Linea Audit Report

Prepared by Cyfrin

Version 2.0

Lead Auditor

Dacian

November 28, 2024

Contents

1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

2 Disclaimer

The Cyfrin team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Linea is a Type 2 zkEVM L2 rollup with full EVM equivalence that aims to provide the "Ethereum experience" at scale.

5 Audit Scope

The audit scope was limited to Solidity smart contracts primarily covering the messaging service, token bridge and rollup, aiming to audit the following new developments:

- refactoring of needless/ineffective checks during blob submission and finalization
- changes to events emitted during blob submission and finalization to simplify L2 state reconstruction and remove obsolete elements
- more granular role-based access control
- changes to `TokenBridge` storage
- new feature to allow a back-up address the ability to finalize if no finalization has occurred for 6 months
- removal of previous "training wheel" feature which allowed finalization without proof
- increase of Solidity version to 0.8.26
- optimized hashing during messaging sending and claiming
- a collection of assorted minor Info/QA/gas improvements from previous audits
- an additional commit e62791d in PR266
- PRs 298, 299, 300, 301, 302, 303, 304, 305, 306, 307 from OpenZeppelin's audit incorporated into commit adb097a

The expected deployment bytecode has not been supplied as other audits are still on-going but once all audits are concluded the expected bytecode should align on a single final commit.

All code relating to zk provers and other code outside the smart contract layer was out of scope. The following contracts were included in the scope for this audit:

```
contracts/LineaRollup.sol
contracts/ZkEvmV2.sol
contracts/interfaces/IGenericErrors.sol
contracts/interfaces/IMessageService.sol
contracts/interfaces/IPauseManager.sol
contracts/interfaces/IPermissionsManager.sol
contracts/interfaces/IRateLimiter.sol
contracts/interfaces/l1/IL1MessageManager.sol
contracts/interfaces/l1/IL1MessageManagerV1.sol
contracts/interfaces/l1/IL1MessageService.sol
contracts/interfaces/l1/ILineaRollup.sol
contracts/interfaces/l1/IPlonkVerifier.sol
contracts/interfaces/l1/IZkEvmV2.sol
contracts/interfaces/l2/IL2MessageManager.sol
contracts/interfaces/l2/IL2MessageManagerV1.sol
contracts/lib/L2MessageServicePauseManager.sol
contracts/lib/LineaRollupPauseManager.sol
contracts/lib/PermissionsManager.sol
contracts/lib/TokenBridgePauseManager.sol
contracts/lib/PauseManager.sol
contracts/lib/Utils.sol
contracts/messageService/MessageServiceBase.sol
contracts/messageService/l1/L1MessageManager.sol
contracts/messageService/l1/L1MessageService.sol
contracts/messageService/l1/TransientStorageReentrancyGuardUpgradeable.sol
contracts/messageService/l1/v1/L1MessageManagerV1.sol
contracts/messageService/l1/v1/L1MessageServiceV1.sol
contracts/messageService/l2/L2MessageManager.sol
contracts/messageService/l2/L2MessageService.sol
contracts/messageService/l2/v1/L2MessageManagerV1.sol
contracts/messageService/l2/v1/L2MessageServiceV1.sol
contracts/messageService/lib/MessageHashing.sol
contracts/messageService/lib/RateLimiter.sol
contracts/messageService/lib/SparseMerkleTreeVerifier.sol
contracts/messageService/lib/TransientStorageHelpers.sol
contracts/tokenBridge/TokenBridge.sol
contracts/tokenBridge/interfaces/ITokenBridge.sol
contracts/tokenBridge/lib/StorageFiller39.sol
```

6 Executive Summary

Over the course of 11 days, the Cyfrin team conducted an audit on the Linea smart contracts provided by Linea. In this period, a total of 12 issues were found.

The findings consist of 1 Medium and 1 Low severity issues with the remainder being informational and gas optimizations.

Both issues related to the refactoring of checks which occur during blob submission and finalization:

- M-1 allowed an operator to finalize for a non-existent `finalShnarf`
- L-1 allowed an operator to submit data via `subDataAsCalldata` for an invalid `parentShnarf`

TokenBridge Storage Upgrade

To support the more granular role-based access control the `TokenBridge` contract had its inheritance structure changed which resulted in significant changes to its storage layout. Given that this is a live contract which will be upgraded that holds significant funds, we were asked to carefully examine these changes.

We used `forge inspect -R "@openzeppelin/=contracts/node_modules/@openzeppelin/" --hardhat TokenBridge storageLayout --pretty` on both the new and old contracts to carefully examine their exact storage layout:

NEW TokenBridge Layout	Type	Slot	Offset	Bytes
# identical				
_initialized	uint8	0	0	1
_initializing	bool	0	1	1
# overwrites gaps, `_paused`, `_owner` and `_pendingOwner`				
_status	uint256	1	0	32
__gap	uint256[49]	2	0	1568
__gap	uint256[50]	51	0	1600
__gap	uint256[50]	101	0	1600
_roles	mapping(bytes32 => struct ACU.RoleData)	151	0	32
__gap	uint256[49]	152	0	1568
# identical				
messageService	contract IMessageService	201	0	20
remoteSender	address	202	0	20
__base_gap	uint256[10]	203	0	320
# overwrites `_status` and gap				
pauseTypeStatuses	mapping(bytes32 => bool)	213	0	32
_pauseTypeStatusesBitMap	uint256	214	0	32
# new				
_pauseTypeRoles	mapping(enum IPauseManager.PauseType => bytes32)	215	0	32
_unPauseTypeRoles	mapping(enum IPauseManager.PauseType => bytes32)	216	0	32
__gap	uint256[7]	217	0	224
__gap_39	uint256[39]	224	0	1248
# identical				
tokenBeacon	address	263	0	20
nativeToBridgedToken	mapping(uint256 => mapping(address => address))	264	0	32
bridgedToNativeToken	mapping(address => address)	265	0	32
sourceChainId	uint256	266	0	32
targetChainId	uint256	267	0	32
__gap	uint256[50]	268	0	1600

OLD TokenBridge Layout	Type	Slot	Offset	Bytes
identical				
_initialized	uint8	0	0	1
_initializing	bool	0	1	1
# overwritten				
__gap	uint256[50]	1	0	1600
_paused	bool	51	0	1
__gap	uint256[49]	52	0	1568
_owner	address	101	0	20
__gap	uint256[49]	102	0	1568
_pendingOwner	address	151	0	20

```
| __gap                  | uint256[49]                                          | 152  | 0      | 1568  |

# identical
| messageService         | contract IMessageService                             | 201  | 0      | 20    |
| remoteSender           | address                                              | 202  | 0      | 20    |
| __base_gap             | uint256[10]                                          | 203  | 0      | 320   |

# overwritten
| _status                | uint256                                              | 213  | 0      | 32    |
| __gap                  | uint256[49]                                          | 214  | 0      | 1568  |

# identical
| tokenBeacon            | address                                              | 263  | 0      | 20    |
| nativeToBridgedToken   | mapping(uint256 => mapping(address => address))      | 264  | 0      | 32    |
| bridgedToNativeToken   | mapping(address => address)                          | 265  | 0      | 32    |
| sourceChainId          | uint256                                              | 266  | 0      | 32    |
| targetChainId          | uint256                                              | 267  | 0      | 32    |
| __gap                  | uint256[50]                                          | 268  | 0      | 1600  |
```

- slot 51 `_paused` is repurposed into a gap. Slot 51 is not wiped as the `TokenBridge` will not be paused prior to upgrade
- slot 101 `_owner` is repurposed into a gap. Slot 101 is wiped as part of the upgrade
- slot 151 `_pendingOwner` is repurposed into `_roles`. Slot 151 is not wiped as the `TokenBridge` will not have a pending owner prior to upgrade
- slot 213 `_status` is moved into slot 1 which was previously a gap. Slot 213 is wiped as part of the upgrade then repurposed to store `pauseTypeStatuses`
- slot 216 previously unused now stores `_unPauseTypeRoles`

We have verified using `cast storage 0x051F1D88f0aF5763fB888eC4378b4D8B29ea3319` for L1 and `cast storage 0x353012dc4a9A6cF55c941bADC267f82004A8ceB9` for L2 that the gap slots in the old storage currently deployed are already zero'd such that existing gaps can be safely repurposed in the new storage layout. The upgrade plan from the old `TokenBridge` storage layout to the new one appears safe.

LineaRollup Storage Upgrade

Some necessary changes have also occurred in the `LineaRollup` storage layout which we have analyzed using `forge inspect -R "@openzeppelin/=contracts/node_modules/@openzeppelin/" --hardhat --evm-version cancun LineaRollup storageLayout --pretty` on both new and old contracts:

```
| NEW LineaRollup Layout           | Type                                    | Slot | Offset | Bytes |
|----------------------------------|-----------------------------------------|------|--------|-------|

# identical
| _initialized                     | uint8                                   | 0    | 0      | 1     |
| _initializing                    | bool                                    | 0    | 1      | 1     |
| __gap                            | uint256[50]                             | 1    | 0      | 1600  |
| __gap                            | uint256[50]                             | 51   | 0      | 1600  |
| _roles                           | mapping(bytes32 => ACU.RoleData)        | 101  | 0      | 32    |
| __gap                            | uint256[49]                             | 102  | 0      | 1568  |
| periodInSeconds                  | uint256                                 | 151  | 0      | 32    |
| limitInWei                       | uint256                                 | 152  | 0      | 32    |
| currentPeriodEnd                 | uint256                                 | 153  | 0      | 32    |
| currentPeriodAmountInWei         | uint256                                 | 154  | 0      | 32    |
| __gap                            | uint256[10]                             | 155  | 0      | 320   |
| outboxL1L2MessageStatus          | mapping(bytes32 => uint256)             | 165  | 0      | 32    |
| inboxL2L1MessageStatus           | mapping(bytes32 => uint256)             | 166  | 0      | 32    |
| __gap_ReentrancyGuardUpgradeable | uint256[50]                             | 167  | 0      | 1600  |
| pauseTypeStatuses                | mapping(bytes32 => bool)                | 217  | 0      | 32    |
```

| _pauseTypeStatusesBitMap | uint256 | 218 | 0 | 32 |

overwrites gap
| _pauseTypeRoles | mapping(enum PauseType => bytes32) | 219 | 0 | 32 |

new
| _unPauseTypeRoles | mapping(enum PauseType => bytes32) | 220 | 0 | 32 |
| __gap | uint256[7] | 221 | 0 | 224 |

identical
nextMessageNumber	uint256	228	0	32
_messageSender	address	229	0	20
__gap	uint256[50]	230	0	1600
currentTimestamp	uint256	280	0	32
currentL2BlockNumber	uint256	281	0	32
stateRootHashes	mapping(uint256 => bytes32)	282	0	32
verifiers	mapping(uint256 => address)	283	0	32
__gap	uint256[50]	284	0	1600
rollingHashes	mapping(uint256 => bytes32)	334	0	32
_messageClaimedBitMap	struct BitMaps.BitMap	335	0	32
l2MerkleRootsDepths	mapping(bytes32 => uint256)	336	0	32
__gap_L1MessageManager	uint256[50]	337	0	1600
systemMigrationBlock	uint256	387	0	32
__gap_L1MessageService	uint256[50]	388	0	1600
dataFinalStateRootHashes	mapping(bytes32 => bytes32)	438	0	32
dataParents	mapping(bytes32 => bytes32)	439	0	32
dataShnarfHashes	mapping(bytes32 => bytes32)	440	0	32
dataStartingBlock	mapping(bytes32 => uint256)	441	0	32
dataEndingBlock	mapping(bytes32 => uint256)	442	0	32
currentL2StoredL1MessageNumber	uint256	443	0	32
currentL2StoredL1RollingHash	bytes32	444	0	32
currentFinalizedShnarf	bytes32	445	0	32

renamed & re-purposed, existing data kept
| blobShnarfExists | mapping(bytes32 => uint256) | 446 | 0 | 32 |

identical
| currentFinalizedState | bytes32 | 447 | 0 | 32 |

new
| fallbackOperator | address | 448 | 0 | 20 |

OLD LineaRollup Layout	Type	Slot	Offset	Bytes

identical
_initialized	uint8	0	0	1
_initializing	bool	0	1	1
__gap	uint256[50]	1	0	1600
__gap	uint256[50]	51	0	1600
_roles	mapping(bytes32 => struct ACU.RoleData)	101	0	32
__gap	uint256[49]	102	0	1568
periodInSeconds	uint256	151	0	32
limitInWei	uint256	152	0	32
currentPeriodEnd	uint256	153	0	32
currentPeriodAmountInWei	uint256	154	0	32
__gap	uint256[10]	155	0	320
outboxL1L2MessageStatus	mapping(bytes32 => uint256)	165	0	32
inboxL2L1MessageStatus	mapping(bytes32 => uint256)	166	0	32
__gap_ReentrancyGuardUpgradeable	uint256[50]	167	0	1600

```
| pauseTypeStatuses                | mapping(bytes32 => bool)                | 217  | 0      | 32    |
| _pauseTypeStatusesBitMap         | uint256                                 | 218  | 0      | 32    |

# overwritten
| __gap                            | uint256[9]                              | 219  | 0      | 288   |

# identical
| nextMessageNumber                | uint256                                 | 228  | 0      | 32    |
| _messageSender                   | address                                 | 229  | 0      | 20    |
| __gap                            | uint256[50]                             | 230  | 0      | 1600  |
| currentTimestamp                 | uint256                                 | 280  | 0      | 32    |
| currentL2BlockNumber             | uint256                                 | 281  | 0      | 32    |
| stateRootHashes                  | mapping(uint256 => bytes32)             | 282  | 0      | 32    |
| verifiers                        | mapping(uint256 => address)             | 283  | 0      | 32    |
| __gap                            | uint256[50]                             | 284  | 0      | 1600  |
| rollingHashes                    | mapping(uint256 => bytes32)             | 334  | 0      | 32    |
| _messageClaimedBitMap            | struct BitMaps.BitMap                   | 335  | 0      | 32    |
| l2MerkleRootsDepths              | mapping(bytes32 => uint256)             | 336  | 0      | 32    |
| __gap_L1MessageManager           | uint256[50]                             | 337  | 0      | 1600  |
| systemMigrationBlock             | uint256                                 | 387  | 0      | 32    |
| __gap_L1MessageService           | uint256[50]                             | 388  | 0      | 1600  |
| dataFinalStateRootHashes         | mapping(bytes32 => bytes32)             | 438  | 0      | 32    |
| dataParents                      | mapping(bytes32 => bytes32)             | 439  | 0      | 32    |
| dataShnarfHashes                 | mapping(bytes32 => bytes32)             | 440  | 0      | 32    |
| dataStartingBlock                | mapping(bytes32 => uint256)             | 441  | 0      | 32    |
| dataEndingBlock                  | mapping(bytes32 => uint256)             | 442  | 0      | 32    |
| currentL2StoredL1MessageNumber   | uint256                                 | 443  | 0      | 32    |
| currentL2StoredL1RollingHash     | bytes32                                 | 444  | 0      | 32    |
| currentFinalizedShnarf           | bytes32                                 | 445  | 0      | 32    |

# renamed & re-purposed, existing data kept
| shnarfFinalBlockNumbers          | mapping(bytes32 => uint256)             | 446  | 0      | 32    |

# identical
| currentFinalizedState            | bytes32                                 | 447  | 0      | 32    |
```

- slot 219 which was previously a gap now stores `_pauseTypeRoles`
- slot 220 previously unused now stores `_unPauseTypeRoles`
- slot 446 `shnarfFinalBlockNumbers` is repurposed into `blobShnarfExists` but its existing data remains the same; for any given key a non-zero value indicates that shnarf exists. This new usage is compatible with its previous usage of mapping shnarfs to block numbers hence there is no need to wipe the data
- slot 448 previously unused now stores `fallbackOperator`

We have verified using `cast storage 0xd19d4B5d358258f05D7B411E21A1460D11B0876F` that the gap slots in the old storage currently deployed are already zero'd such that existing gaps can be safely repurposed in the new storage layout. The upgrade plan from the old `LineaRollup` storage layout to the new one appears safe.

L2MessageService Storage Upgrade

Some small changes required to support the more granular permission roles also occured in `L2MessageService`, hence we used `forge inspect -R "@openzeppelin/=contracts/node_modules/@openzeppelin/" --hardhat L2MessageService storageLayout --pretty` on both the new and old contracts to carefully examine their exact storage layout:

```
| NEW L2MessageService Layout      | Type                                    | Slot | Offset | Bytes |
|----------------------------------|-----------------------------------------|------|--------|-------|

# identical
| _initialized                     | uint8                                   | 0    | 0      | 1     |
```

_initializing	bool	0	1	1
__gap	uint256[50]	1	0	1600
__gap	uint256[50]	51	0	1600
_roles	mapping(bytes32 => struct ACU.RoleData)	101	0	32
__gap	uint256[49]	102	0	1568
periodInSeconds	uint256	151	0	32
limitInWei	uint256	152	0	32
currentPeriodEnd	uint256	153	0	32
currentPeriodAmountInWei	uint256	154	0	32
__gap	uint256[10]	155	0	320
pauseTypeStatuses	mapping(bytes32 => bool)	165	0	32
_pauseTypeStatusesBitMap	uint256	166	0	32

overwrites gap
| _pauseTypeRoles | mapping(enum PauseType => bytes32) | 167 | 0 | 32 |

new
| _unPauseTypeRoles | mapping(enum PauseType => bytes32) | 168 | 0 | 32 |
| __gap | uint256[7] | 169 | 0 | 224 |

identical
inboxL1L2MessageStatus	mapping(bytes32 => uint256)	176	0	32
_status	uint256	177	0	32
__gap	uint256[49]	178	0	1568
__gap_L2MessageService	uint256[50]	227	0	1600
_messageSender	address	277	0	20
nextMessageNumber	uint256	278	0	32
minimumFeeInWei	uint256	279	0	32
lastAnchoredL1MessageNumber	uint256	280	0	32
l1RollingHashes	mapping(uint256 => bytes32)	281	0	32
__gap_L2MessageManager	uint256[50]	282	0	1600
__gap_L2MessageService	uint256[50]	332	0	1600

OLD L2MessageService Layout	Type	Slot	Offset	Bytes

identical
_initialized	uint8	0	0	1
_initializing	bool	0	1	1
__gap	uint256[50]	1	0	1600
__gap	uint256[50]	51	0	1600
_roles	mapping(bytes32 => struct ACU.RoleData)	101	0	32
__gap	uint256[49]	102	0	1568
periodInSeconds	uint256	151	0	32
limitInWei	uint256	152	0	32
currentPeriodEnd	uint256	153	0	32
currentPeriodAmountInWei	uint256	154	0	32
__gap	uint256[10]	155	0	320
pauseTypeStatuses	mapping(bytes32 => bool)	165	0	32
_pauseTypeStatusesBitMap	uint256	166	0	32

overwritten
| __gap | uint256[9] | 167 | 0 | 288 |

identical
inboxL1L2MessageStatus	mapping(bytes32 => uint256)	176	0	32
_status	uint256	177	0	32
__gap	uint256[49]	178	0	1568
__gap_L2MessageService	uint256[50]	227	0	1600
_messageSender	address	277	0	20

```
| nextMessageNumber                | uint256                                | 278 | 0 | 32   |
| minimumFeeInWei                  | uint256                                | 279 | 0 | 32   |
| lastAnchoredL1MessageNumber      | uint256                                | 280 | 0 | 32   |
| l1RollingHashes                  | mapping(uint256 => bytes32)            | 281 | 0 | 32   |
| __gap_L2MessageManager           | uint256[50]                            | 282 | 0 | 1600 |
| __gap_L2MessageService           | uint256[50]                            | 332 | 0 | 1600 |
```

- slot 167 which was previously a gap now stores `_pauseTypeRoles`
- slot 168 previously unused now stores `_unPauseTypeRoles`

We have verified using `cast storage 0x508Ca82Df566dCD1B0DE8296e70a96332cD644ec` that the gap slots in the old storage currently deployed are already zero'd such that existing gaps can be safely repurposed in the new storage layout. The upgrade plan from the old `L2MessageService` storage layout to the new one appears safe.

Optimized Message Hashing

One of the new features is the addition of `messageService/lib/MessageHashing::_hashMessage` which uses assembly to hash message parameters instead of the previous `keccak256(abi.encode(...))`. The protocol uses optimized hashing in a number of places so we created a test suite using Foundry stateless fuzzing and Halmos symbolic execution to verify the optimized hashing produces equivalent output to normal methods.

First add Foundry to Hardhat:

```
cd contracts
pnpm install @nomicfoundation/hardhat-foundry

add to top of hardhat.config.ts: require("@nomicfoundation/hardhat-foundry");
```

Then create the initial `foundry.toml` config using `npx hardhat init-foundry` and add to it:

```
evm_version = 'cancun'
optimizer = true
optimizer_runs = 10_000
no-match-path = '{contracts/test-contracts/*,contracts/tokenBridge/mocks/*}'

[fuzz]
runs = 10_000
```

The test file can be put into a new folder `test/foundry`:

```
// SPDX-License-Identifier: AGPL-3.0
pragma solidity 0.8.26;

import {Utils} from "../../contracts/lib/Utils.sol";
import {MessageHashing} from "../../contracts/messageService/lib/MessageHashing.sol";
import {LineaRollup} from "../../contracts/LineaRollup.sol";

import {Test} from "forge-std/Test.sol";

// run from base project directory with:
// (fuzz test) forge test --match-contract OptimizedKeccakTest
// (halmos)    halmos --function test --match-contract OptimizedKeccakTest
contract OptimizedKeccakTest is Test, LineaRollup {
    // Utils::_efficientKeccak
    function _normalKeccak(bytes32 a, bytes32 b) private pure returns (bytes32 output) {
        output = keccak256(abi.encode(a, b));
    }

    function test_OptimizedKeccak(bytes32 a, bytes32 b) external pure {
```

```
    bytes32 output1 = _normalKeccak(a, b);
    bytes32 output2 = Utils._efficientKeccak(a, b);

    assertEq(output1, output2);
}

function _normalKeccak(uint256 a, address b) private pure returns (bytes32 output) {
    output = keccak256(abi.encode(a, b));
}

function test_OptimizedKeccak(uint256 a, address b) external pure {
    bytes32 output1 = _normalKeccak(a, b);
    bytes32 output2 = Utils._efficientKeccak(a, b);

    assertEq(output1, output2);
}


// MessageHashing::_hashMessage
function _normalHashMessage(
    address from, address to, uint256 fee,
    uint256 valueSent, uint256 messageNumber, bytes calldata _calldata)
private pure returns (bytes32 output) {
    output = keccak256(abi.encode(from, to, fee, valueSent, messageNumber, _calldata));
}

function test_OptimizedMessageHashing(
    address from, address to, uint256 fee,
    uint256 valueSent, uint256 messageNumber, bytes calldata _calldata)
external pure {
    bytes32 output1 = _normalHashMessage(from, to, fee, valueSent, messageNumber, _calldata);
    bytes32 output2 = MessageHashing._hashMessage(from, to, fee, valueSent, messageNumber,
    ↪  _calldata);

    assertEq(output1, output2);
}


// LineaRollup::_computeLastFinalizedState
function _normalComputeLastFinalizedState(
    uint256 _messageNumber, bytes32 _rollingHash, uint256 _timestamp)
private pure returns (bytes32 output) {
    output = keccak256(abi.encode(_messageNumber, _rollingHash, _timestamp));
}

function test_ComputeLastFinalizedState(
    uint256 _messageNumber, bytes32 _rollingHash, uint256 _timestamp)
external pure {
    bytes32 output1 = _normalComputeLastFinalizedState(_messageNumber, _rollingHash, _timestamp);
    bytes32 output2 = _computeLastFinalizedState(_messageNumber, _rollingHash, _timestamp);

    assertEq(output1, output2);
}


// LineaRollup::_computeShnarf
function _normalComputeShnarf(
    bytes32 _parentShnarf, bytes32 _snarkHash, bytes32 _finalStateRootHash,
    bytes32 _dataEvaluationPoint, bytes32 _dataEvaluationClaim)
private pure returns (bytes32 output) {
    output = keccak256(
```

```
            abi.encode(_parentShnarf, _snarkHash, _finalStateRootHash, _dataEvaluationPoint,
            ↪  _dataEvaluationClaim));
    }

    function test_ComputeShnarf(
        bytes32 _parentShnarf, bytes32 _snarkHash, bytes32 _finalStateRootHash,
        bytes32 _dataEvaluationPoint, bytes32 _dataEvaluationClaim)
    external pure {
        bytes32 output1 = _normalComputeShnarf(
            _parentShnarf, _snarkHash, _finalStateRootHash, _dataEvaluationPoint, _dataEvaluationClaim);
        bytes32 output2 = _computeShnarf(
            _parentShnarf, _snarkHash, _finalStateRootHash, _dataEvaluationPoint, _dataEvaluationClaim);

        assertEq(output1, output2);
    }


    // LineaRollup::_computePublicInput (requires changing it to internal)
    function _normalComputePublicInput(
        FinalizationDataV3 calldata _finalizationData, bytes32 _lastFinalizedShnarf,
        bytes32 _finalShnarf, uint256 _lastFinalizedBlockNumber, uint256 _endBlockNumber)
    private pure returns (uint256 output) {
        bytes32 interim =
            keccak256(bytes.concat(
                        abi.encode(_lastFinalizedShnarf, _finalShnarf,
                                _finalizationData.lastFinalizedTimestamp,
                                ↪  _finalizationData.finalTimestamp,
                                _lastFinalizedBlockNumber, _endBlockNumber,
                                _finalizationData.lastFinalizedL1RollingHash,
                                ↪  _finalizationData.l1RollingHash),
                        abi.encode(_finalizationData.lastFinalizedL1RollingHashMessageNumber,
                                _finalizationData.l1RollingHashMessageNumber,
                                _finalizationData.l2MerkleTreesDepth,
                                ↪  keccak256(abi.encodePacked(_finalizationData.l2MerkleRoots)))));

        output = uint256(interim) % MODULO_R;
    }

    function test_ComputePublicInput(
        FinalizationDataV3 calldata _finalizationData, bytes32 _lastFinalizedShnarf,
        bytes32 _finalShnarf, uint256 _lastFinalizedBlockNumber, uint256 _endBlockNumber)
    external pure {
        uint256 output1 = _normalComputePublicInput(
            _finalizationData, _lastFinalizedShnarf, _finalShnarf, _lastFinalizedBlockNumber,
            ↪  _endBlockNumber);
        uint256 output2 = _computePublicInput(
            _finalizationData, _lastFinalizedShnarf, _finalShnarf, _lastFinalizedBlockNumber,
            ↪  _endBlockNumber);

        assertEq(output1, output2);
    }
}
```

Other larger testing contracts were also supplied to the protocol team as part of the audit.

Summary

Project Name	Linea
Repository	linea-monorepo
Commit	5fad7f793cc1...
Audit Timeline	Oct 21st - Nov 4th, 2024
Methods	Manual Review

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	1
Informational	5
Gas Optimizations	5
Total Issues	12

Summary of Findings

[M-1] Operator can finalize for non-existent `finalShnarf`	Resolved
[L-1] Operator can submit data via `LineaRollup::submitDataAsCalldata` for invalid parent shnarf	Resolved
[I-1] Use SafeCast or document assumption that unsafe downcast in `SparkeMerkleTreeVerifier::_verifyMerkleProof` can't overflow	Resolved
[I-2] Mark `L1MessageManagerV1::outboxL1L2MessageStatus` as deprecated	Resolved
[I-3] Remove comments which no longer apply	Resolved
[I-4] Use named mappings in `TokenBridge` and remove obsolete comments	Resolved
[I-5] `L2MessageService::reinitializePauseTypesAndPermissions` should use `reinitializer(2)`	Resolved
[G-1] Remove redundant `L2MessageManagerV1::__L2MessageManager_init` and associated constant	Resolved
[G-2] Cheaper to not cache `calldata` array length	Resolved
[G-3] Use named return variables to save at least 9 gas per variable	Resolved
[G-4] Cache storage variables to avoid multiple identical storage reads	Resolved
[G-5] Fail fast in `LineaRollup::submitBlobs` and `submitDataAsCalldata`	Resolved

7 Findings

7.1 Medium Risk

7.1.1 Operator can finalize for non-existent `finalShnarf`

Description: In the previous version of `LineaRollup::_finalizeBlocks` there was this check which ensured that the final shnarf was associated with a block number:

```
if (
  shnarfFinalBlockNumbers[_finalizationData.finalSubmissionData.shnarf] !=
  _finalizationData.finalSubmissionData.finalBlockInData
) {
  revert FinalBlockDoesNotMatchShnarfFinalBlock(
    _finalizationData.finalSubmissionData.finalBlockInData,
    shnarfFinalBlockNumbers[_finalizationData.finalSubmissionData.shnarf]
  );
}
```

In the new version `shnarfFinalBlockNumbers` was changed to `blobShnarfExists` which links a shnarf to an effective boolean flag (though as an uint due to previous definition), and the above check was removed but no similar check was implemented.

Impact: An operator can finalize for non-existent `finalShnarf`.

Recommended Mitigation: Add an equivalent check in `LineaRollup::_finalizeBlocks` to verify that the computed `finalShnarf` exists:

```
finalShnarf = _computeShnarf(
  _finalizationData.shnarfData.parentShnarf,
  _finalizationData.shnarfData.snarkHash,
  _finalizationData.shnarfData.finalStateRootHash,
  _finalizationData.shnarfData.dataEvaluationPoint,
  _finalizationData.shnarfData.dataEvaluationClaim
);

// @audit prevent finalization for non-existent final shnarf
if(blobShnarfExists[finalShnarf] == 0) revert FinalBlobNotSubmitted();
```

Linea: Fixed in PR226 commit 4286bdb.

Cyfrin: Verified.

7.2 Low Risk

7.2.1 Operator can submit data via `LineaRollup::submitDataAsCalldata` for invalid parent shnarf

Description: `LineaRollup::submitBlobs` has this check to validate the parent shnarf exists:

```
if (blobShnarfExists[_parentShnarf] == 0) {
  revert ParentBlobNotSubmitted(_parentShnarf);
}
```

But `LineaRollup::submitDataAsCalldata` has no similar check, meaning that an operator can submit data for an invalid parent shnarf by calling `submitDataAsCalldata`.

Linea: Fixed in PR223 commit 8800eaa.

Cyfrin: Verified.

7.3 Informational

7.3.1 Use SafeCast or document assumption that unsafe downcast in `SparkeMerkleTreeVerifier::_verifyMerkleProof` can't overflow

Description: `SparkeMerkleTreeVerifier::_verifyMerkleProof` has added the following sanity check:

```
uint32 maxAllowedIndex = uint32((2 ** _proof.length) - 1);
if (_leafIndex > maxAllowedIndex) {
  revert LeafIndexOutOfBounds(_leafIndex, maxAllowedIndex);
}
```

If `_proof.length` > 32 this would overflow as in Solidity casts don't revert but overflow.

The team has stated that *"it is based on the Merkle tree depth coming from the finalization where the length is checked against the depth. That currently is set at 5 and is unlikely to change".*

So the overflow appears to be impossible in practice, however we recommend either:

- using SafeCast to revert if an overflow did occur
- explicitly documenting the assumption that an overflow can't occur in the code

The risk with the comment approach is that in the future the related code in finalization can be changed without the dev realizing that would trigger an overflow in this place.

Linea: Fixed in PR222 commits c20b938 & 77a6e99.

Cyfrin: Verified.

7.3.2 Mark `L1MessageManagerV1::outboxL1L2MessageStatus` as deprecated

Description: `L1MessageManagerV1::outboxL1L2MessageStatus` is never read or written to anymore apart from the test suite:

```
$ rg "outboxL1L2MessageStatus"
test-contracts/TestL1MessageManager.sol
31:    outboxL1L2MessageStatus[_messageHash] = OUTBOX_STATUS_SENT;
39:      uint256 existingStatus = outboxL1L2MessageStatus[messageHash];
46:        outboxL1L2MessageStatus[messageHash] = OUTBOX_STATUS_RECEIVED;

messageService/l1/v1/L1MessageManagerV1.sol
22:  mapping(bytes32 messageHash => uint256 messageStatus) public outboxL1L2MessageStatus;

test-contracts/LineaRollupV5.sol
1944:  mapping(bytes32 messageHash => uint256 messageStatus) public outboxL1L2MessageStatus;

test-contracts/LineaRollupAlphaV3.sol
2011:  mapping(bytes32 messageHash => uint256 messageStatus) public outboxL1L2MessageStatus;

tokenBridge/mocks/MessageBridgeV2/MockMessageServiceV2.sol
40:    outboxL1L2MessageStatus[messageHash] = OUTBOX_STATUS_SENT;
```

Therefore it should be marked as deprecated with a comment similar to how `LineaRollup` handles its deprecated mappings.

Similarly `L1MessageManagerV1::inboxL2L1MessageStatus` is only ever deleted from but no new mappings are inserted; ideally a comment should also indicate this.

Linea: Fixed in PR256 commit ac51e9e.

Cyfrin: Verified.

7.3.3 Remove comments which no longer apply

Description: Comments which no longer apply should be removed as they are now misleading.

File: `L1MessageServiceV1.sol`

```
// @audit `claimMessage` no longer uses `_messageSender` so these comments are incorrect
   * @dev _messageSender is set temporarily when claiming and reset post. Used in sender().
   * @dev _messageSender is reset to DEFAULT_SENDER_ADDRESS to be more gas efficient.
```

File: `L1MessageService.sol`

```
// @audit `_messageSender` no longer initialized as it is not used anymore by L1 Messaging
   * @dev _messageSender is initialised to a non-zero value for gas efficiency on claiming.
```

Linea: Fixed in PR256 commits ac51e9e & b875723.

Cyfrin: Verified.

7.3.4 Use named mappings in `TokenBridge` and remove obsolete comments

Description: `TokenBridge` should use named mappings and remove obsolete comments:

```
-   /// @notice mapping (chainId => nativeTokenAddress => brigedTokenAddress)
-   mapping(uint256 => mapping(address => address)) public nativeToBridgedToken;
-   /// @notice mapping (brigedTokenAddress => nativeTokenAddress)
-   mapping(address => address) public bridgedToNativeToken;

+   mapping(uint256 chainId => mapping(address native => address bridged)) public nativeToBridgedToken;
+   mapping(address bridged => address native) public bridgedToNativeToken;
```

Linea: Fixed in PR256 commit ac51e9e.

Cyfrin: Verified.

7.3.5 `L2MessageService::reinitializePauseTypesAndPermissions` should use `reinitializer(2)`

Description: `TokenBridge::reinitializePauseTypesAndPermissions` uses `reinitializer(2)` because:

```
export ETH_RPC_URL=mainnet_rpc
cast storage 0x051F1D88f0aF5763fB888eC4378b4D8B29ea3319 0
0x0000000000000000000000000000000000000000000000000000000000000001
export ETH_RPC_URL=linea_rpc
cast storage 0x353012dc4a9A6cF55c941bADC267f82004A8ceB9 0
0x0000000000000000000000000000000000000000000000000000000000000001
```

`LineaRollup::reinitializeLineaRollupV6` uses `reinitializer(6)` because:

```
export ETH_RPC_URL=mainnet_rpc
cast storage 0xd19d4B5d358258f05D7B411E21A1460D11B0876F 0
0x0000000000000000000000000000000000000000000000000000000000000005
```

But `L2MessageService::reinitializePauseTypesAndPermissions` uses `reinitializer(6)` even though:

```
export ETH_RPC_URL=linea_rpc
cast storage 0x508Ca82Df566dCD1B0DE8296e70a96332cD644ec 0
```

```
0x0000000000000000000000000000000000000000000000000000000000000001
```

For consistency `L2MessageService::reinitializePauseTypesAndPermissions` should use `reinitializer(2)`.

Linea: Fixed in PR271 commit 53f43d3.

Cyfrin: Verified.

7.4 Gas Optimization

7.4.1 Remove redundant `L2MessageManagerV1::__L2MessageManager_init` and associated constant

Description: `L2MessageManagerV1::__L2MessageManager_init` is no longer called by `L2MessageService::initialize` which uses the new `PermissionsManager` contract.

Hence it should be removed along with its associated constant `L1_L2_MESSAGE_SETTER_ROLE`. This constant is referenced in comments throughout the code so those should also be updated.

The test suite still contains calls to `L2MessageManagerV1::__L2MessageManager_init`; the test suite should also be updated use only the new method for initialisation.

```
// output of: rg "__L2MessageManager_init"
messageService/l2/v1/L2MessageManagerV1.sol
39:  function __L2MessageManager_init(address _l1l2MessageSetter) internal onlyInitializing {

test-contracts/TestL2MessageManager.sol
32:    __L2MessageManager_init(_l1l2MessageSetter);
41:    __L2MessageManager_init(_l1l2MessageSetter);

test-contracts/L2MessageServiceLineaMainnet.sol
1620:  function __L2MessageManager_init(address _l1l2MessageSetter) internal onlyInitializing {
1992:    __L2MessageManager_init(_l1l2MessageSetter);

// output of: rg "L1_L2_MESSAGE_SETTER_ROLE"
messageService/l2/L2MessageManager.sol
26:   * @dev Only address that has the role 'L1_L2_MESSAGE_SETTER_ROLE' are allowed to call this
↪  function.
40:  ) external whenTypeNotPaused(PauseType.GENERAL) onlyRole(L1_L2_MESSAGE_SETTER_ROLE) {

messageService/l2/v1/L2MessageManagerV1.sol
18:  bytes32 public constant L1_L2_MESSAGE_SETTER_ROLE = keccak256("L1_L2_MESSAGE_SETTER_ROLE");
37:   * @param _l1l2MessageSetter The address owning the L1_L2_MESSAGE_SETTER_ROLE role.
40:    _grantRole(L1_L2_MESSAGE_SETTER_ROLE, _l1l2MessageSetter);

interfaces/l2/IL2MessageManager.sol
48:   * @dev Only address that has the role 'L1_L2_MESSAGE_SETTER_ROLE' are allowed to call this
↪  function.

test-contracts/L2MessageServiceLineaMainnet.sol
1601:  bytes32 public constant L1_L2_MESSAGE_SETTER_ROLE = keccak256("L1_L2_MESSAGE_SETTER_ROLE");
1618:   * @param _l1l2MessageSetter The address owning the L1_L2_MESSAGE_SETTER_ROLE role.
1621:    _grantRole(L1_L2_MESSAGE_SETTER_ROLE, _l1l2MessageSetter);
1626:   * @dev Only address that has the role 'L1_L2_MESSAGE_SETTER_ROLE' are allowed to call this
↪  function.
1629:  function addL1L2MessageHashes(bytes32[] calldata _messageHashes) external
↪  onlyRole(L1_L2_MESSAGE_SETTER_ROLE) {
```

Linea: Fixed in PR212 commit 3b30a8a.

Cyfrin: Verified.

7.4.2 Cheaper to not cache `calldata` array length

Description: When an array is passed as `calldata` it is cheaper not to cache the length:

```
// PermissionsManager::__Permissions_init
  function __Permissions_init(RoleAddress[] calldata _roleAddresses) internal onlyInitializing {
-     uint256 roleAddressesLength = _roleAddresses.length;

-     for (uint256 i; i < roleAddressesLength; i++) {
```

```
+    for (uint256 i; i < _roleAddresses.length; i++) {
```

The same applies to:

- `PauseManager::__PauseManager_init`
- `LineaRollup::submitBlobs`
- `L2MessageManager::anchorL1L2MessageHashes`

Linea: Fixed in PR247 commits 8bf9d86, 0ffc752 & commit 968b257.

Cyfrin: Verified.

7.4.3 Use named return variables to save at least 9 gas per variable

Description: Using named return variables saves at least 9 gas per variable; named returns are already used in some functions of the protocol but not in others:

```
PauseManager.sol
136:  function isPaused(PauseType _pauseType) public view returns (bool)

l1/L1MessageManager.sol
98:  function isMessageClaimed(uint256 _messageNumber) external view returns (bool) {

l1/L1MessageService.sol
150:  function sender() external view returns (address addr) {

l2/v1/L2MessageServiceV1.sol
165:  function sender() external view returns (address) {

lib/SparseMerkleTreeVerifier.sol
32:  ) internal pure returns (bool) {

TokenBridge.sol
 function _safeName(address _token) internal view returns (string memory) {
  function _safeSymbol(address _token) internal view returns (string memory) {
  function _safeDecimals(address _token) internal view returns (uint8) {
  function _returnDataToString(bytes memory _data) internal pure returns (string memory) {
```

Linea: Fixed in PR247 commit 968b257.

Cyfrin: Verified.

7.4.4 Cache storage variables to avoid multiple identical storage reads

Description: Cache storage variables to avoid multiple identical storage reads:

File: `TokenBridge.sol`

```
// @audit use `_initializationData.sourceChainId` instead of `sourceChainId`
147:        nativeToBridgedToken[sourceChainId][_initializationData.reservedTokens[i]] =
↪  RESERVED_STATUS;

// @audit cache 'sourceChainId' from storage and use cached copy
371:        nativeToBridgedToken[sourceChainId][_nativeTokens[i]] = DEPLOYED_STATUS;
```

Linea: Fixed in PR247 commit 968b257.

Cyfrin: Verified.

7.4.5 Fail fast in `LineaRollup::submitBlobs` and `submitDataAsCalldata`

Description: `LineaRollup::submitBlobs` does a lot of processing then after the `for` loop there is this first check which ensures the computed shnarf matches the provided expected input:

```
if (_finalBlobShnarf != computedShnarf) {
  revert FinalShnarfWrong(_finalBlobShnarf, computedShnarf);
}
```

If the first check did not revert, this means that `_finalBlobShnarf == computedShnarf`.

Then a second check reverts if this schnarf already exists:

```
if (blobShnarfExists[computedShnarf] != 0) {
  revert DataAlreadySubmitted(computedShnarf);
}
```

But since the second check can only execute if `_finalBlobShnarf == computedShnarf`, it is much more efficient to delete the second check and put a new check at the beginning of the function like this:

```
if (blobShnarfExists[_finalBlobShnarf] != 0) {
  revert DataAlreadySubmitted(_finalBlobShnarf);
}
```

Ideally the beginning of the function would have these 4 checks before doing or declaring anything else:

```
  function submitBlobs(
    BlobSubmission[] calldata _blobSubmissions,
    bytes32 _parentShnarf,
    bytes32 _finalBlobShnarf
  ) external whenTypeAndGeneralNotPaused(PauseType.BLOB_SUBMISSION) onlyRole(OPERATOR_ROLE) {
    uint256 blobSubmissionLength = _blobSubmissions.length;

    if (blobSubmissionLength == 0) {
      revert BlobSubmissionDataIsMissing();
    }

    if (blobhash(blobSubmissionLength) != EMPTY_HASH) {
      revert BlobSubmissionDataEmpty(blobSubmissionLength);
    }

    if (blobShnarfExists[_parentShnarf] == 0) {
      revert ParentBlobNotSubmitted(_parentShnarf);
    }

    if (blobShnarfExists[_finalBlobShnarf] != 0) {
      revert DataAlreadySubmitted(_finalBlobShnarf);
    }

    // variable declarations and processing follow
```

The same applies in `submitDataAsCalldata`:

```
  function submitDataAsCalldata(
    CompressedCalldataSubmission calldata _submission,
    bytes32 _parentShnarf,
    bytes32 _expectedShnarf
  ) external whenTypeAndGeneralNotPaused(PauseType.CALLDATA_SUBMISSION) onlyRole(OPERATOR_ROLE) {
```

```
    if (_submission.compressedData.length == 0) {
      revert EmptySubmissionData();
    }

    if (blobShnarfExists[_expectedShnarf] != 0) {
      revert DataAlreadySubmitted(_expectedShnarf);
    }

    // ...
```

Linea: Fixed in PR247 commit 968b257.

Cyfrin: Verified.