OpenZeppelin | security

Protocol Defense Audit

ZKsync

September 9, 2024

Table of Contents

Summary

Type	Layer 2
Timeline	From 2024-06-10 To 2024-06-21
Languages	Solidity & Yul

Total Issues	27 (20 resolved, 3 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	3 (2 resolved, 1 partially resolved)
Notes & Additional Information	24 (18 resolved, 2 partially resolved)

Scope

We audited Pull Request #524 of the matter-labs/era-contracts repository at commit 1957571.

In scope were the following files:

```
.
├── gas-bound-caller
│   └── contracts
│       ├── GasBoundCallerErrors.sol
│       └── GasBoundCaller.sol
├── l1-contracts
│   └── contracts
│       ├── bridge
│       │   ├── interfaces
│       │   │   ├── IL1ERC20Bridge.sol
│       │   │   ├── IL1SharedBridge.sol
│       │   │   ├── IL2Bridge.sol
│       │   │   └── IWETH9.sol
│       │   ├── L1ERC20Bridge.sol
│       │   └── L1SharedBridge.sol
│       ├── bridgehub
│       │   ├── Bridgehub.sol
│       │   └── IBridgehub.sol
│       ├── common
│       │   ├── Config.sol
│       │   ├── Dependencies.sol
│       │   ├── interfaces
│       │   │   └── IL2ContractDeployer.sol
│       │   ├── L1ContractErrors.sol
│       │   ├── L2ContractAddresses.sol
│       │   ├── libraries
│       │   │   ├── L2ContractHelper.sol
│       │   │   ├── SemVer.sol
│       │   │   ├── UncheckedMath.sol
│       │   │   └── UnsafeBytes.sol
│       │   ├── Messaging.sol
│       │   └── ReentrancyGuard.sol
│       ├── governance
│       │   ├── Governance.sol
│       │   └── IGovernance.sol
│       ├── state-transition
│       │   ├── chain-deps
│       │   │   ├── DiamondInit.sol
│       │   │   ├── DiamondProxy.sol
│       │   │   ├── facets
│       │   │   │   ├── Admin.sol
│       │   │   │   ├── Executor.sol
│       │   │   │   ├── Getters.sol
│       │   │   │   ├── Mailbox.sol
│       │   │   │   └── ZkSyncHyperchainBase.sol
│       │   │   └── ZkSyncHyperchainStorage.sol
│       │   ├── chain-interfaces
│       │   │   ├── IAdmin.sol
│       │   │   ├── IDiamondInit.sol
│       │   │   ├── IExecutor.sol
│       │   │   ├── IGetters.sol
│       │   │   ├── ILegacyGetters.sol
│       │   │   ├── IMailbox.sol
│       │   │   ├── ITransactionFilterer.sol
│       │   │   ├── IVerifier.sol
│       │   │   ├── IZkSyncHyperchainBase.sol
│       │   │   └── IZkSyncHyperchain.sol
│       │   ├── IStateTransitionManager.sol
│       │   ├── l2-deps
│       │   │   └── ISystemContext.sol
│       │   ├── libraries
│       │   │   ├── Diamond.sol
│       │   │   ├── LibMap.sol
│       │   │   ├── Merkle.sol
│       │   │   ├── PriorityQueue.sol
│       │   │   └── TransactionValidator.sol
│       │   ├── StateTransitionManager.sol
│       │   ├── ValidatorTimelock.sol
│       │   └── Verifier.sol
│       ├── upgrades
│       │   ├── BaseZkSyncUpgradeGenesis.sol
│       │   ├── BaseZkSyncUpgrade.sol
│       │   ├── UpgradeHyperchains.sol
│       │   ├── Upgrade_v1_4_1.sol
│       │   └── ZkSyncUpgradeErrors.sol
│       └── vendor
│           └── AddressAliasHelper.sol
├── l2-contracts
│   └── contracts
│       ├── bridge
│       │   ├── interfaces
│       │   │   ├── IL1ERC20Bridge.sol
│       │   │   ├── IL1SharedBridge.sol
│       │   │   ├── IL2SharedBridge.sol
│       │   │   ├── IL2StandardToken.sol
│       │   │   └── IL2WrappedBaseToken.sol
│       │   ├── L2SharedBridge.sol
│       │   ├── L2StandardERC20.sol
│       │   └── L2WrappedBaseToken.sol
│       ├── Dependencies.sol
│       ├── errors
│       │   └── L2ContractErrors.sol
│       ├── interfaces
│       │   ├── IPaymasterFlow.sol
│       │   └── IPaymaster.sol
│       ├── L2ContractHelper.sol
│       ├── SystemContractsCaller.sol
│       ├── TestnetPaymaster.sol
│       └── vendor
│           └── AddressAliasHelper.sol
└── system-contracts
    ├── bootloader
    │   └── bootloader.yul
    └── contracts
        ├── AccountCodeStorage.sol
        ├── BootloaderUtilities.sol
        ├── ComplexUpgrader.sol
        ├── Compressor.sol
        ├── Constants.sol
        ├── ContractDeployer.sol
        ├── DefaultAccount.sol
        ├── ImmutableSimulator.sol
        ├── interfaces
        │   ├── IAccountCodeStorage.sol
        │   ├── IAccount.sol
        │   ├── IBaseToken.sol
        │   ├── IBootloaderUtilities.sol
        │   ├── IComplexUpgrader.sol
        │   ├── ICompressor.sol
        │   ├── IContractDeployer.sol
        │   ├── IImmutableSimulator.sol
        │   ├── IKnownCodesStorage.sol
        │   ├── IL1Messenger.sol
        │   ├── IL2StandardToken.sol
        │   ├── IMailbox.sol
        │   ├── INonceHolder.sol
        │   ├── IPaymasterFlow.sol
        │   ├── IPaymaster.sol
        │   ├── IPubdataChunkPublisher.sol
        │   ├── ISystemContextDeprecated.sol
        │   ├── ISystemContext.sol
        │   └── ISystemContract.sol
        ├── KnownCodesStorage.sol
        ├── L1Messenger.sol
        ├── L2BaseToken.sol
        ├── libraries
        │   ├── EfficientCall.sol
        │   ├── RLPEncoder.sol
        │   ├── SystemContractHelper.sol
        │   ├── SystemContractsCaller.sol
        │   ├── TransactionHelper.sol
        │   ├── UnsafeBytesCalldata.sol
        │   └── Utils.sol
        ├── MsgValueSimulator.sol
        ├── NonceHolder.sol
        ├── PubdataChunkPublisher.sol
        ├── SystemContext.sol
        └── SystemContractErrors.sol
```

In addition, we diff-audited the matter-labs/era-contracts repository at HEAD commit 874bc6b against the BASE commit 7ccade5. The diff-audit covered the following changes

- Updating "zkSync" to "ZKsync"

- Code clean-up and formatting
- Removal of `upgradeSystemContextIfNeeded` function in `bootloader.yul`.

All resolutions and the final state of the audited codebase mentioned in this report are contained at commit 874bc6b.

System Overview

Introduction

The era-contracts repository is under continuous development and strives to be at the forefront of quality, maintainability, and readability. By constantly adopting the best practices, the goal is to make the code more scalable and robust.

This audit primarily focuses on reviewing the following updates to the codebase:

- Replacing string-based reverts with custom errors.
- Adding stricter `solhint` rules
- Adding support for version 5 of the OpenZeppelin library.
- Utilizing floating pragmas for interfaces and libraries.
- Implementing minor gas optimization changes.

Given the extensive list of the in-scope files, this audit is only a diff audit, primarily focusing on the changes made rather than the entire files.

Low Severity

L-01 Misleading Errors

Throughout the codebase, some errors are misleading or do not provide enough information about the root cause:

- The `ValueMismatch(uint256 expected, uint256 actual)` custom error is not suitable for use in scenarios where a token transfer does not transfer the exact specified amount due to fees or other non-standard transfer logic. For instance, in the `deposit` function of the `L1ERC20Bridge` contract, the error message does not provide relevant information for the end user as the user is not aware of the internal `amount` variable. This lack of clarity may lead users to retry the transaction with the other amount shown in the error instead of using a different token. A similar issue occurs in the `bridgehubDepositBaseToken` function of the `L1SharedBridge` contract. As such, consider changing the error message to clearer alternatives like `TokenNotSupported(address token)` or `TokensWithFeesNotSupported()`. The former is already included in `L1ContractErrors.sol` while the latter provides more information to the end user and is cheaper as it does not have parameters.

- The `unfreezeDiamond` function reverts with the `DiamondAlreadyFrozen` error when the diamond is not frozen. Consider adding a new error (e.g., `DiamondNotFrozen`) to clearly demonstrate the actual issue.

- Within the `Compressor` contract, the `ValuesNotEqual` error does not specify what values are being compared, making debugging difficult due to the lack of context. This error is used in multiple places for different types of comparisons (e.g., initial writes, enum indices, and final values), making it hard to trace the specific issue when the error is thrown. Consider creating specific errors for each type of value mismatch or include additional parameters that provide relevant context such as which part of the process failed.

- The `DictionaryLengthNotFourTimesSmallerThanEncoded` error does not accurately describe the condition being checked. The actual condition being checked is whether the dictionary length divided by 8 is greater than the encoded data length divided by 2. The error message should be more descriptive and precise to reflect this condition accurately.

Update: *Resolved in pull request #569 at commit 815b737.*

L-02 Inconsistent Input Validation

Throughout the codebase, several inconsistencies in input validations were found. The following is a non-extensive listing of instances that serve as a reference.

For example, in `UpgradeHyperchains`, the `upgrade` function validates the values of the `chainId`, the `bridgehubAddress`, the `stateTransitionManager`, and the `sharedBridgeAddress` parameters. However, the `chainAdmin` and `validatorTimelock` inputs lack validation, allowing to migrate the Era-blockchain to the hyperchain ecosystem without setting a valid admin or timelock validator. This could impact the finalization of transactions and disable all critical guarded functions for the admin if invalid values are passed to the `upgrade` function.

Moreover, in the `AdminFacet` contract, the following inputs are not validated in their respective functions:

- The `_stateTransitionManager` input is not validated in the `setStateTransitionManager` function.
- The `_newPendingAdmin` input is not validated in the `setPendingAdmin` function.
- The `_validator` input is not validated in the `setValidator` function.

The aforementioned instances illustrate inconsistencies in input validation when changing key ecosystem values. In favor of standardization, consider revisiting the input validation for all functions that modify fundamental values and try to adopt a consistent validation scheme.

Update: *Partially resolved in pull request #570 at commit f5ad651. The Matter Labs team stated:*

> *Given that these are only callable by the owner of the contract and used in scripts/tests we are less concerned with validation on the inputs for the additional cost.*

L-03 `getAllHyperchains` Function Reverts Due to Invalid Key Access

Recent changes to the `getAllHyperchains` function have introduced an issue where the function attempts to retrieve values from the `hyperchainMap` using indices rather than valid chain IDs. The previous implementation used `hyperchainMap.get(keys[i])`, where `keys[i]` are valid chain IDs. However, the new implementation uses `hyperchainMap.get(i)`, which refers to the loop counter. Since `i` may not correspond to a valid chain ID, the function may revert when accessing invalid keys such as chain ID 0 or 2. This results in the `getAllHyperchains` function reverting for all calls, rendering it unusable.

Consider reverting to the previous implementation which ensures that only valid chain IDs from the `keys` array are used to access the `hyperchainMap`. This will ensure that only valid chain IDs are used and prevent the function from reverting.

Update: *Resolved in pull request #571 at commit 7a7174e.*

Notes & Additional Information

N-01 Standardize Custom Error for Consistency

The contracts currently use multiple custom errors for the same underlying errors, resulting in inconsistency and potential confusion. Some examples are:

- `InvalidCaller` and `Unauthorized`
- `EmptyAddress` and `ZeroAddress`
- `WithdrawFailed` and `WithdrawalFailed`
- `ValuesNotEqual` and `ValueMismatch`
- `ProtocolVersionShouldBeGreater` and `ProtocolVersionTooSmall`
- `OnlyEraSupported` [1] [2] and `LegacyMethodIsSupportedOnlyForEra`

To enhance standardization and clarity, consider selecting and consistently using the error names that best reflect the error cause. This will help improve code readability and maintenance.

Update: *Resolved in pull request #572 at commit 96a53cc.*

N-02 Typographical Errors

The following typographical errors were identified in the codebase:

- `ShareadBridgeValueNotSet` should be `SharedBridgeValueNotSet`

- `_oldprotocolVersionDeadline` should be `_oldProtocolVersionDeadline`
- encure should be incur.

Consider fixing the aforementioned typographical errors in order to improve the readability of the codebase.

Update: *Resolved in pull request #573 at commit 1934420.*

N-03 Duplicate Error Import

The `L1SharedBridge` contract currently imports the `WithdrawalFailed` error twice. This redundancy can lead to confusion and maintainability issues.

Consider removing one of the import statements to enhance code clarity and maintainability.

Update: *Resolved in pull request #572 at commit 96a53cc.*

N-04 Risk with Floating Pragma and Yul Optimizer Bug

The audited refactor introduced the use of a floating pragma with a minimum version requirement of `0.8.20` for all interfaces and libraries. This change potentially exposes users to a known bug in the Yul optimizer, specifically related to the `FullInliner` step when using a custom optimizer step sequence.

Although the likelihood of encountering this bug is low, it can alter the behavior of contracts by reordering function call arguments with side effects. To prevent this issue, consider updating the floating pragma to start from Solidity version `0.8.21` instead of `0.8.20` or include a disclaimer to inform users about the potential risk.

Update: *Resolved in pull request #574 at commit e3423ed.*

N-05 Unused Errors

Throughout the codebase, there are a few instances of unused errors:

- `ProtocolVersionTooBig` in `L1ContractErrors.sol`
- `EncodingLengthMismatch` in `SystemContractErrors.sol`
- `InvalidData` in `SystemContractErrors.sol`

To improve the codebase's overall clarity, intentionality, and readability, consider removing any currently unused errors.

Update: *Resolved in [pull request #575](#) at commit [dff8431](#).*

N-06 Improved Organization of Error Files

Maintaining a clear and organized structure for error files is crucial for enhancing maintainability and readability. Currently, the errors within these files lack any consistent organization, making it difficult to identify and manage duplicate or unused errors.

Consider ordering errors alphabetically or grouping them by category. This approach will facilitate easier navigation and management of the error files, helping developers identify and resolve issues quickly.

Update: *Resolved in [pull request #576](#) at commit [a165cb2](#).*

N-07 Naming Issues

Throughout the codebase, several error declarations could be renamed to better reflect their purpose:

- `OperationShouldBeReady` can be renamed to `OperationMustBeReady`.
- `OperationShouldBePending` can be renamed to `OperationMustBePending`.
- `PubdataAllowanceAndGasLeftLessThanPubdataGasAndOverhead` can be renamed to `NotEnoughGasForPubdata`.
- `RevertedBatchBeforeNewBatch` can be renamed to `RevertedBatchNotAfterNewLastBatch`.
- `VerifyProofCommittedVerifiedMismatch` can be renamed to `VerifiedBatchesExceedsCommittedBatches`.
- `PubdataPerBatchIsLessThanTxn` could be renamed to `PriorityTxPubdataExceedsMaxPubDataPerBatch`.
- `MaxGasLessThanGasLeft` can be renamed to `InsufficientGasProvided`. The error declaration could benefit from two parameters for additional information like `uint256 requiredGas` and `uint256 providedGas`.

Consider addressing these naming issues to improve the readability of the codebase.

Update: *Resolved in [pull request #577](#) at commit [39d8e3c](#).*

N-08 Unnamed Error Parameters

Throughout the codebase, there are multiple error declarations with unnamed parameters:

- `LogAlreadyProcessed(uint8)`
- `InvalidCaller(address)`
- `UnimplementedMessage(string)`
- `PointEvalCallFailed(bytes)`

Instead of just specifying the type, including a descriptive name for each parameter makes the purpose of the error more clear and the code more self-documenting. Consider naming parameters within error declarations to significantly enhance code readability and maintainability.

***Update:** Resolved in pull request #578 at commit 1d0c4bb.*

N-09 Misleading Use of "I" Prefix in Abstract Contract

Using the "I" prefix for abstract contracts is generally not recommended as it is conventionally reserved for interfaces. Mixing this convention could lead to confusion. The file name `ISystemContract` uses this prefix even though it is an abstract contract.

Consider renaming this contract to avoid confusion. For example, use `SystemContractBase` or a similar name which indicates that it is an abstract contract rather than an interface.

***Update:** Resolved in pull request #579 at commit 2ff66f9*

N-10 Unused Enum

The `Global` enum in `SystemContractHelper.sol` is unused.

To improve the overall readability of the codebase, consider either using or removing any currently unused enums.

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

> *We decided to keep for now until we can be sure that this isn't used anywhere in our ecosystem.*

N-11 File and Contract Names Mismatch

Throughout the codebase, there are multiple file names containing contracts with different names:

- The `Admin.sol` file name does not match the `AdminFacet` contract name.
- The `Executor.sol` file name does not match the `ExecutorFacet` contract name.
- The `Getters.sol` file name does not match the `GettersFacet` contract name.
- The `Mailbox.sol` file name does not match the `MailboxFacet` contract name.

To make the codebase easier to understand for developers and reviewers, consider renaming the files to match the contract names.

Update: *Acknowledged, not resolved. The Matter Labs team stated:*

> *This is to make sure that we know these contracts are used within our DiamondProxy and not standalone.*

N-12 Misplaced Error Declarations

The following list of errors should be declared in `ZkSyncUpgradeErrors.sol`:

- `PatchCantSetUpgradeTxn`
- `L2UpgradeNonceNotEqualToNewProtocolVersion`
- `L2BytecodeHashMismatch`
- `ProtocolVersionTooSmall`
- `ProtocolVersionTooBig`
- `PreviousProtocolMajorVersionNotZero`
- `NewProtocolMajorVersionNotZero`
- `ProtocolVersionMinorDeltaTooBig`
- `PatchUpgradeCantSetDefaultAccount`
- `PatchUpgradeCantSetBootloader`

To improve standardization and clarity, consider declaring errors in their designated file.

Update: *Resolved in pull request #580 at commit 9ec05f3.*

N-13 Unused Named Return Variable

Named return variables are a way to declare variables that are meant to be used within a function's body for the purpose of being returned as that function's output. They are an alternative to explicit in-line `return` statements.

In `TestnetPaymaster.sol`, the `context` return variable for the `validateAndPayForPaymasterTransaction` function is unused.

Consider either using or removing any unused named return variables.

Update: *Resolved in pull request #581 at commit 8c5758c.*

N-14 Constants Not Using `UPPER_CASE` Format

Throughout the codebase, there are constants that have not been declared using the `UPPER_CASE` format:

- The `offset` constant declared on line 22 in `AddressAliasHelper.sol` (`l1-contracts` directory)
- The `offset` constant declared on line 22 in `AddressAliasHelper.sol` (`l2-contracts` directory)

According to the Solidity Style Guide, constants should be named in all capital letters with underscores separating words. For better code readability, consider following this convention.

Update: *Acknowledged, not resolved. The Matter Labs team stated:*

> *We decided to leave this as is for the time being and will do a deeper dive on potential implications when changing it.*

N-15 Unnecessary Casts

Throughout the codebase, there are instances of unnecessary casts:

- The `uint256(protocolVersion)` cast in the `StateTransitionManager` contract.
- The `bytes32(storedBatchZero)` cast in the `StateTransitionManager` contract.

- The `uint32(Utils.safeCastToU32(data.length))` cast in the `SystemContractsCaller` contract.
- The `uint32(Utils.safeCastToU32(data.length))` cast in the `SystemContractsCaller` contract.

To improve the overall clarity, intent, and readability of the codebase, consider removing unnecessary casts.

Update: *Resolved in pull request #582 at commit a0d9e9a.*

N-16 Risk of Disabling Valid Warnings

Solhint and Slither offer developers the flexibility to enforce the rules or detectors they find valuable. Even when rules are generally followed, it is common to use the `disable-next-line` comment to disable a rule for a specific line when necessary. However, an issue arises when two such comments are used consecutively, one for each tool. In this situation, the first comment does not affect the intended line if it is immediately followed by the other tool's comment.

To circumvent this issue within the codebase, the `solhint-disable` is used for Solhint and `slither-disable-next-line` for Slither. The problem with this approach is that `solhint-disable` disables the targeted rule for the rest of the file, potentially ignoring legitimate warnings in the subsequent lines rather than just the specific instance.

Examples of this misuse include:

- line 303 in `Mailbox.sol`
- line 54 in `AddressAliasHelper.sol`

Consider using `solhint-disable-line` directly on the target line or disable the rules for a group of lines as documented in the Solhint guidelines. This method prevents unintentional suppression of valid warnings and maintains code quality.

Update: *Resolved in pull request #583 at commit d05d292.*

N-17 Unjustified Use of `solhint-disable` Comments

Multiple instances of `solhint-disable` comments are present in the code without proper justification. This practice can give the impression that there is no intention to address the underlying warnings, which may lead to reduced code quality and maintainability. Examples include:

- Line 167 of `L1ERC20Bridge.sol`
- Line 308 of `Executor.sol`
- Line 3 of `SystemContractsCaller.sol`
- Line 56 of `SystemContractHelper.sol`

Consider adding comments to explain the rationale for disabling the rules or. Alternatively, fix the warnings to follow best practices.

Update: *Partially resolved in pull request #585 at commit 433be7e.*

N-18 Inconsistent Application of Solhint Rule `gas-length-in-loops`

The use of the `solhint-disable` statement for the `gas-length-in-loops` rule appears to be applied arbitrarily across the codebase. For instance, in the `Executor` contract, some `for` loops cache the length of arrays to optimize gas usage, while others use the `solhint-disable` statement without clear justification, such as avoiding a "stack too deep" error. This inconsistency may lead to unnecessary gas consumption and reduce the clarity of the code.

For consistency and gas savings, consider caching the length of arrays in all `for` loops. If there are specific reasons for not caching the length, such as the "stack too deep" error, inline comments should be added to explain these exceptions.

Update: *Resolved in pull request #584 at commit 0b7c75f.*

N-19 Duplicate Code

In `Executor.sol`, the computation on line 75 is unnecessarily repeated on line 78. This may lead to unnecessary gas consumption and reduce code clarity.

Rather than duplicating code, consider caching the values of complex computation in a variable and using it whenever the duplicated value is required.

Update: *Resolved in pull request #586 at commit 245d985.*

N-20 Overly Exposed State Variable

In `StateTransitionManager`, the state variable `protocolVersion`'s visibility is set to `public`. The `protocolVersion` represents a packed semantic version. The `getSemverProtocolVersion` function gives access to human-readable version. Having two separate endpoints to retrieve the version with different formatting could be confusing. In addition, the following constant state variables are only used in the contracts they are declared in. For instance:

- The `EMPTY_STRING_KECCAK` constant in the `AccountCodeStorage` contract
- The `GAS_TO_PASS` constant in the `MsgValueSimulator` contract
- The `MSG_VALUE_SIMULATOR_STIPEND_GAS` constant in the `MsgValueSimulator` contract

To better convey the intended use of functions and state variables to potentially realize some additional gas savings, consider restricting the aforementioned state variables to private.

Update: *Partially resolved in pull request #587 at commit bd3504a.*

N-21 Inconsistent Initialization of Local Variables

Throughout the codebase, some local variables are implicitly initialized to their default value. For instance:

- The `processedLogs` local variable in `Executor.sol`
- The `reconstructedChainedLogsHash` local variable in `L1Messenger.sol`
- The `reconstructedChainedMessagesHash` local variable in `L1Messenger.sol`
- The `reconstructedChainedL1BytecodesRevealDataHash` local variable in `L1Messenger.sol`

In other instances, local variables have been explicitly initialized to their default values. For instance:

- The `versionedHashIndex` variable in `Executor.sol`
- The `calldataPtr` variable in `L1Messenger.sol`
- The `costForPubdata` variable in `TransactionValidator.sol`

To improve the overall clarity, intent, consistency, and readability of the codebase, consider explicitly initializing all local variables.

Update: *Resolved in pull request #588 at commit f69ae9c.*

N-22 Todo Comments in the Code

During development, having well-described TODO comments will make the process of tracking and solving them easier. Without this information, these comments might age and important information for the security of the system might be forgotten by the time it is released to production. These comments should be tracked in the project's issue backlog and resolved before the system is deployed.

Throughout the codebase, multiple instances of TODO comments were found:

- The `TODO` comment in line 21 of `Config.sol`.
- The `TODO` comment in line 361 of `Mailbox.sol`.

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

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

> *We have tags to our internal task tracker to show we're planning on fixing the todo's.*

N-23 Misleading Documentation

Throughout the codebase, there are multiple instances of misleading documentation:

- This comment should be placed between lines 81 and 82.
- There is a missing word between "that the" and "is not overriden".
- The comment in `BaseZkSyncUpgradeGenesis.sol` describes the difference from `BaseZkSyncUpgrade.sol`. The following comparison change `> to >=` should be changed to `<= to <`.
- The comments about the `MAX_ALLOWED_FAIR_PUBDATA_PRICE` and `MAX_ALLOWED_FAIR_L2_GAS_PRICE` are incorrect. The values are in wei, not in gwei. Additionally, providing more context on why `2^64 - 1` was chosen would be beneficial.

Consider correcting the documentation to align with the code's behavior. This will help improve the clarity and readability of the codebase.

Update: *Resolved in pull request #589 at commit a6d48f4.*

N-24 Lack of Indexed Event Parameters

Indexing event parameters facilitates the task of off-chain services searching and filtering for specific events. In particular, the `NewPriorityRequest` event may benefit from the addition of the `indexed` keyword to the `txId` and `txHash` parameters.

Consider indexing these event parameters to avoid hindering off-chain services from searching and filtering for specific events.

Update: *Resolved in pull request #590 at commit ea7cd15.*

Conclusion

The era-contracts repository continues to evolve, focusing on quality, maintainability, and scalability by adopting best practices. This audit reviewed updates such as replacing string-based reverts with custom errors, adding stricter solhint rules, supporting version 5 of the OpenZeppelin library, utilizing floating pragmas for interfaces and libraries, and implementing minor gas optimizations.

The audit yielded three low-severity issues along with several recommendations for code improvement. The changes to the codebase were well-written, straightforward to follow, and well-documented. The Matter Labs team was very responsive throughout the engagement and answered all our questions.