Report

v. 1.0

Customer

zkLink

Smart contract Audit

Cost Optimisation

28th June 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	27.06.24	A. Zveryanskaya	Initial Draft
0.2	28.06.24	A. Zveryanskaya	Minor revision
1.0	28.06.24	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

zkLink is building an aggregated rollup infrastructure, secured by zero-knowledge proofs and multi-chain state synchronization, to aggregate and unify the liquidity from various Layer 1 and Layer 2 networks. zkLink Nova is an Aggregated Layer 3 Rollup zkEVM network built with ZK Stack and zkLink Nexus. Users on Nova have immediate access to the aggregated assets from Ethereum and integrated Ethereum Layer 2 networks. Nova inherits Ethereum's security by achieving multi-chain state synchronization through canonical rollup bridges.

3 Project scope

We're asked to review zkLinkProtocol and Era contracts diff:

l1-contracts/contracts/zksync/facets/		
	Mailbox.sol	
l1-contracts/contracts/zksync/interfaces/		
	IMailbox.sol	IZkLink.sol
l1-contracts/contracts/zksync/libraries/		
	Merkle.sol	

Then we reviewed the synchronization cost optimization and the included files:

/		
	Arbitrator.sol	ZkLink.sol
interfaces/		
	IArbitrator.sol	IZkLink.sol
gateway/ethereum/		
	EthereumGateway.sol	
zksync/l1-contracts/zksync/libraries/		
	Merkle.sol	

The fixes were provided at the following link.

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We provided Client with a few recommendations.

	Info	Fixed
Moderate	2	2

Fixed 2 out of 4 Moderate issues

6 Moderate Issues

CVF-1. INFO

- **Category** Suboptimal
- **Source** Mailbox.sol

Description The "_forwardEthAmount" value is passed into the "sendMessage" call three times: once as a separate argument, another time as an encoded part of "gatewayDataList[0]", and once again as a part of "callData", encoded into :gatewayDataList[0]". This seems suboptimal.

Recommendation Consider refactoring.

Client Comment *We have designed a common message format for message transmission between the primary chain and the secondary chains. The message consists of three parts:*

- *l1GatewayAddress: used to indicate which target chain to send to*
- *value: used to indicate the eth carried by the message*
- *callData: message content*

The secondary chain will only send messages to the primary chain, so l1GatewayAddress is omitted from the secondary chain's message. The primary chain can send messages to multiple secondary chains, so messages from the primary chain all carry l1GatewayAddress. Since the primary chain can send messages to multiple secondary chains at the same time, the message sent by the primary chain is a list. Considering that callData is a bytes of unknown length, we use abi.encode instead of designing an encoding protocol with a length parameter. This is beneficial to both the readability of the contract and the implementation of the front-end and back-end. It is necessary for value to be used as a parameter alone. For example, when the primary chain sends multiple messages to the secondary chain, value is the sum of eth carried by all secondary chain messages.

```
212  +bytes memory callData = abi.encodeCall(
212 213      IZkLink.syncBatchRoot,
213 214      (_batchNumber, l2LogsRootHash, _forwardEthAmount)
214 215  );
```

```
216  +gatewayDataList[0] = abi.encode(_secondaryChainGateway,
        ↪ _forwardEthAmount, callData);
```

```
218  +s.gateway.sendMessage{value: msg.value + _forwardEthAmount}(
        ↪ _forwardEthAmount, abi.encode(gatewayDataList));
```

CVF-2. INFO

- **Category** Suboptimal
- **Source** Mailbox.sol

Description Nesting ABI encoding calls are inefficient, as each nesting level allocates new memory and copies data into it.

Recommendation Consider refactoring to construct the final call data in one go.

Client Comment *Same as CVF-1.*

```
212 +bytes memory callData = abi.encodeCall(
```

```
216 +gatewayDataList[0] = abi.encode(_secondaryChainGateway,
    ↪ _forwardEthAmount, callData);
```

```
218 +s.gateway.sendMessage{value: msg.value + _forwardEthAmount}(
    ↪ _forwardEthAmount, abi.encode(gatewayDataList));
```

```
254 +        bytes memory gatewayCallData = abi.encodeCall(
```

```
258 +        gatewayDataList[i] = abi.encode(_secondaryChainGateway,
    ↪ _forwardEthAmount, gatewayCallData);
```

```
272 +    abi.encode(gatewayDataList)
```

```
283 +bytes memory callData = abi.encodeCall(IZkLink.syncL2TxHash, (op.
    ↪ canonicalTxHash, _l2TxHash));
284 +gatewayDataList[0] = abi.encode(op.gateway, 0, callData);
```

```
286 +s.gateway.sendMessage{value: msg.value}(0, abi.encode(
    ↪ gatewayDataList));
```

CVF-3. FIXED

- **Category** Overflow/Underflow
- **Source** Mailbox.sol

Description Impossibility of overflow is not clearly guaranteed by the surrounding code. Even if overall business logic is supposed to guarantees it, the logic could have bugs.

Recommendation Consider using safe math.

```
250 +// Withdraw eth amount impossible overflow
251 +totalForwardEthAmount += _forwardEthAmount;
```

CVF-4. FIXED

- **Category** Flaw
- **Source** ZkLink.sol

Description There is no check to ensure that _fromBatchNumber <= totalBatchesExecuted, i.e. that there is no gap between already executed batches and the new executed range.

```
513 +totalBatchesExecuted = _toBatchNumber;
```

7 Recommendations

CVF-5. FIXED

- **Category** Suboptimal
- **Source** Mailbox.sol

Description The storage slot address of "s.secondaryChains[_secondaryChainGateway].totalPendingWithdraw" is calcualted twice.

Recommendation Consider calculating once and reusing.

```
249 +uint256 _forwardEthAmount = s.secondaryChains[
       ↪ _secondaryChainGateway].totalPendingWithdraw;
```

```
252 +s.secondaryChains[_secondaryChainGateway].totalPendingWithdraw = 0;
```

CVF-6. INFO

- **Category** Suboptimal
- **Source** Arbitrator.sol

Recommendation This check makes the "_value" argument redundant.

Client Comment *The existence of parameter value is necessary. We need to deal with the situation where msg.value contains fee.*

```
159 +require(msg.value == _value, "Invalid␣msg␣value");
```

CVF-7. INFO

- **Category** Suboptimal
- **Source** Arbitrator.sol

Recommendation It would be more efficient to pack a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *We are not going to modify.*

```
223 +bytes[] memory gatewayDataList = abi.decode(_receiveCallData, (
       ↪ bytes[]));
224 +bytes[] memory gatewayForwardParamsList = abi.decode(_forwardParams
       ↪ , (bytes[]));
```

CVF-8. INFO

- **Category** Suboptimal
- **Source** ZkLink.sol

Recommendation If batch numbers don't exceed 128 bits, it would be more efficient to just pack two batch numbers into a single word as use this word as a key.

Client Comment *We are not going to modify.*

```
88 +/// The key is keccak256(abi.encodePacked(fromBatchNumber,
      ↪ toBatchNumber))
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting