CANTINA

zkSync Lido Bridge

Security Review

Cantina Managed review by:
Noah Marconi, Lead Security Researcher
Cccz, Security Researcher
Sujith Somraaj, Associate Security Researcher

October 19, 2023

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Directly* exploitable security vulnerabilities that need to be fixed.
High	Security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All high issues should be addressed.
Medium	Objective in nature but are not security vulnerabilities. Should be addressed unless there is a clear reason not to.
Low	Subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. When determining the severity one first needs to determine whether the finding is subjective or objective. All subjective findings are considered of Minor severity.

Next it is determined whether the finding can be regarded as a security vulnerability. Some findings might be objective improvements that need to be fixed, but do not impact the project's security overall (Medium).

Finally, objective findings of security vulnerabilities are classified as either critical or major. Critical findings should be directly vulnerable and have a high likelihood of being exploited. Major findings on the other hand may require specific conditions that need to be met before the vulnerability becomes exploitable.

2 Security Review Summary

The primary objective of this project is to establish a robust foundation to facilitate the seamless integration of Lido into zkSync protocol. Additionally, the project aims to expand the availability of Lido's staking token (stETH) within the decentralized finance (DeFi) ecosystem on zkSync network. This initiative is part of a broader, long-term strategy to enhance Lido's presence and functionality in the evolving landscape of blockchain technology and to expand the zkSync DeFi possibilities.

From Aug. 2nd - Aug. 18th the Cantina team conducted a review of zkSync Lido Bridge on commit hash a4d5c5...ea34ba. The team identified a total of **22** issues in the following risk categories:

- Critical Risk: 1
- High Risk: 0
- Medium Risk: 5
- Low Risk: 8
- Gas Optimizations: 2
- Informational: 6

In the issues below, fixes by the zkSync team are noted with commit hashes where applicable. The final commit hash following the fixes is fa6a77...54ac9d.

3 Findings

3.1 Critical Risk

3.1.1 `setGovernance` does not validate the caller

Severity: Critical Risk

Context: L2BridgeExecutor.sol#L78

Description: The `setGovernance` function allows anyone to update `_ethereumGovernanceExecutor`, which then can propose malicious action sets to the queue.

Recommendation:

- Remove the function as the `updateEthereumGovernanceExecutor` function serves the same purpose.
- If the function is still considered valid, add appropriate caller validations to ensure only allowed actors can call this function.

zkSync: We removed permissionless function in commit 8af93d.

Cantina: Confirmed.

3.2 Medium Risk

3.2.1 Unrecoverable states through misconfiguration

Severity: Medium Risk

Context: L2BridgeExecutor.sol#L64

Description: Governance may produce transactions which render the contracts into an unrecoverable state:

- `_ethereumGovernanceExecutor` may be set to an incorrect address, making it impossible to queue new actions.
- Setting `_delay` to `type(uint256).max` or other high value will make queuing excessively long.

Although the impact is high, the likelihood is low.

Recommendation: There are a number or ways to reduce the potential for misconfiguration:

1. Have configuration updates thoroughly reviewed before proceeding through governance.
2. Make use of two step ownership updates for owners and admin (e.g. `_ethereumGovernanceExecutor`).
3. Consider and upper bound on `_maximumDelay`.

zkSync: We introduced the `MAXIMUM_DELAY` constant set to 2 weeks to prevent values that are too high (see BridgeExecutorBase.sol#L19).

Cantina: Confirmed `MAXIMUM_DELAY` is now enforced.

3.2.2 Duplicate action is not enforced between actions queued in different blocks

Severity: Medium Risk

Context: BridgeExecutorBase.sol#L304-L314

Description: Queuing aims to prevent the same action from being queued if the action is already queued within a set that has not been executed/cancelled.

Queuing stores a boolean to indicate a particular `actionHash` is queued, cancelling or executing updates the bool to `false`. Expiring does not update the boolean.

The `actionHash`, however, is determined using the action details along with the current block's timestamp (`uint256 executionTime = block.timestamp + _delay`):

```
bytes32 actionHash = keccak256(
    abi.encode(
        targets[i],
        values[i],
        signatures[i],
        calldatas[i],
        executionTime,
        withDelegatecalls[i]
    )
    // ...
```

Duplicates may be introduced by queuing the same action in another block, thereby giving it a different `executionTime` and a different `actionHash`.

Caution: if the implementation is amended to strictly enforce no queued duplicates, when an action expires it will still be recorded as `_queuedActions[actionHash] = true;`

Recommendation: Consider whether duplicate detection is required at the action level at all. If duplicate avoidance is required, a refactor is required to detect duplicates without the use of the block's timestamp included in the `actionHash` (or other duplicate detection hash).

Note the caution above and in other tickets as a queued action with strict duplicate detection may DoS desired actions.

zkSync: Since enforcing the strict duplicate action detection may cause DoS, I believe it is not necessary to make changes to the current implementation of the `BridgeExecutorBase` contract. The current implementation only prevents duplicate actions to be queued in a single block, which is good enough in our use case.

Cantina: Acknowledged.

3.2.3 Updating `_gracePeriod` may cause expired actions to be executable.

Severity: Medium Risk

Context: BridgeExecutorBase.sol#L148-L151, BridgeExecutorBase.sol#L224-L238, BridgeExecutorBase.sol#L111-L113

Description: In `getCurrentState`, `_gracePeriod` is used to determine if the action is expired:

```
} else if (block.timestamp > actionsSet.executionTime + _gracePeriod) {
    return ActionsSetState.Expired;
} else {
    return ActionsSetState.Queued;
}
```

The problem here is that `_gracePeriod` can be changed by calling `updateGracePeriod` by actions, and if `_gracePeriod` gets larger, it may cause the expired action to be executable:

```
function updateGracePeriod(uint256 gracePeriod) external override onlyThis {
    if (gracePeriod < MINIMUM_GRACE_PERIOD) revert GracePeriodTooShort();
    _updateGracePeriod(gracePeriod);
}
```

To make matters worse, action in `Queued` state can be canceled, but action in `Expired` state cannot be canceled:

```
function cancel(uint256 actionsSetId) external override onlyGuardian {
    if (getCurrentState(actionsSetId) != ActionsSetState.Queued)
        revert OnlyQueuedActions();
```

This leads to the fact that if there is a malicious action that has expired, that action cannot be canceled, and once the `_gracePeriod` is increased, that action may be executed.

Recommendation: It is recommended to set the expiration time when queuing the action to prevent the state of the action from changing due to the increase of `_gracePeriod`.

```
  function _queue(
      address[] memory targets,
      uint256[] memory values,
      string[] memory signatures,
      bytes[] memory calldatas,
      bool[] memory withDelegatecalls
  ) internal {
      if (targets.length == 0) revert EmptyTargets();
      uint256 targetsLength = targets.length;
      if (
          targetsLength != values.length ||
          targetsLength != signatures.length ||
          targetsLength != calldatas.length ||
          targetsLength != withDelegatecalls.length
      ) revert InconsistentParamsLength();

      uint256 actionsSetId = _actionsSetCounter;
      uint256 executionTime = block.timestamp + _delay;
+     uint256 expireTime = executionTime + _gracePeriod;
```

zkSync: Fixed based on your recommendation, check commit 251bda.

Cantina: Confirmed.

3.2.4 When `_ethereumGovernanceExecutor` is updated to an EOA account, the contract cannot be recovered

Severity: Medium Risk

Context: L2BridgeExecutor.sol#L63-L71, ZkSyncBridgeExecutor.sol#L11-L17

Description: The `onlyEthereumGovernanceExecutor` modifier will always de-alias `msg.sender`. And it works well when `_ethereumGovernanceExecutor` is a contract address:

```
modifier onlyEthereumGovernanceExecutor() override {
    if (
        AddressAliasHelper.undoL1ToL2Alias(msg.sender) !=
        _ethereumGovernanceExecutor
    ) revert UnauthorizedEthereumExecutor();
    _;
}
```

However, `updateEthereumGovernanceExecutor` allows updating `_ethereumGovernanceExecutor` to an arbitrary address.

```
function updateEthereumGovernanceExecutor(
    address ethereumGovernanceExecutor
) external onlyThis {
    emit EthereumGovernanceExecutorUpdate(
        _ethereumGovernanceExecutor,
        ethereumGovernanceExecutor
    );
    _ethereumGovernanceExecutor = ethereumGovernanceExecutor;
}
```

If `_ethereumGovernanceExecutor` is updated to an EOA account, and when the EOA account wants to queue actions, `onlyEthereumGovernanceExecutor` modifier will de-alias the EOA account address to an unknown address, thus making `onlyEthereumGovernanceExecutor`'s validation fail and ZkSyncBridgeExecutor cannot recover due to the unavailability of queuing actions.

The likelihood of updating `_ethereumGovernanceExecutor` to an EOA account is low, but due to the high impact, this issue is considered as medium risk.

Recommendation: There are code based and docs based options:

- Since it is unable to know if the address of L1 is a contract address on L2, it is possible to implement a method dedicated to updating the `_ethereumGovernanceExecutor` of L2 from L1, and to check if the address is a contract address of L1.
- Well-documented the risk of updating `_ethereumGovernanceExecutor` to EOA account.

zkSync: Changes have been introduced to the ZkSyncBridgeExecutor contract that have enabled for EOA or a contract on L1 to become the governance executor.

In case the governance executor is a contract on L1, aliasing from L1 to L2 address must be performed in an off-chain script before assinging/updating the governor address. To aliase the contract's address from L1 to L2 (or vice versa) in a script, the zksync-web3 npm package is used.

Since the address of the EOA is same on L1 and L2, it requires no aliasing (check commit 6f8f11).

Cantina: Confirmed.

3.2.5 Bridge Executor `delegatecall` can bring governance into a unexpected or unusable states

Severity: Medium Risk

Context: BridgeExecutorBase.sol#L9375

Description: The `delegatecall` allows the owner (via `L1Executor.sol`) to intentionally or accidentally update sensitive state variables and can execute malicious actions like:

- **Set grace periods less than** `MINIMUM_GRACE_PERIOD`:
  - Delegate calling to a contract with a different `MINIMUM_GRACE_PERIOD` or direct access to `_gracePeriod` to circumvent the constant `MINIMUM_GRACE_PERIOD`, which, when set to `0`, can effectively block all actions from execution.
- **Change** `actionSet`**'s status**:
  - Activating an already executed action `actionsSetId`.
  - Switching a canceled/executed `actionSetId` to status `Queued`.
  - Increase grace period and switch `actionSetId` status, effectively executing any past action sets in the future.
  - Switching an upcoming `actionSetId` status to executed, preventing delivery/execution of future events.
- **Change** `actionHash`**'s status**:
  - Setting `_queuedActions[actionHash] = true` (particularly troublesome when duplicate actions are not permitted).
- **Change** `actionSet`**'s target**:
  - Change the target address during execution and reset it after execution, resulting in false positives about the execution status.
- **Set** `actionSetCounter` **to** `type(uint256).max`:
  - Can permanently push the governance contract into unusable state by setting the `actionSetCounter` to a max value of `uint256`.
- **Set** `_delay` **to** `type(uint256).max` **or other high value**:
  - Can make queuing excessively long
- **Add and execute new queue transactions**:
  - By effectively modifying state variables like `_delay` and `_gracePeriod`, actionsSets can be proposed and executed in the one atomic transaction.

Recommendation: There are three options for consideration:

- Option 1: The `ZkSyncBridgeExecutor.sol` contract can be separated into two contracts, not allowing some variables to be changed during `delegatecall`.
- Option 2: The `delegatecall` can be removed.
- Option 3: The risks associated with malicious `delegatecall` in `BridgeExecutorBase.sol` should, at minimum, be well documented, and extreme caution should be exercised whenever used.

zkSync: To remove the risks we have deleted the `delegatecall` option in commit 3d21e5.

Cantina: Confirmed.

3.3 Low Risk

3.3.1 `_disableInitializers` not used where indicated

Severity: Low Risk

Context: L1ERC20Bridge.sol#L53

Description: A comment indicates `_disableInitializers` is used in the constructor when it is not present. An uninitialised implementation contract may be initialized maliciously.

Recommendation: For safety add the function to the constructor.

zkSync: Resolved by adding `_disableInitializers` to constructor function. See L1ERC20Bridge.sol#L53-L55 and see commit a01cfe.

Cantina: Confirmed.

3.3.2 Inconsistent use of storage gaps in upgradeable contracts

Severity: Low Risk

Context: BridgeableTokensUpgradable.sol, BridgingManager.sol, AccessControl import in BridgingManager.sol, L2ERC20Bridge.sol, L1ERC20Bridge.sol, ReentrancyGuard import in L1ERC20Bridge.sol, L1Executor.sol, BridgeExecutorBase.sol, L2BridgeExecutor.sol, ZkSyncBridgeExecutor.sol, L2CrossDomainEnabled.sol, ERC20BridgedUpgradeable, ERC20CoreUpgradeable, ERC20MetadataUpgradeable

Description: The `ERC20PermitUpgradeable` and `NoncesUpgradeable` contracts make use of storage gaps for safety: `uint256[N] private __gap;` however, many of the other upgradeable contracts do not. Furthermore, not all upgradeable contracts import from the `@openzeppelin/contracts-upgradeable` package.

Recommendation: There are two options to consider for all of the upgradeable contracts:

1. For consistency and safety, prefer `@openzeppelin/contracts-upgradeable` for upgradeable contracts and make use of storage gaps, reducing the opportunity for accidental storage slot collisions.
2. Consider EIP-1967-like manually derived storage slots.

zkSync: We have added storage gaps to some contracts (e.g. `BridgeableTokensUpgradable`), but we don't see the need for all contract to have the storage gaps, since many of them implement just a specific set of functionalities that won't require storage updates, for example `BridgingManager`.

Cantina: Acknowledged.

3.3.3 Address validation

Severity: Low Risk

Context: L2BridgeExecutor.sol#L33, L2BridgeExecutor.sol#L64

Description: The `L2BridgeExecutor.constructor` and `L2BridgeExecutor.updateEthereumGovernanceExecutor` do not include a zero address check.

Recommendation: Protections can be added to the contracts themselves or to the deploy process:

1. Add a zero address check to avoid a misconfigured updates.
2. Consider using a factory based deployment similar to how the `L1ERC20Bridge` initialization doubles as a factory-like deployer for the `L2ERC20Bridge`.

zkSync: Zero address check added, check commit d8ba7c.

Cantina: Confirmed.

3.3.4 Too many actions in the set may cause execution to fail due to exceeding the block gas limit

Severity: Low Risk

Context: BridgeExecutorBase.sol#L281-L319, BridgeExecutorBase.sol#L81-L94

Description: There is no limit on the amount of actions in a set, so if there are too many actions in a set, the execution may fail due to exceeding the block gas limit.

```
function execute(uint256 actionsSetId) external payable override {
    if (getCurrentState(actionsSetId) != ActionsSetState.Queued)
        revert OnlyQueuedActions();

    ActionsSet storage actionsSet = _actionsSets[actionsSetId];
    if (block.timestamp < actionsSet.executionTime)
        revert TimelockNotFinished();

    actionsSet.executed = true;
    uint256 actionCount = actionsSet.targets.length;

    bytes[] memory returnedData = new bytes[](actionCount);
    for (uint256 i = 0; i < actionCount; ) {
    // ...
```

Recommendation: It is recommended to limit the amount of actions in the set when queuing.

zkSync: To stay on a safe side we limit the number of actions in the set to 3 (see BridgeExecutorBase.sol#L280).

Cantina: Confirmed.

3.3.5 Guardian may prevent their own removal and dos `updateGuardian`

Severity: Low Risk

Context: BridgeExecutorBase.sol#L137

Description: When governance decides to remove a guardian account and replace it with a new one, the existing guardian may cancel the action.

Recommendation: A code based approach or an offchain approach may be used to address this topic:

1. Consider a refactor to stop the guardian from cancelling this particular function. Caution would be needed in implementing as the restriction should be for the particular action and not the broader `actionSet`.
2. The guardian may DoS any function and should be carefully chosen. A guardian is effectively a trusted party in the system.

zkSync: The guardian is a trusted party and caution should be exercised when assigning the guardian role due to the guardian's ability to cancel proposals that contain a function call for updating the guardian's address. To prevent this from happening, the guardian role can remain uninitialized (role assigned to zero address) which would prohibit canceling proposals in general. We added additional docs to emphasis this, check commit 6053e2.

Cantina: Acknowledged.

3.3.6 Prefer manual storage slots with unknown preimage

Severity: Low Risk

Context: ERC20MetadataUpgradeable.sol#L22, BridgingManager.sol#L31-L32, BridgingManager.sol#L31-L32

Description: A number of the contract use a manually constructed storage slot by passing a string to `keccak256` such as `keccak256("BridgingManager.bridgingState")`. The preimage of these storage slots are known and may be vulnerable to future malicious upgrades combined with targeted payloads.

See EIP 1967 comment and Eth Magicians discussion for context.

Recommendation: Subtract `1` from the hashed value as shown in the example code snippet below:

```
bytes32(uint256(keccak256("BridgingManager.bridgingState")) - 1);
```

zkSync: Resolved based on recommendation in commit 179933.

Cantina: Confirmed for in scope contracts. The out of scope BridgingManager.sol#L31-L32 is unmodified.

3.3.7 `package.json` dependencies with vulnerability fixes available

Severity: Low Risk

Context: package.json#L43-L85

Description: `npm audit` shows a number of patches available for dependencies correcting issues of varying severity.

Recommendation: Where possible apply patches or update to later versions of npm based dependencies.

zkSync: Out of scope for now, we need to do this together with the Lido team since there are a lot of outdated dependencies they already use for other bridges in the repo.

Cantina: Acknowledged.

3.3.8 Ownership and contract upgrade management and timelocks

Severity: Low Risk

Context: L1Executor.sol#L9C24-L9C42, OssifiableProxy.sol#L52-L74, BridgingManager.sol#L21-L28

Description: Some admin level state changes are not timelocked:

- The `owner` of the `L1Executor`.
- The `DEFAULT_ADMIN_ROLE` may make role updates, e.g. changing deposit/withdrawal enabler/disabler roles to accounts without timelocks.
- The `OssifiableProxy` admin.
- The `OssifiableProxy` implementation.

The ZkSync team has informed us the deposit/withdrawal enabler/disabler roles are to route through the governance contract which uses queuing delays as a timelock.

Further, the intended use of `DEFAULT_ADMIN_ROLE` is temporary (see `initialize-bridge-roles.ts`).

Recommendation: The following code updates and post deploy checks are recommended:

1. Ensure all admin actions are either routed through governance or have appropriate timelocks in place.
2. Prefer two step ownership transfers to avoid accidental assignment to addresses such as `address(0)` or L2 aliased contract addresses on L1. See OZ's Ownable2Step.
3. Prefer OZ's AccessControlDefaultAdminRules for `DEFAULT_ADMIN_ROLE`.
4. Post deployment, confirm an EOA no longer has control over the `DEFAULT_ADMIN_ROLE`.

zkSync: After the discussion with Lido team we decided not to go with the 2 step approach for L1Executor owner and not to use OZ's AccessControlDefaultAdminRules for DEFAULT_ADMIN_ROLE on BridgingManager contract. The motivation for this decision was the fact that we need the deployer to be the initial admin for the L1 and L2 bridges, so it can grant the proper roles for deposit/withdrawal enablers/disablers, and then transfer the ownership and DEFAULT_ADMIN_ROLE to Lido Agent without the need to do the on-chain votings from the Lido DAO in order to accept ownership or DEFAULT_ADMIN_ROLE role (it would be cumbersome because the Aragon process has a 1 month-wide cadence) (see commit 8e7361).

Cantina: Acknowledged.

3.4 Gas Optimization

3.4.1 Reverting on L1 due to zero address `_l2Receiver` will save gas

Severity: Gas Optimization

Context: L1ERC20Bridge.sol#L120

Description: There is zero address check on `_l2Receiver` until the transaction attempts to execute on L2. This comes at the expense of L1 gas to initiate the tx, L2 gas for the tx to fail, and finally L1 gas to `claimFailedDeposit`.

Recommendation: Perform a zero address check for `_l2Receiver` on the L1 side.

zkSync: Resolved, zero address check is added in L1ERC20Bridge.sol#L153.

Cantina: Confirmed.

3.4.2 Token and Bridge data may be immutable

Severity: Gas Optimization

Context: L1Executor, L1ERC20Bridge, L1ERC20Bridge.l2Bridge, BridgeableTokensUpgradable.sol#L11-L14, L1ERC20Bridge.sol#L216

Description: In the token bridge contracts the `zkSyncAddress` is stored as an immutable, while in the `L1Executor` it is passed in as an argument.

In the L1 and L2 bridge contracts the token addresses are passed around as arguments and compared to storage variables. The pattern is consistent with the more generalised bridging contracts in the `@matterlabs/zksync-contracts` package.

Given these addresses are not intended to change, there is unnecessary gas overhead in passing/storing/reading these variables.

Recommendation: Exercise caution if ever upgrading and moving from constant to storage or vice versa:

1. For governance, store the `zkSyncAddress` as a constant or immutable addresses.
2. For the bridges store `l1Token`, `l2Token`, `l1Bridge`, and `l2Bridge` as constants or immutable addresses.
3. Do not pass `l1Token` as data from L1 to L1.
4. Store `gettersData` on the L2 side as immutable or consider not logging it at all.

zkSync: The following things are fixed:

- Removed `gettersData` function from the `L1ERC20Bridge` contract.
- `_data` argument is no longer used in `finalizeDeposit` function in the `L1ERC20Bridge` contract.
- `_data` argument is no longer emitted in `finalizeDeposit` event in the `L1ERC20Bridge` contract.
- The address of zksync is passed as an argument during initialization in the `L1Executor` contract.

Cantina: Confirmed list of changes, which are included in commit 58fe11. *Note: reading the address of zksync from storage does incur an extra `sload` cost. If the address is certain to never change, consider making it a `constant` or `immutable`.*

3.5 Informational

3.5.1 Increase test coverage

Severity: Informational

Context: Global

Description: There is less than complete test coverage of key contracts under review. Adequate test coverage and regular reporting is an essential process in ensuring the codebase works as intended. Insufficient code coverage may lead to unexpected issues and regressions arising due to changes in the underlying smart contract implementation.

Recommendation: Add to test coverage ensuring all execution paths are covered. A draft set of test stubs is available here. See also Paul R Berg's BTT thread and talk.

3.5.2 Mapping may be simplified as there is only one L2 token address

Severity: Informational

Context: L1ERC20Bridge.sol#L46

Description: Given there is only one L1 token address, the extra nesting of the `depositAmount` mapping is not needed.

Recommendation: Simplify the mapping to not consider the L1 token:

```
- /// @dev A mapping account => L1 token address => L2 deposit transaction hash => amount
+ /// @dev A mapping account => L2 deposit transaction hash => amount
  /// @dev Used for saving the number of deposited funds, to claim them in case the deposit transaction will
  ↪  fail
- mapping(address => mapping(address => mapping(bytes32 => uint256)));
+ mapping(address => mapping(bytes32 => uint256));
  public depositAmount;
```

zkSync: This is resolved based on your recommendation, check commit a6dab8.

Cantina: Confirmed.

3.5.3 Inconsistent L2 Bridge interfaces in use

Severity: Informational

Context: IL2ERC20Bridge.sol

Description: The `L2ERC20Bridge` implements the `IL2ERC20Bridge` interface. This interface varies slightly from the `@matterlabs/zksync-contracts` package in that events are added and the `_` is a suffix rather than a prefix on function arguments. It also differs from the `zksync/l1/contracts/interfaces/IL2ERC20Bridge.sol` version which only has an `initializate` function.

Both interfaces are in use either on the L1 side or the L2 side.

Further, the events emitted in the `zksync/l2/contracts/interfaces/IL2ERC20Bridge.sol` interface do differ from the events in the `contracts/optimism/interfaces/IL2ERC20Bridge.sol` version of the interface.

Recommendation: Consider if possible to adopt and use single interface. Where differences are needed, consider using the `@matterlabs` published interface and supplementing with an extension interface only.

zkSync: Bridge interfaces are aligned with the ones from `@matterlabs/zksync-contracts` package, with the only difference that `DepositInitiated` event contains the `refundRecipient` address based on your recommendation in the issue "`DepositInitiated` event does not log the `refundRecipient`" (check commit a6dab8).

Cantina: Acknowledged interface updates.

3.5.4 Avoid variable shadowing

Severity: Informational

Context: L1ERC20Bridge.sol#L285, L1ERC20Bridge.sol#L322

Description: The storage variable `l1Token` is shadowed locally in some functions.

Recommendation: Rename the local variables to avoid variable shadowing.

zkSync: Resolved based on your recommendation, check commit f50f44.

Cantina: Confirmed.

3.5.5 `DepositInitiated` event does not log the `refundRecipient`

Severity: Informational

Context: L1ERC20Bridge.sol#L187

Description: The `refundRecipient` is contextually relevant but is not included in the `DepositInitiated` event.

Recommendation: Add `refundRecipient` to the event.

zkSync: Resolved, check PR 45.

Cantina: Confirmed.

3.5.6 Interface naming inconsistent with `@matterlabs` repo

Severity: Informational

Context: IL2Messenger

Description: The local `zksync/l2/contracts/interfaces/IL2Messenger.sol` names the interface with `L2` in the prefix whereas the `@matterlabs` repo refers to this contract as the `L1Messenger`.

Recommendation: To avoid confusion or potential future error, rename to match `@matterlabs`.

zkSync: Fixed, L1Messenger from `@matterlabs` repo is used, check PR 45.

Cantina: Confirmed.