `forcedSload` call for EVM emulated contracts or documenting that this function should never be called for them.

Update: *Resolved in pull request #1255. The Matter Labs team stated:*

> *We added a way to perform forced deployments of EVM contracts without constructor calls.*

Low Severity

L-01 Ambiguity in Verification Key Hash Retrieval in `DualVerifier`

There are two functions for retrieving the verification key hash in `DualVerifier`: `verificationKeyHash` and `verificationKeyHash(uint256)`. The first function returns the PLONK verification key hash by default, but the contract also supports FFLONK, which can lead to uncertainty about which verification key is actually being used.

Consider updating the function names or adding explicit references to each verifier type to ensure that the correct verification key is always retrieved.

Update: *Resolved in pull request #1273. The Matter Labs team stated:*

> *This function is used to keep backward compatibility with older Verifier implementations. We added comments to reduce uncertainty.*

L-02 Lack of Validation of `l2EvmEmulatorBytecodeHash` in `DiamondInit`

The `DiamondInit` contract initializes various parameters, including `_initializeData.l2EvmEmulatorBytecodeHash`, which is intended to represent the bytecode hash of an EVM emulator. In the `initialize` function, there is no validation for `_initializeData.l2EvmEmulatorBytecodeHash` before it is stored, which may allow incorrect values to be accepted.

Consider validating the `l2EvmEmulatorBytecodeHash` parameter as well as other parameters passed to the `initialize` function, similar to the validation performed for `baseTokenAssetId'.

Update: *Resolved in pull request #1276. The Matter Labs team stated:*

> *We added validation for hashes.*

Notes & Additional Information

N-01 `TestnetVerifier` Can Be Deployed on ZK Gateway

In the constructor of `TestnetVerifier`, there is a check to prevent deployment on Ethereum mainnet. However, no such restriction exists for ZK Gateway. Since `TestnetVerifier` is used in both L1 and Gateway contracts, this omission may result in the unintended use of a testnet verifier on the Gateway.

Consider adding a check to ensure that `TestnetVerifier` is not deployed on Gateway.

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

> *The most sense would be to add a check here that does not allow it to be deployed on Gateway. This can be done after Gateway is launched.*

N-02 Gas Inefficiency in `_extractProof`

The `_extractProof` function of the `DualVerifier` contract has been designed to retrieve proof elements from the `_proof` calldata array for the subsequent call to a verifier. In its current implementation, `_extractProof` loops over the `_proof` array to copy all elements after the first into the `result` memory array. While straightforward, this approach is gas-inefficient because it incurs costs proportional to the array length. Since `_proof` is stored in calldata, the `calldatacopy` opcode in an assembly block can be used instead, reducing gas usage by roughly 13 times for a 24-element FFLONK proof.

Consider using `calldatacopy` in an assembly block for `_proof` array copying from the first element to the `result` array to reduce gas consumption.

Update: *Resolved in pull request #1274. The Matter Labs team stated:*

> *This optimization is less effective when using verifier on Gateway. The issue was fixed.*

N-03 Inconsistent or Incomplete Documentation Across the Codebase

Throughout the codebase, multiple instances of inconsistent or incomplete documentation were identified:

- The `SystemContext` contract includes a `coinbase` variable that is intended to represent the block proposer's address. Commit b494799 removed the important docstring associated with the `coinbase` variable. This removal may lead to confusion about the variable's usage, especially since the EVM Emulator still does not support dynamic modifications to the coinbase variable.
- In `VerifierFflonk`, a misleading docstring states that the values are stored starting from the initial free memory pointer `0x80`, whereas the correct value is `0x00`.
- In the `verify` function of `DualVerifier` there is an obsolete comment stating that "the first element of `_recursiveAggregationInput` determines the verifier type (either FFLONK or PLONK)". This reflects an older version of code, since in the latest version, the verifier type is stored in the first element of the proof `_proof[0]`.
- The newly added input `_callConstructor` to `_performDeployOnAddress` is not documented in the function docstrings.
- The docstring description to `precreateEvmAccountFromEmulator` can be improved as the function does not only check if the contract can be deployed but also returns its deployment address.

Consider thoroughly documenting all functions and their parameters that are part of a contract's public API, and then updating the documentation to reflect the latest changes in functionality.

Update: *Resolved in pull request #1257, pull request #1275. The Matter Labs team stated:*

> *1. Coinbase should not be changed during the transaction (as expected from block-scope params). But its value isn't hardcoded in the EVM emulator, so it retrieves it from `SystemContext` contract.*
> *2. EVM emulator-related comments are addressed in PR #1257.*
> *3. Verifiers-related comments are addressed in PR #1275.*

Conclusion

This system upgrade introduced changes to two sets of contracts that were previously audited by OpenZeppelin: **Dual Verifier** and **EVM Equivalence**.

The Dual Verifier functionality supports a second verifier FFLONK, alongside the original PLONK, serving as a transitional step toward full FFLONK adoption. The latest changes include storing the verifier type flag into the proof, updating the verification key and other constants (e.g., offsets), and adapting the L1 dual verifier contracts for L2. In this part of the scope, we reported one low-severity vulnerability stemming from ambiguity in the verification key hash retrieval.

The EVM Equivalence part focuses on an EVM emulator contract that supports the execution of EVM bytecode within EraVM. These changes affect multiple contracts and were introduced to incorporate EVM equivalence functionality into the next ZKsync release. We reported a medium-severity vulnerability stemming from a bytecode hash restoration failure in EVM contracts.

Apart from the changes mentioned above, multiple other contracts were also diff-audited as they were affected by updates for the next release. We thank the Matter Labs team for their responsiveness and for promptly addressing any questions that arose in the course of the audit.