Report

v. 1.0

Customer

zkLink

Smart Contract Audit

zkLink Protocol Phase II

9th August 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	09.08.23	A. Zveryanskaya	Initial Draft
0.2	09.08.23	A. Zveryanskaya	Minor revision
1.0	09.08.23	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 a trading-focused multi-chain L2 network with unified liquidity secured by ZK-Rollups.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	EmptyVerifier.sol	Storage.sol	ZkLink.sol
	ZkLinkPeriphery.sol		
interfaces/			
	IVerifier.sol		
zksync/			
	Config.sol	Events.sol	Operations.sol
	Verifier.sol		

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.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 1 major, and a few less important issues. All identified Major issues have been fixed.

	Active	Fixed
Major	0	1
Minor	1	2

Fixed 3 out of 4 issues

6 Major Issues

CVF-1. FIXED

- **Category** Procedural
- **Source** Operations.sol

Recommendation One should always abbreviate Layer 2 as “L2”, rather than “l2”, as the latter looks identical to “12” in some fonts.

```
112 +//bytes12 addressPrefixZero; -- address bytes length in l2 is 32
163 +//bytes12 addressPrefixZero; -- address bytes length in l2 is 32
198 +//bytes12 addressPrefixZero; -- address bytes length in l2 is 32
225 +//bytes12 addressPrefixZero; -- address bytes length in l2 is 32
```

7 Minor Issues

CVF-2. INFO

- **Category** Suboptimal
- **Source** Config.sol

Description In EVM zero bytes are in many cases cheaper than non-zero bytes.

Recommendation Consider using a value with most bytes set to zero. Like 0xFF.

Client Comment *We won't modify it.*

```
107 +bytes32 internal constant GLOBAL_ASSET_ACCOUNT_ADDRESS =
    0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff;
```

CVF-3. FIXED

- **Category** Documentation
- **Source** Events.sol

Description It is unclear why the decimals property is needed here.

Recommendation Consider explaining or removing.

```
54 +event NewToken(uint16 indexed tokenId, address indexed token, uint8
      ↪ decimals);
```

CVF-4. FIXED

- **Category** Documentation
- **Source** Storage.sol

Recommendation Consider explaining why overflow is not possible here.

```
187 +pendingBalances[_address][_tokenId] = balance + _amount;
```

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